

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

Java EE

XML

# E-Book

## Alle Themen der Website!



Copyright 2013-2017 by Homepage-Webhilfe, Benjamin Jung - All rights reserved!

Webseite: <https://www.homepage-webhilfe.de/>

E-Mail: [kontakt@homepage-webhilfe.de](mailto:kontakt@homepage-webhilfe.de)

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

Java EE

XML

# E-Book

## Allgemeines über Webseiten





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Allgemein](#) » [Einstieg](#)

## Einstieg

Als Anfänger und Einsteiger im Gebiet der Webseiten hat man am Anfang viele Fragen. Das große Problem dabei ist, dass es im Internet zwar viele Antworten zu konkreten Fragen gibt, jedoch zumeist keine Einstiegs-Tutorial für Anfänger. Doch wenn man erstmals nach konkreten Fragen gesucht hat, hat man schon den ersten Fehler begangen. Viel besser ist es, wenn man ein gutes Einstiegs-Tutorial liest und im Nachhinein noch nach Antworten zu offenen und konkreten Fragen sucht. Und hier kommt unsere Website ins Spiel.

Hier auf Homepage-Webhilfe wollen wir möglichst viele Informationen zum Thema Webseiten bereitstellen. Das Kapitel „Allgemein“ ist das perfekte Einstiegs-Tutorial für alle Anfänger. Und das Beste ist: Wenn Sie später noch nach [Software](#), [Provider](#) oder Tutorials für Programmiersprachen suchen, werden Sie hier auf dieser Website fündig. Und zum Thema offene Fragen können wir Einsteigern nur empfehlen, auf unserer Website in den [FAQs](#) und im [Glossar](#) nachzuschauen. Dieser beantworten wir Ihre Fragen auch im [Forum](#) oder per E-Mail ([Beratungsformular](#)). Doch nun genug geredet: Jetzt wollen wir endlich anfangen alle Grundlagen zum Thema Webseiten kennenlernen.

Viel Spaß und Erfolg wünscht Ihnen Ihr Homepage-Webhilfe-Team.

## Idee



Bevor wir eine Website erstellen wollen, benötigen wir eine Idee. Dafür gibt es natürlich ein paar grundlegende Fragen: Was wollen wir auf der Website teilen? An wen richtet sich der Inhalt der Website? Was für eine Art von Website soll es sein? Was ist der Grund für die Erstellung einer Website? Wie soll die Webseite erstellt werden? Was muss ich noch alles lernen, um dieses Vorhaben realisieren zu können?

Viele Fragen und erstmals wenige Antworten. Das ist ganz normal. Wir werden im Laufe dieses Kapitels auf die Fragen noch genauer eingehen. Doch vorab sollten Sie sich schon einmal ein paar Gedanken machen. Lesen Sie sich diese Seite vollständig durch, machen Sie sich ein paar Gedanken, schlafen Sie eine Nacht darüber und dann können wir schon morgen anfangen, unsere Gedanken auszubauen, zu erweitern und beginnen ein Konzept aufzustellen.

## Spaß und Hingabe

Ganz wichtig beim Erstellen einer Website ist natürlich, dass wir Spaß am Erstellen der Website haben. Dies ist in manchen Fällen jedoch leider nicht immer gegeben, z. B. wenn wir eine Website für die eigene Firma für unsere Kunden erstellen müssen. Doch genau in solchen Fällen sollte man überlegen, ob man die Website dann nicht eher von einer Webagentur erstellen lässt. Doch vermutlich sind Sie eher hier, weil Sie eine Website erstellen wollen. Natürlich sollten Sie sich im Klaren sein, dass eine Website eine Daueraufgabe ist. Denn um eine gute und professionelle Website aufzubauen, sollten Sie diese stets erweitern und verbessern. Dies gilt sowohl für die Inhalte, als auch für die Programmierung.

## Erstellung eines Konzeptes

Wir werden innerhalb dieses Kapitels ein vollständiges Konzept für Ihre eigene Website zusammenstellen bzw. erarbeiten. Doch was ist eigentlich ein Website-Konzept? Ein Website-Konzept enthält die Zusammenstellung der Ziele, der Strategien, der Anforderungen, des Marketings, des Designs und der Inhalte. Doch warum dieser ganze Aufwand? Der Grund dafür ist ganz einfach: Um so ein besseres und ausgearbeitetes Konzept hinter einer Website steckt, umso besser wird auch die Umsetzung gelingen. Der Vorteil ist einfach zu erkennen: mehr Besucher, höhere Bekanntheit, evtl. mehr Kunden und Umsatz und vieles mehr.

Bildquelle: [Vektor-Grafik von Freepik](#)



**Übrigens:** Wie Ihnen vielleicht schon aufgefallen ist, stellen wir hier auf der Website verschiedene Software und Provider erst nach dem Kapitel „Allgemein“ vor. Dies mag für Sie evtl. etwas verwirrend sein, da Sie ja unter anderem auch Software zum Erstellen eines Konzeptes benötigen. Der Grund dafür ist schnell erklärt: Das Kapitel „Allgemein“ ist dazu gedacht, dass Sie sich das Kapitel vollständig durchlesen und sich nebenher ein paar Notizen machen. Die komplette Ausarbeitung des Konzeptes erfolgt also erst sobald Sie sich die Einführungs-Kapitel ([Allgemein](#), [Software](#) und [Provider](#)) durchgelesen haben.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: Homepage-Webhilfe » Allgemein » Geschichte

## Geschichte

Die Geschichte des Internets, des World Wide Webs und der Protokolle HTTP und HTTPS muss man zwar nicht zwingend kennen, doch es ist immer von Vorteil, wenn man darüber ein paar Dinge weiß. Falls Sie die Geschichte schon kennen oder kein Interesse haben, dann überspringen Sie das Thema einfach und lesen Sie beim Thema [Webseite und Website](#) weiter.

**Inhalt dieser Seite:**

1. Internet
2. World Wide Web
3. HTTP und HTTPS

## Internet

Als Internet wird die Zusammenfassung von allen Rechnernetzwerken bezeichnet. Erst durch die Entwicklung des Internets sind Dienste wie E-Mail (SMTP, POP3, IMAP), Fernwartung, das World Wide Web und vieles mehr möglich geworden. Die Kommunikation in diesem Netzwerk erfolgt über Protokolle der sogenannten Internetprotokollfamilie (TCP/IP).

Das Internet entstand durch den Vorläufer Arpanet. Das Arpanet entstand durch das Projekt Advanced Research Project Agency (ARPA) des amerikanischen Verteidigungsministeriums. Der Grund für das Projekt war, dass die UdSSR den ersten Satellit ins All schoss und sich die USA die führende Rolle im Gebiet der Technik zurückholen wollte. Die erste bedeutende Applikation des Internets waren E-Mails, welche 1971 erfunden wurden. Ebenfalls 1971 wurden die Protokolle Telnet und FTP entwickelt. Im Jahre 1974 wurde dann erstmals der Name „Internet“ verwendet. Dieser tauchte in einer früheren Version der TCP-Spezifikation auf, bei welchem es sich um ein Transportprotokoll für gesteuerte Kommunikationen, wie es auch bei HTTP der Fall ist, handelt.



## World Wide Web

In der Zwischenzeit wurden einige weitere Protokolle wie z. B. IPv4, ICMP und DNS entwickelt. Ab 1989 folgten dann die ersten Grundlagen des World Wide Webs (kurz WWW). Diese wurden von Tim Berners-Lee spezifiziert, welcher zu diesem Zeitpunkt am CERN (Europäische Organisation für Kernforschung) arbeitete. Am 6. August 1991 machte Berners-Lee das Projekt des Hypertextes-Dienstes öffentlich. Ende 1992 wurde dann die erste Spezifikation der Auszeichnungssprache HTML veröffentlicht, welche bis heute die Grundlage für Webseiten ist. Ebenfalls von Tim Berners-Lee gegründet, entstand 1994 das World Wide Web Consortium (kurz W3C). Diese beschäftigt sich mit der Spezifikation der Techniken im WWW.

Das WWW ist bis heute eines der wichtigsten Dienste des Internets, welche Websites wie Facebook, Google, YouTube, etc. erst möglich gemacht haben.

## HTTP und HTTPS

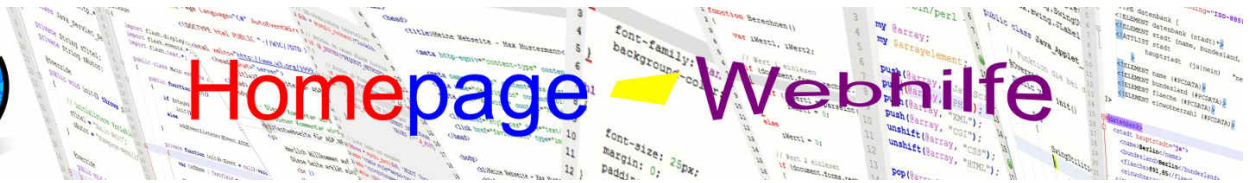


HTTP (HyperText Transfer Protocol) ist ein Anwendungsprotokoll, welches zur Übertragung von Hypertext verwendet wird. Es dient zur Übertragung von Webseiten und deren Bestandteilen (Stylesheets, Bilder, Icons, Videos, ...).

HTTP wurde 1991 eingeführt und 1996 unter HTTP/1.0 spezifiziert. 1999 folgte die Spezifikation für HTTP/1.1. Erst im Jahre 2015 folgte die Spezifikation für HTTP/2.0. Diese wird noch nicht auf allen Webservern unterstützt. "Normale" Web-Entwickler und -Designer müssen sich jedoch darüber zumeist keine Gedanken machen.

HTTPS ist eine Ergänzung zum Protokoll HTTP, wovon das S für Secure steht. Vom Protokoll entspricht HTTPS dem Protokoll HTTP. Bei HTTPS werden die Daten jedoch per SSL/TLS verschlüsselt. Die Bedeutung von HTTPS wird auf Grund der Internetkriminalität immer wichtiger.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisligen </p> <p><b>Web:</b> <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> <b>E-Mail:</b> <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Allgemein](#) » [Webseite und Website](#)

## Webseite und Website

Gerade für Anfänger ist es manchmal schwierig, die verschiedenen Begriffe im Bereich der Web-Technologien auseinander zu halten. Vor allem die Begriffe Webseite und Website werden oft durcheinander gebracht. Alles was Sie über diese zwei Begriffe wissen müssen, erfahren Sie hier.

### Inhalt dieser Seite:

1. Webseite
2. Webseite-Typen
3. Website
4. Website-Typen
5. Mobile Webseiten

### Webseite

Eine Webseite (auch Webdokument genannt) ist ein Dokument, welches einen Teil einer Website darstellt. Eine Webseite (oft auch einfach nur als Seite bezeichnet) kann mittels eines (Web-)Browsers betrachtet werden und ist meistens auf einem Webserver abgelegt. Zur Kommunikation dient das World Wide Web (HTTP(S)-Protokoll) und die URLs (Uniform Resource Locators) zur Angabe der Adresse. In der Umgangssprache wird auch oft von einer HTML-Seite oder einem HTML-Dokument gesprochen, welches dem Begriff Webseite gleichgesetzt werden kann (sofern HTML als Auszeichnungssprache verwendet wird).

### Webseite-Typen

Bei Webseiten wird zwischen verschiedenen Typen unterschieden. Eine **statische Seite** enthält einen Inhalt, der auf dem Server (in einer Datei) fest hinterlegt ist und an den Browser ohne Veränderung verschickt wird. Hier werden auch keine Änderungen am Dokument durch den Browser durchgeführt. Bei **dynamischen Seiten** wird der Inhalt dynamisch vom Webserver generiert, d. h. der Webserver erhält eine Anfrage und erstellt an Hand einer Vorlage und Programmierbefehlen die Webseite. Technisch realisiert wird dies durch verschiedene Skript- oder Programmiersprachen, die vom Webserver ausgeführt werden (serverseitige Sprachen). Hierzu zählen PHP, Perl, ASP.NET und Java EE. Der dritte Typ von Webseiten sind **aktive Webseiten**. Hier liegt auf dem Server eine „statische Seite“, die ohne Veränderung an den Browser geschickt wird. Der Webbrowser führt jedoch nun clientseitigen Skript- oder Programmiercode aus (programmiert wird in Sprachen wie z. B. JavaScript, Java oder Flash), wodurch der Inhalt der Webseite beeinflusst wird. In der heutigen Zeit wird vor allem die Kombination von dynamischen und aktiven Seiten verwendet, da die Interaktion zwischen Browser und Server immer wichtiger werden. Als Beispiel wäre z. B. ein Autovervollständigungs-Mechanismus zu nennen (Benutzer erhält während der Eingabe einige Vorschläge, was er evtl. eingeben möchte). Zur Kommunikation zwischen Browser und Server werden Technologien wie AJAX eingesetzt.

### Website

Als Website wird ein virtueller Platz im World Wide Web bezeichnet, deren Inhalte Webseiten und andere Ressourcen (wie z. B. Stylesheets und Skriptdateien, aber auch Bilder, Grafiken und Videos) sind. Andere Begriffe für Website sind Webauftritt, Webangebot und Webpräsenz. Eine Website kann auf einem oder mehreren Servern gespeichert sein.

Fälschlicherweise wird der Begriff Website auch oft mit dem Begriff **Homepage** gleichgesetzt. Die Homepage ist jedoch eine spezielle Webseite, bei welcher es sich um eine Seite handelt, welche die komplette Website repräsentiert (auch bekannt als Startseite). Üblicherweise befindet sich die Homepage im Root-Verzeichnis des Webserverns und besitzt den Dateinamen index.html, index.htm, index.php oder default.aspx.

### Website-Typen

Auf Grund der Inhalte, welche auf einer Website zur Verfügung gestellt werden, kann man Websites in verschiedene Kategorien bzw. Typen unterteilen. Die Typen sind nicht festgelegt und werden daher unterschiedlich interpretiert und eingeordnet. Zu erkennen sind die Typen meistens relativ einfach an Hand ihrer Merkmale.

Der einfachste Typ sind die sogenannten **Informations-Websites**. Hier werden Informationen über bestimmte Themen veröffentlicht. Ein weiterer Typ sind **Community-Websites**. Dieser Typ kann nochmals in weitere Typen unterteilt werden: Blogs, Foren, Wikis (z. B. Wikipedia), soziale Netzwerke (z. B. Facebook) und Onlineshops (z. B. Amazon). Gerade von Websites der Typen Blog, Forum und Wiki gibt es sehr viele im World Wide Web, da diese bei den meisten Internet-Surfern sehr beliebt sind. Ein weiterer Typ von Website, der wohl von fast jedem Internet-Surfer verwendet wird, ist die **Suchmaschine**. Die bekannteste Suchmaschine ist Google. Der letzte Typ von Websites sind **Web-Applikationen**. Hierzu zählen Office-Anwendungen und Spiele, welche im Web-Browser ausgeführt werden.

### Mobile Webseiten

Mobile Webseiten werden in der heutigen Zeit immer wichtiger. Unter einer mobilen Webseite versteht man eine Webseite, die speziell für die Anzeige auf Mobilgeräten (Smartphones und Tablets) optimiert ist. Bei einer solchen Optimierung gilt zu beachten, dass Mobilgeräte über eine geringere Auflösung verfügen und die Eingabe nicht per Maus, sondern per Touchscreen erfolgt. Die technische Umsetzung erfolgt meist nur über CSS, d. h. die Seite für Desktop-Geräte und Mobil-Geräte sind ein und dieselbe Webseite. Trotzdem spricht man im Allgemeinen von mobilen Webseiten.



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

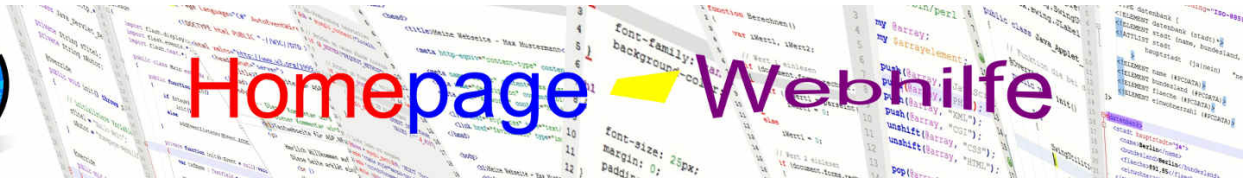
#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Allgemein](#) » [Ziele und Zielgruppe](#)

## Ziele und Zielgruppe

Sobald die Idee für Ihre Website steht, sollten Sie sich Gedanken über die Ziele und die Zielgruppe Ihrer Website machen. Damit haben Sie dann schon einen weiteren Teil zur Erstellung Ihres Konzeptes durchgeführt.

### Inhalt dieser Seite:

1. Ziele
2. Zielgruppe

### Ziele

Als erstes sollten wir festlegen, was wir mit unserer Webseite erreichen wollen. Für Firmen mag diese wohl vor allem Umsatzsteigerung und Erweiterung des Kundenstamms sein. Ziele von Betreibern von Privatseiten mag wohl eher die Anzahl der Besucher sein. Ein Ziel, wo sich fast alle setzen, ist die Erhöhung der Reichweite und Wahrnehmung. Natürlich gibt es noch viele weitere Ziele, doch die Frage ist, wie lege ich diese fest: Grundsätzlich müssen Sie überlegen, was Sie mit Ihrer Webseite erreichen wollen. Warum erstellen Sie überhaupt eine Webseite? Natürlich können auch Zeit, Ressourcen und Finanzen die Möglichkeiten der Ziele einschränken, die dann eben dementsprechend eingeplant werden müssen. Außerdem müssen Sie sich bei der Zielsetzung Gedanken machen, wie Sie die entsprechenden Ziele erreichen können. Auch die Analyse der Konkurrenz ist ein wichtiger Bestandteil zur Zielsetzung.

**Übrigens:** Sie werden im Bereich der Website-Konzeption immer wieder merken, dass diese Themen viel mit Marketing zu tun haben. Es ist also durchaus ratsam, sich mit anderen Personen zusammensetzen, die sich mit dem Thema Marketing auskennen.



### Zielgruppe



Nachdem wir die Ziele (auch als strategische Ziele bezeichnet) festgelegt haben, müssen wir unsere Zielgruppe fixieren. Als Zielgruppe wird eine Gruppe von Personen bezeichnet, die ein ähnliches Interesse haben oder bei denen bestimmte Faktoren übereinstimmen. Um eine Zielgruppe festzulegen, müssen wir uns verschiedene Zielgruppen-Kriterien herausarbeiten und dafür einen Bereich festlegen. Kriterien für Zielgruppen sind z. B. das Alter, das Geschlecht, der Familienstand, der Wohnort, der Beruf, das Gehalt und die eigene Meinung.

Doch warum erstellt man überhaupt Zielgruppen? Kann meine Website nicht einfach für alle sein? Generell ja, wenn Sie jedoch die Zielgruppe Ihrer Website kennen bzw. festgelegt haben, können Sie auch Ihre Inhalte dementsprechend aufbereiten und anbieten. Beim Erstellen einer komplett neuen Website kennt man seine Zielgruppe nicht unbedingt. Firmen können hier z. B. die Zielgruppe über die aktuellen Kundeninformationen erstellen. Bei Privatseiten geht dies meistens eher nicht. Hier gibt es eigentlich nur die Möglichkeit, die Website zu veröffentlichen und während einer „Testphase“ Ihre Besucher zu analysieren. Nach der Analysierung der Besucher und somit auch der Zielgruppe kann dann die Website überarbeitet, angepasst und perfektioniert werden. Die beliebtesten Tools zur Zielgruppen-Analyse auf einer Website sind Google Analytics und Piwik.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

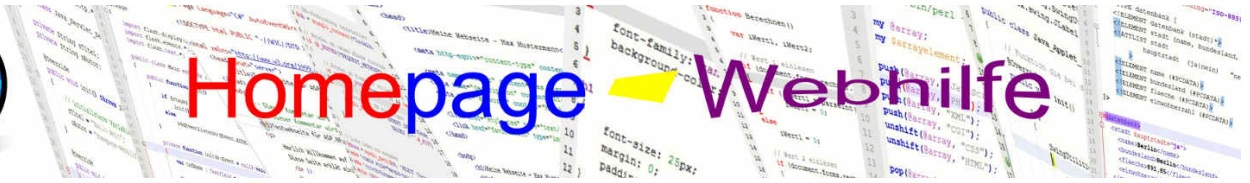
#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Allgemein](#) » [Struktur](#)

## Struktur

Wichtig für einen Besucher einer Webseite ist, dass er sich zurechtfindet. Findet ein Besucher auf Ihre Webseite und findet er sich nicht zurecht, so müssen Sie wahrscheinlich damit rechnen, dass dies der erste und letzte Besuch dieser Person war. Doch dies wollen wir vermeiden. Auf dieser Seite lernen Sie, wie Sie Ihre Seite aufbauen sollten, sodass sich Besucher leicht zurechtfinden.

### Inhalt dieser Seite:

1. Ansicht und Navigation
2. Ordner

## Ansicht und Navigation



Um den Besucher durch Ihre Website zu führen, benötigen Sie Navigationsleisten. Hier gibt es grundsätzlich 4 verschiedene Typen.

Der erste Typ ist die sogenannte Haupt-Navigation. Diese befindet sich normalerweise relativ weit oben auf der Seite (meistens unterhalb des Logos bzw. Banners). Die **Haupt-Navigation** soll es ermöglichen auf bestimmte Bereiche (z. B. Forum, Blog und Kontaktformular) der Website relativ schnell von überall zugreifen zu können. Gerne werden auch Menüs verwendet, welche ausgeklappt werden, sobald man mit der Maus darüber fährt. Eine weitere wichtige Navigationsleiste ist die **Subnavigation** (auch bekannt als lokale Navigation). Diese befindet sich oft auf der linken Seite und dient zur Navigation innerhalb eines Bereichs. Der dritte Typ von Navigationsleisten ist die sogenannte **Breadcrumb-Navigationsleiste** (Brotkrümel-Navigationsleiste). Diese ist oft zwischen Hauptnavigation und Inhaltsbereich zu finden. Mit Hilfe der Brotkrümel-Navigationsleiste soll der Besucher erkennen, in welcher Hierarchie-Ebene er sich befindet bzw. was der Weg (Pfad) zur aktuellen Seite ist. Die Breadcrumb-Navigation wird lediglich als einzelne Zeile angezeigt. Die einzelnen Elemente des Pfads werden mit Pfeilen getrennt. Oft befindet sich auch ein Text wie „*Sie befinden sich hier:*“ vor dem ersten Element der Navigationsleiste. Das erste Element der Brötkrümel-Navigation ist üblicherweise die Startseite (Homepage). Der 4. und letzte

Typ ist gerade in der letzten Zeit immer beliebter geworden: die **Footer-Navigation**. Die Footer-Navigation befindet sich wie der Name schon sagt in der Fußzeile einer Webseite. Heutzutage ist die Fußzeile jedoch meistens nicht nur eine einzelne Zeile, es ist vielmehr ein größerer Block mit diversen Angaben: von Adresse über Copyright und Urheberrecht bis hin zu diversen Links. Die Anordnung der Links und somit der Navigation ist völlig unterschiedlich: nebeneinander, untereinander oder beides zusammen.

Nicht immer sind alle 4 Typen von Navigationsleisten notwendig. Sie sollten sich genau überlegen, wie Sie Ihre Website von der Struktur aufbauen möchten. Nach dieser Überlegung wissen Sie dann auch relativ schnell, welche Navigationsleisten sinnvoll oder sogar notwendig sind.

**Übrigens:** Auf unserer Website finden Sie alle 4 Typen von Navigationsleisten. Dies liegt aber auch daran, dass unsere Website sehr groß und komplex ist.

## Ordner

Für eine bessere Übersichtlichkeit ist es hilfreich, nicht alle Dateien in das Rootverzeichnis der Website zu legen, sondern Unterordner zu erstellen. Dies gilt sowohl für die verschiedenen Webseiten als auch für andere Dateien wie Stylesheets, Skripte und Bilder. Für eine gute Übersicht hinsichtlich der einzelnen Webseiten ist es zu empfehlen, für jeden Bereich bzw. für jede Hierarchie-Ebene, einen Unterordner zu erstellen. Haben Sie also z. B. auf Ihrer Website einen Bereich namens „Software“, dann erstellen Sie einfach den Ordner „Software“ im Rootverzeichnis Ihrer Website. Sollte dieser Bereich noch in weitere Bereiche unterteilt sein, so sollten Sie Ordner mit dem Namen der Bereiche im Ordner „Software“ erstellen. Hierdurch haben Sie auf Ihrer Website eine klare Struktur und Navigation. Im Endeffekt profitieren 3 Gruppen von Personen davon: Sie als Website-Betreiber, da Sie Ihre Dokumente besser im Überblick haben, Ihre Besucher, da die URL und die Hierarchie auf der Website (welche z. B. durch die Breadcrumb-Navigationsleiste zu sehen ist) übereinstimmt und Suchmaschinen wie Google und Co.. Sie sehen also, dass von einer klaren Struktur Ihrer Website sowohl für die Anzeige als auch für die dahintersteckende Ordner nicht nur Sie, sondern auch andere Personen profitieren.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Allgemein](#) » [Erscheinungsbild](#)

## Erscheinungsbild

Das Erscheinungsbild ist eines der wichtigsten Bestandteile zur Planung einer Website. Wir werden hier einige Grundlagen für das Erscheinungsbild festlegen und im nächsten Thema mit der konkreten Planung und Konzeption des Webdesigns weitermachen. Doch nun wollen wir erst einmal ein paar Grundlagen setzen.

Als Erscheinungsbild bezeichnet man die Erscheinung, also wie etwas angezeigt werden soll, einer Seite auf einer Website. Dabei ist vor allem darauf zu achten, dass das Erscheinungsbild auf allen Webseiten einer Website gleich ist, da die Website andernfalls unruhig und nicht professionell aussieht. Die Kunst ist es, beim Programmieren eines Webdesigns, eine Grundlage zu schaffen, die später einfach und ohne viel Aufwand angepasst werden kann.

### Inhalt dieser Seite:

1. Schrift
2. Farben
3. Icons und Grafiken

## Schrift



Die Wahl der Schriftart ist nicht immer einfach, aber man sollte sich genauestens überlegen, welche Schriftart man verwenden möchte. Die Schriftart sollte einheitlich sein und auf jeder Seite verwendet werden. Auch Überschriften und Texte sollten grundsätzlich in der gleichen Schriftart sein. Für Schriftzüge, Banner oder ähnliches ist es jedoch sogar zu empfehlen, eine andere Schriftart zu verwenden. Falls Sie bereits über Dokumente oder Briefpapiere verfügen, sollten Sie darauf achten, dass auch hier die gleiche Schriftart verwendet wird.

Ein wichtiger Punkt bei der Wahl der Schriftart ist die Kompatibilität zu verschiedenen Betriebssystemen, denn wir wollen ja, dass die Schriftarten auf allen Computern gleich dargestellt werden. Hierfür gibt es einige sogenannte web-feste Schriftarten. Hierzu zählen Arial, Verdana, Helvetica, Sans-Serif, Times und Times New Roman. Bei Times und Times New Roman handelt es sich um sogenannte Serifen-Schriften. Diese Schriftarten verfügen über feine Linien am Ende eines Buchstabenstrichs quer zur Grundrichtung. Umgangssprachlich spricht man von Schnörkeln. Bei den anderen Schriften handelt es sich um sogenannte serifenlose Schriften, welche meistens beliebter sind, da Sie einfacher zu lesen sind.

Für informelle Websites ist daher auf jeden Fall eine der oben genannten Schriften zu empfehlen. In anderen Fällen sind evtl. andere „ausgefallene“ Schriften von Nöten. Auch für Schriftzüge und Banner sind durchaus ausgefallene Schriften notwendig. Doch wie verwendet man diese, wenn Sie nicht web-fest sind? Eine Möglichkeit wäre der Import von Schriftarten mittels CSS. Zu beachten gibt es jedoch, dass dies bei manchen Browsern nicht immer funktioniert. Ebenfalls kann es vorkommen, dass unter einem bestimmten Betriebssystem das Format für die Schriftart nicht richtig gelesen werden kann. Eine weitere Möglichkeit, die auf dem gleichen technischen Prinzip basiert, jedoch das Kompatibilitätsproblem umgeht, bietet Google mit [Google Fonts](#) an. Eine weitere Möglichkeit, welche jedoch nicht immer passend ist, ist die Verwendung von Bildern, d. h. Sie erstellen z. B. einen Banner oder Schriftzug mit Hilfe eines Grafikprogramms (in welchem die gewünschte Schrift verfügbar ist) und exportieren das Ganze im Anschluss als Grafik.

## Farben

Die Auswahl der Farben spielt für eine Website ebenfalls eine große Rolle. Zu viele oder zu unterschiedliche Farbe führen zur Verwirrung. Nur zwei Farben hingegen wirken schlichtweg langweilig und einfallslos. Ein erster Schritt bei der Farbauswahl ist die Auswahl von ein bis maximal drei verschiedenen Grundfarben, welche als Basis dienen sollen. Hierbei sollte man sich aber auch über die jeweilige Bedeutung der Farben (Farbsymbolik) Gedanken machen. So steht z. B. grün für Natur, Hoffnung und Ökologie, blau für Himmel und Wasser, rot für Liebe und Gefahr und gelb für Sonne und Adel. Vielleicht haben Sie sich schon mal gefragt, warum ein Grauton gerne als Hintergrundfarbe verwendet wird. Dafür gibt es mehrere mögliche Gründe: Zum einen wäre dies, dass grau eine leichte Farbe ist und vor allem zu schwarz und weiß ganz gut passt, zum anderen könnte diese Auswahl aber auch von der Bedeutung dieser Farbe kommen, denn grau steht u. A. für Weisheit und Sachlichkeit.

Nachdem wir unsere Grundfarbe(n) ausgesucht haben, müssen wir überlegen wie wir nun die anderen Farben wählen. Dazu gibt es nun verschiedene Farbregeln: ähnliche Farben (ähnliche aneinander angrenzende Farbtöne), monochrome Farben (gleicher Farbton mit unterschiedlichen Sättigungs- und Helligkeits-Werten), Triade-Farben (drei unterschiedliche Farbtöne im gleichen Abstand mit zusätzlicher Verwendung von unterschiedlichen Sättigungs- und Helligkeits-Werten), komplementäre Farben (zwei „gegenüberliegende“ Farbtöne mit zusätzlicher Verwendung von unterschiedlichen Sättigungs- und Helligkeits-Werten) und Schattierungs-Farben (gleicher Farbton und Sättigung mit unterschiedlicher Helligkeit). Bei der Auswahl von Farben werden Sie immer wieder auf verschiedene Farbmodelle stoßen: RGB (Red, Green, Blue), HSV (Hue, Saturation, Value) und HSL (Hue, Saturation, Lightness). Die „Helligkeit“ wird beim HSV-Farbraum, welcher für die Auswahl von Farben am praktischsten ist, üblicherweise als Dunkelstufe bezeichnet. Ein Tool, welches uns beim Erstellen eines Farbschemas hilft, stellt Adobe mit [Adobe Color CC](#) kostenlos als Web-Applikation zur Verfügung. Hier werden die Farbwerte mittels RGB-Modell und HSV-Modell (Farbkreis) ausgewählt. In Websites bzw. in deren Stylesheets erfolgt die Angabe einer Farbe meistens mit dem RGB-Modell. Die gewählte Schreibweise ist hier oft die hexadezimale Angabe (#RRGGBB). Mehr zu den unterschiedlichen Farbmodellen und der Umrechnung zwischen den Modellen finden Sie im [Kapitel Weiterführendes](#) auf unserer Website.

Auch bei den Farben kann es natürlich sein, dass Sie sich die Farben nicht aussuchen können, da diese bereits durch andere Dokumente oder Richtlinien vorgeschrieben sind. Diese sollten Sie dann natürlich übernehmen, denn die Farben müssen natürlich durchaus zu anderen Dokumenten und / oder zum Logo passen.



## Icons und Grafiken

Die Verwendung von Icons, Bildern und Grafiken ist für eine schöne und moderne Website unumgänglich. Denn seien wir mal ehrlich, wer liest sich schon gerne einen langen Text durch, wenn kein einziges Bild darin enthalten ist?

Gerade in der heutigen Zeit werden Sie auf immer mehr Websites treffen, bei denen viele Icons zu finden sind. Aber warum das Ganze? Es ist einfacher: Wenn wir z. B. auf einer Firmenwebsite sind und mit dieser Kontakt aufnehmen wollen, dann klicken wir einfach auf das Icon mit dem Telefon oder dem Brief. Das Icon sehen wir gleich, da es uns ins Auge fällt. Wir müssen also nicht lange nach einem Link mit der Beschriftung „Kontakt“ suchen. Ein Landkarten-Icon z. B. ersetzt den Link mit der Aufschrift „So erreichen Sie uns“ oder „Anfahrt“. Das Dateiformat für solche Icons sind zumeist PNG oder auch SVG (Vektor-Grafiken). Bei den sogenannten Favicons (das ist das Icon, welches in der Titelleiste des Browsers neben dem Webseitentitel angezeigt wird), wird auch oft auf den Microsoft Dateityp ICO zurückgegriffen.

Neben Icons gibt es noch Bilder und Grafiken. Gängige Dateiformate sind hier PNG, JPG oder GIF. GIF wird jedoch heutzutage fast nur noch für animierte Bilder und Grafiken verwendet. Vorteil von PNG im Vergleich zu JPG ist, dass PNG-Bilder über einen Alphakanal verfügen können und somit Teile des Bildes transparent sein können. Das Spektrum von möglichen Bildern und Grafiken ist groß: Zeichnungen, fotografiertes Bild, Logos, Banner, Schriftzüge, Diagramme und vieles mehr. Auch der Einsatz solcher Daten ist völlig unterschiedlich. Eine wichtige Grafik, die sich auf jeder Seite wiederfinden sollte, ist ein Logo, Banner oder Schriftzug.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

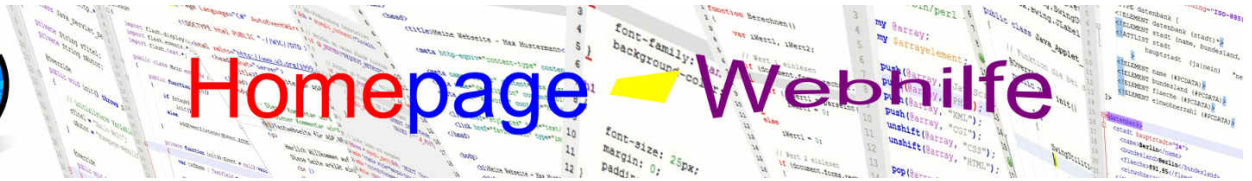
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





Sie befinden sich hier: [Homepage-Webhilfe](#) » [Allgemein](#) » [Design](#)

## Design



Das Erstellen des Designs für die Website ist sehr wichtig, jedoch sollten Sie sich dabei über eines immer Klaren sein: Ein Design ist selten für die Ewigkeit. Dies mag etwas traurig klingen, wenn man sich überlegt wie viel Zeit man sich für die Ausarbeitung eines Designs genommen hat, doch das Ganze ist wie in der Mode: Es gibt Trends, Fortschritte und gesellschaftliche Änderungen. Diese Veränderungen sollten nicht ignoriert werden, sondern genutzt werden. Wir müssen also zum Erhalt einer modernen und aktuellen Website, das Design immer wieder anpassen, überarbeiten und erneuern.

Beim Designen einer Website sollte man darauf achten, dass der Grund bzw. der Inhalt der Website klar zu erkennen ist, die Navigation einfach zu bedienen ist und Effekte zu integrieren, um die Website modern wirken zu lassen (Hover-Effekt). Immer im Hinterkopf sollte man auch seine Zielgruppe und die Ziele haben. Viele Grafiker verwenden sehr gerne hochwertige Grafiken und Animationen, doch hier ist Achtung geboten: Zu große Grafik-Dateien sorgen für langen Ladezeiten der Website. Dies sollte man auf jeden Fall vermeiden. Natürlich muss die Ladezeit auch bei anderen Dateien beachtet werden, doch gerade bei Grafiken entstehen sehr schnell große Dateien. Um dies zu vermeiden, sollten Sie Dateiformate mit Komprimierung nutzen und zudem das Bild nur mit der maximal benötigten Größe auf dem

Server ablegen (speichern Sie auf dem Server niemals eine Grafik mit z. B. 3000x300px, wenn Sie die Grafik nur in der Größe von 1000x100px benötigen).

Ein weiterer wichtiger Punkt beim Erstellen eines Designs ist, dass Websites in der heutigen Zeit responsiv sein müssen. Aber was heißt das überhaupt? Sie haben bestimmt schon mal etwas von **responsivem Webdesign** gehört. Es handelt sich dabei um eine Denkweise, ein Webdesign so zu gestalten, dass es auf mehreren Endgeräten möglichst gut aussieht. Technisch gesehen wird so etwas mit Hilfe von „Media Queries“ in **CSS** und mit JavaScript realisiert. Aber warum das Ganze? Heute sind die meisten Geräte internetfähig, was zur Folge hat, dass Ihre Website u. U. auf ganz unterschiedlichen Endgeräten und Endgeräte-Arten angezeigt wird. Diese besitzen jedoch meistens unterschiedliche Auflösungen und Eingabemedien. So verfügt ein Desktop-PC oder Laptop über einen Monitor mit einer Auflösung von ca. 1366x768px bis 1920x1080px, eine Tastatur und eine Maus. Ein Tablet-PC hingegen verfügt meistens über Auflösungen zwischen 800x600px bis 1280x800px und ein Touchscreen als Eingabemedium. Bei Handys ist ebenfalls ein Touchscreen zur Eingabe vorhanden. Die Auflösungen sind jedoch geringer und liegen oft nur zwischen 320x480px und 1280x720px. Sie müssen also vermutlich nicht nur 1 Design erstellen, sondern eher 2 bis 8 Designs. Eine mögliche Kombination von Designs wäre Desktop-Geräte, Tablet groß, Tablet klein, Handy groß und Hand klein. Die Kunst später beim Programmieren ist es, die Webseite vom HTML-Code so aufzubauen, dass dieser nur über CSS (und falls notwendig JavaScript) so angepasst wird, dass Sie die verschiedenen Designs für die verschiedenen Endgeräte haben.

Bildquelle: [Vektor-Grafik von Freepik](#)

## Papier-Entwurf

Der erste Entwurf für ein Webdesign erfolgt meistens auf einem Blatt Papier. Hierfür gibt es zwei verschiedene Möglichkeiten: Entweder Sie zeichnen auf einem Blatt alle Bestandteile direkt auf oder Sie erstellen einzelne Teile in Papierform und kleben diese nachher auf ein weiteres Blatt auf. Als Papier eignet sich am besten ein A4- oder A3-Blatt mit 70g/m<sup>2</sup> bis 100g/m<sup>2</sup>. Zeichnen Sie am besten alles mit Bleistift, sodass Sie Teile im Nachhinein nochmals wegradieren können. Wenn Sie möchten, können Sie gerne auch schon Farben (am besten Holzfarben) ins Spiel bringen. Schon bei diesem Entwurf sollten Sie sich grobe Gedanken über Dimensionierung und Abstände der Bestandteile machen. Beim Entwerfen Ihrer Website sollte Sie am besten mit der Desktop-Anzeige anfangen, denn dies ist die Ansicht, wo Sie am meisten sehen. Sobald dieser Entwurf fertig ist, können Sie mit dem Erstellen der weiteren Entwürfe fortfahren.

**Übrigens:** Sie werden es vermutlich nicht schaffen, das Design beim 1. Mal zu treffen. Seien Sie also nicht deprimiert und enttäuscht, wenn die ersten Entwürfe nichts geworden sind. Bleiben Sie ruhig und nehmen Sie sich die Zeit, die Sie brauchen.

## Entwurf am PC gestalten

Das Erstellen eines weiteren (etwas detaillierteren) Design-Entwurfs am PC erfolgt mit Hilfe der groben Vorlage, welche Sie bereits auf Papierform erstellt haben, und eines [Grafikprogramms](#), mit welchem Sie sich vorher schon etwas vertraut machen sollten. Hier sollten Sie sich schon mehr Gedanken über Positionierung und Dimensionierung machen. Des Weiteren ist es natürlich spätestens jetzt notwendig, Farben ins Spiel zu bringen. Ihr gewähltes Farbschema sollte natürlich zu diesem Zeitpunkt ebenfalls idealerweise feststehen.

## Entwurf anwenden

Nun ist es endlich soweit: Sie haben Ihre Entwürfe sowohl auf Papierform als auch am Computer erstellt. Der nächste Schritt ist nun die Website bzw. das Layout zu programmieren. Bevor wir uns jedoch dem Design widmen, müssen wir zuvor das „HTML-Layout“ erstellen. Gruppieren und strukturieren Sie Ihre Elemente und Bestandteile der Webseite und übersetzen Sie das Ganze in HTML-Code. Verwenden Sie dafür Positionierungs-Elemente wie *div* und *span* oder falls Sie HTML5 verwenden zusätzlich noch *header*, *footer*, *section*, *main* und *nav*. Wenn Sie sich nun die Seite im Browser anschauen, wird Ihnen vermutlich auffallen, dass fast alle Elemente untereinander angeordnet sind. Dies muss sich natürlich noch ändern. Und hier kommt nun CSS ins Spiel: Positionierung, Dimensionierung, Schrift und Farbe ist die Aufgabe von CSS. Wenn Sie auf Effekte setzen, ist es unumgänglich die Version 3 von CSS einzusetzen. Eventuell ist sogar noch JavaScript notwendig.

Spätestens jetzt ist es beim Erstellen des Designs mit Augenmaß vorbei. Ab jetzt werden Sie mit „rohen“ Zahlen arbeiten. Legen Sie sich also am besten einen Taschenrechner zurecht. Da dies nun der letzte Schritt des Designervorgangs ist, arbeiten Sie am Layout bis es Ihrer Meinung nach perfekt ist. Fragen Sie auch andere Personen was Sie von dem Layout halten, bevor Sie die Website veröffentlichen.

Bildquelle: [Vektor-Grafik von Freepik](#)



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Allgemein](#) » [Inhalte](#)

## Inhalte

Der Inhalt einer Website sind längst nicht mehr nur Texte: auch Bilder und Videos sind wichtiger Bestandteil. Und seien wir doch mal ehrlich: Was wäre eine Website ohne soziale Netzwerke und Werbung? Nachdem wir nun unsere Ziele und Zielgruppen definiert, eine Struktur aufgebaut, Schrift und Farbe festgelegt und ein Design ausgearbeitet haben, können wir damit anfangen unsere Inhalte zu erstellen.

**Inhalt dieser Seite:**

1. Texte
2. Bilder und Grafiken
3. Videos
4. soziale Netzwerke
5. Werbung

## Texte

Die Texte auf Webseiten sind zentraler Bestandteil. Überlegen Sie sich wie Sie die Personen anreden, welche Sprache, Akzent und Slang Sie verwenden. Zu beachten gilt auch, dass der Inhalt von Websites von den Besuchern oft nicht vollständig gelesen, sondern überflogen wird. Wichtig ist auch: Lädt Ihre Webseite zu langsam, gibt es zu viel Werbung oder ist die Seite überladen, so ist Ihr Besucher schneller wieder weg, wie er zu Ihrer Webseite gelangt ist. Wie bei anderen Texten auch: verwenden Sie Absätze, keine zu langen Sätze, nutzen Sie Hervorhebungen wie Fett-Druck und kursiven Text, vermeiden Sie hingegen unterschiedliche Farben in einem Satz (außer es handelt sich z. B. um eine Syntax-Hervorhebung für Programmcode), beziehen Sie Ihre Besucher ein (stellen Sie Ihnen Fragen o. Ä.), achten Sie auf Rechtschreibung und Grammatik. Bevor Sie Texte veröffentlichen, lesen Sie sich den Text nochmals selbst durch und kontrollieren Sie die in diesem Absatz genannte Kriterien.



## Bilder und Grafiken

Bilder und Grafiken sind in der heutigen Zeit zu wichtigen Bestandteilen von Webseiten geworden. Wir haben dieses Thema schon kurz im Thema [Erscheinungsbild](#) aufgegriffen. Verwendet werden Bilder und Grafiken zur Veranschaulichung, denn wie man so schön sagt: Ein Bild sagt mehr als tausend Wörter. Doch Vorsicht: Die Verwendung von zu vielen oder zu großen Bildern machen eine Webseite unübersichtlich und überladen. Ein nützlicher Hinweis noch: Wenn Sie Bilder in Ihre Webseite einbauen (HTML-Tag `img`) geben Sie den Bildern einen Titel (HTML-Attribut `title`), welcher beim Positionieren der Maus innerhalb des Bildes angezeigt wird.

Einige tun sich beim Erstellen von Grafiken, Bildern, aber auch Icons schwer. Dies mag u. A. daran liegen, dass die Minderheit der Website-Betreiber Grafiker von Beruf sind oder dies zu Ihrem Hobby gemacht haben. In solchen Fällen helfen meist nur Agenturen (Fotografen oder Grafiker), die solche Dienste anbieten oder das Suchen nach Grafiken im Internet. Interessante Angebote finden Sie hier z. B. bei [Freepik](#), [Iconarchive](#) oder [Iconfinder](#). Hier finden Sie kostenpflichtige, backlinkpflichtige und kostenlose Icons und Grafiken.

## Videos

Videos sind in der heutigen Zeit auch ganz gerne gesehen, doch sie befinden sich eher auf einem kleineren Teil der Websites. Dies liegt zum einen am Aufwand und zum anderen am Thema, denn für manche Themen lassen sich einfach nicht besonders gut Videos erstellen.

Um Videos in eine Website zu integrieren, gibt es aus technischer Sicht 3 Möglichkeiten: direkte Einbindung per HTML5, Einbindung per Flash oder Einbindung über iFrames über YouTube. Die letzte Möglichkeit findet sich auf den meisten Websites wieder, denn man hat zum einen keinen großen Programmieraufwand (die Programmierung von Flash ist nicht notwendig) und man kann sich aber auch fast sicher sein, dass so gut wie alle Browser das Video auch anschauen können. Diese Tatsache ist nämlich vor allem bei der direkten Einbindung mittels HTML5 nicht ohne weiteres gegeben.

## soziale Netzwerke



Was wäre das Internet heutzutage ohne soziale Medien und soziale Netzwerke? Was ein soziales Netzwerk ist, muss man wohl kaum mehr erklären: eine Community im WWW, über welche Personen aus aller Welt kommunizieren können. Unter sozialen Medien (englisch Social Media) versteht man die Medien und Technologien, welche die Kommunikation von Nutzern, und somit auch der sozialen Netzwerke, ermöglichen. Die bekanntesten sozialen Netzwerke sind Facebook und Twitter.

Viele Firmen, aber auch private Website-Betreiber, gehen mit dem Trend mit und erstellen für Ihre Website ein Profil (oder auch „Seite“ genannt) auf Facebook oder [Google +1](#). Warum sollte ich das tun? Ganz einfach: Personen die mit Ihren Dienstleistungen und Angeboten zufrieden sind, können Ihre Meinung teilen und Ihre Seite weiterempfehlen. Des Weiteren können neue Besucher gleich sehen, wie viele andere Personen Sie schon weiterempfohlen haben und was für eine Meinung Sie über Ihre Seite haben. Ein weiterer Vorteil: Personen die Ihnen folgen, können Sie mit Beiträgen immer auf dem neusten Stand halten.

**Achtung:** Gerade im Anfangsstadium einer neuen Website läuft man schnell in Gefahr, soziale Medien zu integrieren, aber keine Personen zu finden, welche Bewertungen abgeben. Bleibt dies auf längere Zeit so, so wirkt die Integrierung der sozialen Netzwerke schnell lächerlich. Suchen Sie also aktiv nach Personen und bitten Sie diese um Bewertung. Auch Freunde können hier behilflich sein.

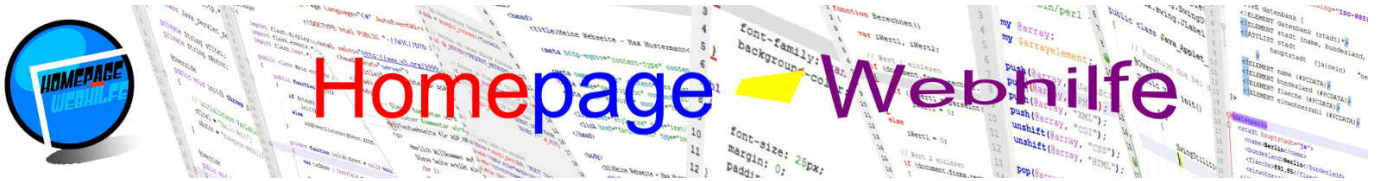
Bildquelle: [Vektor-Grafik von Freepik](#)

## Werbung

Werbung ist natürlich auch für Websites wichtig. Wie Sie auf sich im Internet aufmerksam machen wollen, kann ganz unterschiedlich erfolgen: Beispiele wären Werbung auf fremden Websites oder direkt bei Google schalten (z. B. [Google AdWords](#)) oder Backlinks auf anderen Webseiten zu platzieren (bitten Sie Besucher und Kunden auf der eigenen Website etwas Werbung für Sie zu platzieren). Gerade das Thema Backlinks ist sehr wichtig. Einige Backlinks können Sie aber auch selbst erstellen: hierzu zählen soziale Netzwerke, Signaturen in Foren und Blogs und Eintragungen in Webkataloge und Webverzeichnisse. Die Eintragung in Webkataloge und Webverzeichnisse hat leider nicht mehr den Effekt wie früher.

Das Gegenstück zu dieser Art von Werbung (also Werbung für die eigene Seite auf anderen Websites) wäre die Fremdwerbung (also Werbung auf der eigenen Website für fremde Seiten). Ein beliebter Anbieter ist hier Google mit [Google AdSense](#). Sie sollten sich jedoch gut überlegen, ob Sie Ihre Website mit Werbung (Text oder Bannern) veranstalten möchten und ob Ihnen die Einnahmen auch wirklich etwas bringen. Viele wollen Google AdSense oder andere Angebote nutzen, um die Kosten, die Ihnen durch das Hosting entstehen, zu decken. Leider klappt dies nicht immer wie gewünscht. Zusätzlich dazu ist hier Vorsicht geboten: In der Regel ist

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Allgemein](#) » [Inhalte](#)

die Anmeldung eines Gewerbes notwendig, da Sie mit dieser Art von Werbung gezielt Geld erwirtschaften wollen.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

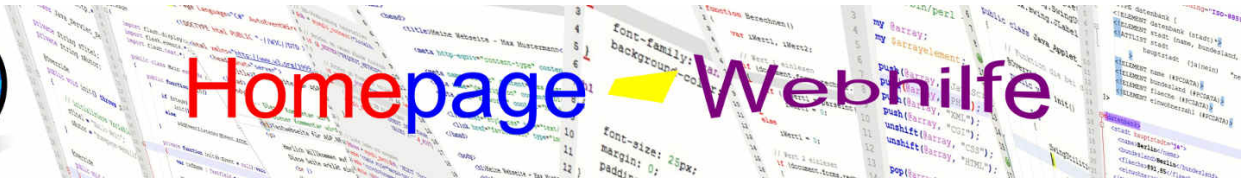
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Allgemein](#) » [Weiter geht's ...](#)

## Weiter geht's ...

Jetzt ist es soweit: Das erste Kapitel unserer Website ist zu Ende. Wir haben in diesem Kapitel nun die Grundlage der Web-Technologien, einige wichtige Fachbegriffe und die Vorgehensweise zum Erstellen eines Konzeptes gelernt. Nun ist es also an der Zeit, unser Website-Konzept zu erstellen und auszuarbeiten. Hierfür sind jedoch unter Umständen noch ein paar andere Kenntnisse oder Komponenten notwendig, welche wir Ihnen hier vorstellen möchten.

### Inhalt dieser Seite:

1. Software
2. Provider
3. Programmierung

## Software

Für die Erstellung eines Konzeptes und auch später für das Programmieren oder Designen der Website benötigen Sie Programme. Hier auf dieser Website wollen wir Ihnen ein paar Anwendungen vorstellen, die sowohl für Einsteiger als auch für Profis gut geeignet sind. [Zum Software-Bereich...](#)

## Provider

Spätestens vor dem Programmieren der Website sollten wir uns Gedanken über die technischen Rahmenbedingungen machen. Wenn Sie noch keinen Anbieter für Domains, Hosting, E-Mail oder Webseiten-Tools haben, sollten Sie sich einen Provider suchen, bevor Sie mit der Umsetzung und Realisierung Ihrer Website beginnen. [Zum Provider-Bereich...](#)

## Programmierung

Für die technische Umsetzung Ihrer Website werden verschiedene Programmier- Skript- und Auszeichnungssprachen eingesetzt. Einige kennen Sie vielleicht schon, eine andere müssen Sie eventuell noch lernen. Tutorials zu den verschiedenen Sprachen finden Sie hier auf unserer Website. [HTML](#) und [CSS](#) ist die Basis aller Websites. [JavaScript](#) und [ActionScript](#) wird als Client-Sprache für Browser-Anwendungen eingesetzt. [Perl](#), [PHP](#) und [ASP.NET](#) werden hingegen für Server-Anwendungen verwendet.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

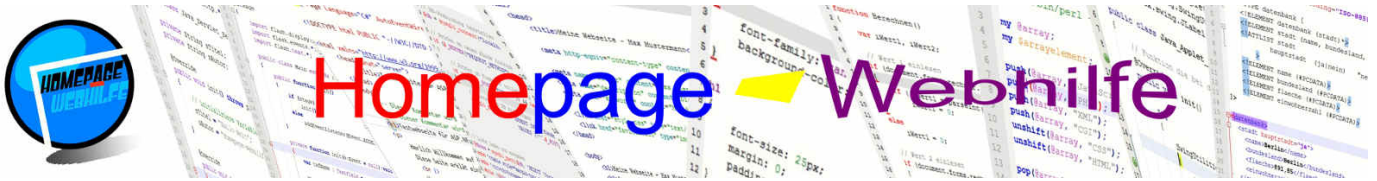
Java EE

XML

# E-Book

## Software





Sie befinden sich hier: [Homepage-Webhilfe](#) » [Software](#) » [Planung](#)

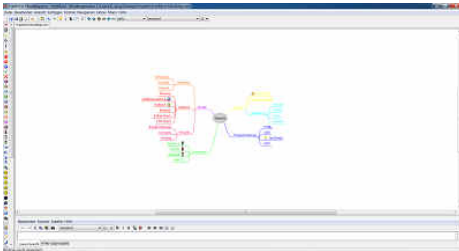
## Planung

Für die Planungs-Phase einer Website benötigen wir Programme diverser Art. Gerade jetzt sind aber auch noch Papier und Stift ein ganz wichtiges Werkzeug. Die wohl am meisten benötigten Programme sind Textverarbeitungsprogramme (Microsoft Word, LibreOffice Writer, OpenOffice Writer, WordPad) und Tabellenkalkulationsprogramme (Microsoft Excel, LibreOffice Calc, OpenOffice Calc). Eine weitere nützliche Art von Programmen sind Programme zur Visualisierung (Mindmaps, Flowcharts, Diagramme, etc.). Des Weiteren gibt es noch Programme, mit welchen Projekte geplant werden können. Diese können natürlich nicht nur für die Website-Erstellung eingesetzt werden.

### Inhalt dieser Seite:

1. FreeMind
2. Microsoft Visio
3. GanttProject
4. Microsoft Project

## FreeMind



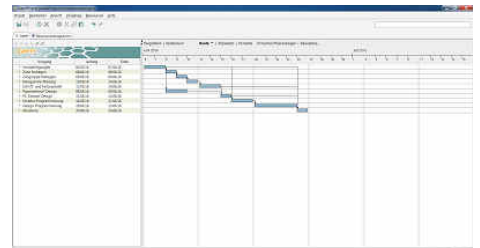
FreeMind ist ein kostenloses Programm zum Erstellen von Mindmaps. Das Programm ist also perfekt dafür geeignet, ein Brainstorming am Computer durchzuführen oder ein auf Papier erstelltes Brainstorming zu digitalisieren. Da das Programm in Java geschrieben wurde, benötigt es zwar die Java Laufzeitumgebung (JRE), ist dafür jedoch aber auch plattformunabhängig. Das Erstellen der Mindmaps ist mit diesem Programm sehr einfach. Einstellungen wie Farbe und Icons sind mit wenigen Klicks erledigt. Des Weiteren verfügt das Programm über eine Funktion zum Exportieren der Mindmap in HTML. Die Entwicklung des Programms erfolgt durch eine freie Community.

## Microsoft Visio

Microsoft Visio gehört zu der Office-Reihe von Microsoft und stellt eine professionelle Möglichkeit für das Visualisieren von Inhalten und Daten zur Verfügung. In Microsoft Visio können Flussdiagramme, Netzwerkpläne, Terminpläne (Gantt-Diagramm, PERT-Diagramm), Grundrisse, Konstruktionspläne (Elektronik, Pneumatik, Schaltkreise) und vieles mehr erstellt werden. Die Bedienung des Programms ist eine perfekte Kombination zwischen Einfachheit und Komplexität. Das Programm ist im Gegensatz zu FreeMind kostenpflichtig.

## GanttProject

GanttProject ist ein Programm zum Planen eines Projekts. Dabei besteht die Hauptaufgabe des Programms darin, ein Gantt-Diagramm darzustellen, aus welchem einfach die Start- und Endzeit sowie die Vorgänger der einzelnen Vorgänge abgelesen werden können. Das Programm verfügt lediglich über einfache Funktionen zur Projektplanung: Vorgangs-Planung und Ressourcen-Planung. Für größere oder professionellere Projekte ist dieses Programm, auf Grund der Einschränkungen in einigen Bereichen, nicht unbedingt zu gebrauchen. Ein Vorteil von GanttProject, wie bei FreeMind auch, ist, dass es plattformunabhängig ist.



## Microsoft Project

Microsoft Project ist das kostenpflichtige Projektplanungsprogramm aus dem Hause Microsoft. Im Gegensatz zu GanttProject verfügt Microsoft Project über deutlich mehr Einstellungen für die einzelnen Vorgänge und Ressourcen. Für eine professionelle Planung und Realisierung von Projekten darf natürlich auch eine Überwachungs- und Steuerungs-Funktion nicht fehlen. Diese Funktionen sowie die Erstellung von Berichten sind kein Problem für Microsoft Project.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

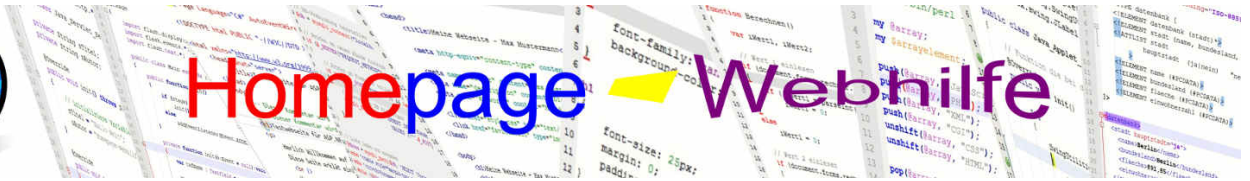
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Software](#) » [Grafik](#)

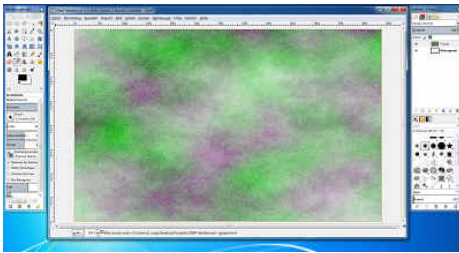
## Grafik

Programme zur Grafikbearbeitung und -erstellung sind meistens sehr teuer. Gerade Anfängern ist ein solcher Preis für die Verwendung eines Grafikprogramms nicht wert. Doch es gibt auch gute kostenlose Alternativen, mit welchen wir Grafiken erstellen oder vorhandene Bilder modifizieren können.

### Inhalt dieser Seite:

1. GIMP
2. Paint.NET
3. Microsoft Expression Design
4. Adobe Photoshop
5. Adobe Illustrator

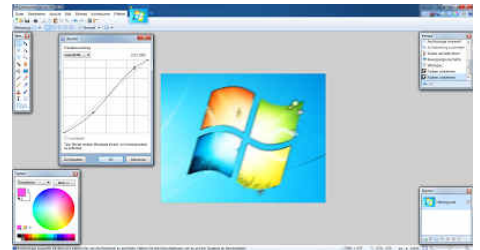
### GIMP



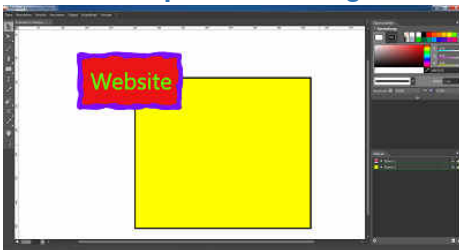
GIMP ist wohl der bekannteste und beliebteste Ersatz für Adobe Photoshop und Adobe Illustrator, wenn es um kostenlose Grafikprogramme geht. GIMP enthält alle Funktionen die Anfänger brauchen und mehr. Einige professionelle Grafiker mögen vielleicht behaupten, dass Ihnen ein paar Funktionen fehlen, doch diese sind nun für uns eher weniger interessant, zumindest wenn wir ein kostenloses Programm zum Bearbeiten und Erstellen von Grafiken suchen. GIMP unterstützt die Farbräume RGB, HSV und CMYK und kann sowohl mit Pixel- als auch mit Vektorgrafiken arbeiten. Markierungen nach Farbe, nach Bereichen (an Hand der Farbe und Kanten) oder nach Formen (rund, rechteckig und formfrei) stellen keine Hürde für GIMP dar. Des Weiteren ermöglicht GIMP die Erstellung und Verwendung von Skripten und Plug-ins zur Automatisierung von Vorgängen und Erweiterungen des Programms.

### Paint.NET

Paint.NET ist ein Bildbearbeitungsprogramm für Windows-Betriebssysteme, welches von Programmierern, welche bei Microsoft arbeiten, entwickelt wird. Die Oberfläche und Bedienung ist dabei an Adobe Photoshop angelehnt. Ursprünglich war Paint.NET eine kostenlose Alternative für das Programm Microsoft Paint, welches mit Windows bereits mitgeliefert wird. Im Laufe der Jahre wurde das Programm um immer mehr Funktionen und Effekte erweitert, sodass es für Standard-Bildbearbeitungen aber auch für professionellere Bearbeitungen ausreichend ist.



### Microsoft Expression Design



Microsoft Expression Design ist ein professionelles Grafikprogramm von Microsoft aus der Expression-Reihe. Das Erstellen von einfachen Grafiken mit Hilfe von Rechtecken, Kreisen, Linien und Texten fällt hier besonders einfach. Das Programm wird seit Ende 2012 nicht mehr weiterentwickelt und wurde daher von Microsoft kostenlos zum Download bereitgestellt.

### Adobe Photoshop

Adobe Photoshop ist wohl das Bildbearbeitungsprogramm schlicht hin. Das Programm aus dem Hause Adobe erschien erstmals 1990 und ist Weltmarktführer im Bereich der Bildbearbeitung. Photoshop ist für die Betriebssysteme Windows und Mac OS X verfügbar. Adobe Photoshop ist nur zur Bearbeitung für Pixelgrafiken gedacht.

### Adobe Illustrator

Adobe Illustrator ist die perfekte Ergänzung zu Adobe Photoshop für jeden Grafiker. Im Gegensatz zu Photoshop ist Illustrator zur Bearbeitung und Erstellung von Vektorgrafiken gedacht. Adobe Illustrator ist ebenfalls für Windows und Mac OS X erhältlich. Die erste Version erschien bereits im Jahr 1987.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

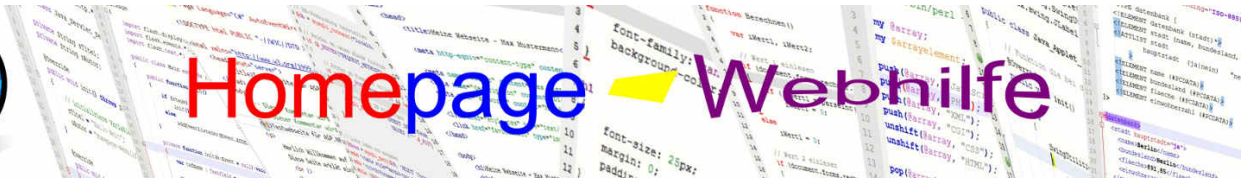
#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Software](#) » [Webserversystem](#)

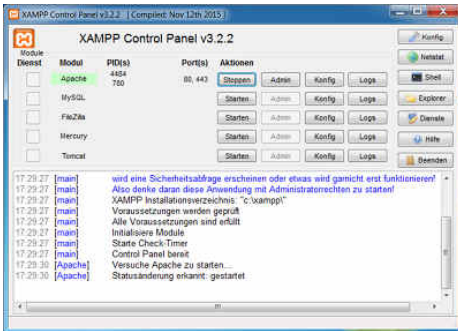
## Webserversystem

Um eine Website lokal (also auf einem Computer im Netzwerk bzw. dem eigenen Computer) zu betreiben, ist ein Webserver(-system) notwendig. Neben dem Webserver (Apache, IIS) ist oft auch noch eine Datenbankanbindung (MySQL, MariaDB, MSSQL) notwendig. Hier wollen wir Ihnen Software-Pakete vorstellen, mit welchen Programmierer und Designer während der Entwicklung der Website arbeiten.

### Inhalt dieser Seite:

1. XAMPP
2. Microsoft IIS

## XAMPP



XAMPP ist eine Zusammenstellung von verschiedenen Software-Paketen, wobei das X für die Unterstützung verschiedener Betriebssysteme steht. Die anderen Buchstaben sind die jeweiligen Anfangsbuchstaben der im Programm enthaltenen Pakete: Apache (Webserver), MariaDB (SQL-Server, früher MySQL), Perl (Server-Skriptsprachen-Interpreter), PHP (Server-Skriptsprachen-Interpreter). Weitere enthaltene Pakete sind Mercury Mail (Mailserver für POP, IMAP und SMTP), OpenSSL (zur HTTPS-Unterstützung), FileZilla-Server (FTP-Server), Apache Tomcat (Webserver für JSP und Servlet Anwendungen), WebDAV, Webalizer und phpMyAdmin (PHP-Tool zur Administration von MySQL-Datenbanken). Dieses Tool ist für Webentwickler so gut wie unumgänglich.

## Microsoft IIS

Microsoft IIS (Internet Information Services) ist eine Plattform, um Daten über die Protokolle HTTP, HTTPS, FTP und WebDAV in einem Netzwerk (auch über das Internet) bereitzustellen. Grundbestandteil des IIS ist der Webserver, auf welchem ASP.NET-Anwendungen betrieben werden können. Mit Hilfe von Erweiterungen (sogenannten ISAPI-Filtern) kann die Unterstützung von PHP- und sogar JSP-Anwendungen hinzugefügt werden. Der IIS kommt auf Windows-Server-Systemen zum Einsatz. Es existiert jedoch eine Express-Version, die unter anderem in [Visual Studio](#) enthalten ist, um ASP.NET-Anwendungen entwickeln und testen zu können. Diese Version ist jedoch vom Funktionsumfang eingeschränkt und dient lediglich zu Test- und Entwicklungs-Zwecken.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





Sie befinden sich hier: [Homepage-Webhilfe](#) » [Software](#) » [Editor](#)

## Editor

Editoren (und Entwicklungsumgebungen) sind eines der wichtigsten Programme für Entwickler. Bei Editoren unterscheidet man grundsätzlich zwischen Quellcode- bzw. Text-Editoren und WYSIWYG-Editoren (What You See Is What You Get). Entwicklungsumgebungen (kurz IDE für Integrated Development Environment) sind Programme, welche meistens sowohl über einen Quellcode-Editor, wie auch einen Compiler und u. U. sogar einen Debugger verfügen. Zum Schreiben von Quellcode könnte man zwar auch einen einfachen Texteditor (z. B. Notepad) nutzen, jedoch verfügen Quellcode-Editoren über erweiterte Funktionen zur Anzeige von Hilfen (Tooltips) und zur automatischen Vervollständigung.

**Inhalt dieser Seite:**

1. Adobe Brackets
2. Webocton Scriptly
3. Notepad++
4. FlashDevelop
5. NetBeans IDE
6. Visual Studio
7. Microsoft Expression Web
8. Adobe Dreamweaver

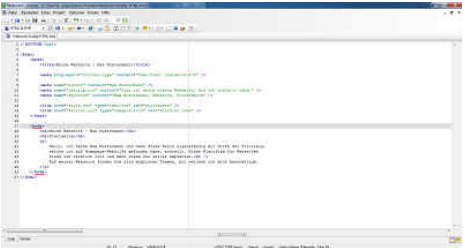
### Adobe Brackets



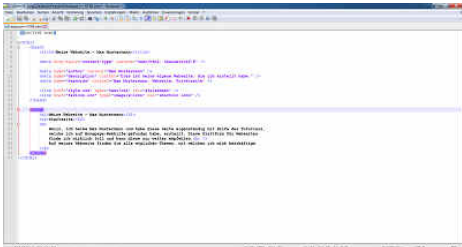
Adobe Brackets ist ein professioneller Editor zum Entwickeln von Web-Applikationen. Neben der Unterstützung für das Syntax-Highlighting von Sprachen wie HTML, CSS, JavaScript, PHP, Perl, Python, SQL, XML und vielen mehr, bietet der Editor Funktionen zur Projekt bzw. Website-Verwaltung und eine Live-Vorschau. Das Programm wird von einer Community entwickelt. Des Weiteren sind viele Plugins zur Erweiterung verfügbar. Hierzu zählen Dokumentationen für Funktionen, Autovervollständigungen und Linter (Tools zur automatischen Code-Analyse).

### Webocton Scriptly

Webocton Scriptly ist ein Editor für HTML, CSS, JavaScript, PHP und XML. Der Editor ist nur für Windows Betriebssysteme geeignet. Das Programm bietet Syntax-Hervorhebung und Autovervollständigung. Des Weiteren kann der Editor eine Verbindung mit einem MySQL-Server und FTP-Server herstellen. Der Editor wird aktuell nicht mehr weiterentwickelt und unterstützt nicht mehr alle aktuellen Tags, Attribute, Eigenschaften, Funktionen der verschiedenen Sprachen. Gerade für Anfänger ist dieser Editor bei Verwendung von modernen Technologien (HTML5, CSS3) mit Vorsicht zu genießen.



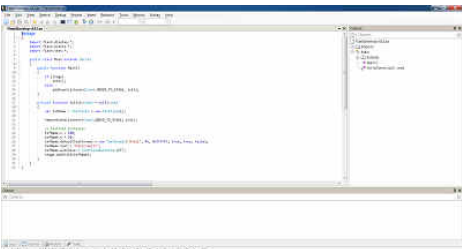
### Notepad++



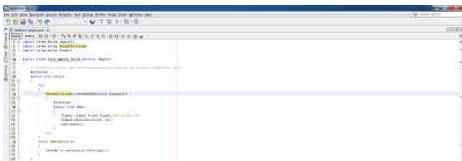
Notepad++ ist ein Text-Editor, welcher eine Vielzahl von Datei-Kodierungen (ASCII, UTF-8, ISO-8859-1, Windows-1252, ...) und Sprachen (HTML, CSS, JavaScript, PHP, aber auch C, C++, C# und viele mehr) unterstützt. Notepad++ ist einfach zu bedienen, enthält eine Autovervollständigung, ist jedoch ebenfalls für Anfänger mit Vorsicht zu verwenden, da Notepad++ bei der Autovervollständigung alles vorschlägt, was bereits eingegeben wurde (und somit auch falsche Eingaben). Notepad++ kann mit Plug-Ins um Funktionen wie z. B. den Dateivergleich erweitert werden.

### FlashDevelop

FlashDevelop ist ein IDE, um Flash-Anwendungen zu schreiben, entwickeln, kompilieren und debuggen. Um AS3-Anwendungen (Action Script 3) erstellen zu können, nutzt das Programm die Apache Flex SDK, wobei es sich um eine freie SDK (Software Development Kit) der Apache Software Foundation handelt. FlashDevelop kann auf Windows-Betriebssystemen ab Windows XP eingesetzt werden. Gerade Einsteiger nutzen die FlashDevelop IDE gerne, da diese kostenlos erhältlich ist. Natürlich ist der Funktionsumfang von FlashDevelop nicht mit Adobe Flash Professional zu vergleichen.

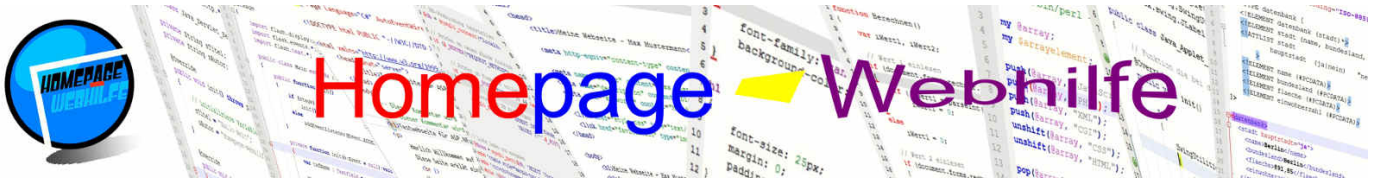


### NetBeans IDE



NetBeans IDE ist eine Entwicklungsumgebung von Oracle, welche hauptsächlich zur Entwicklung von Java-Anwendungen (Applets, Servlets, Server Pages, Server Faces) dient. Jedoch unterstützt die Entwicklungsumgebung in der Zwischenzeit auch Sprachen wie HTML, CSS, JavaScript und PHP sowie C und C++. Zum Entwickeln bietet das Programm Debug-Funktionen und einen GUI-Builder für Java-Anwendungen. NetBeans kann über Plug-ins erweitert werden. Die Anwendung ist einfach zu bedienen und ist zum Erstellen von Java-Anwendungen sehr zu empfehlen.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--

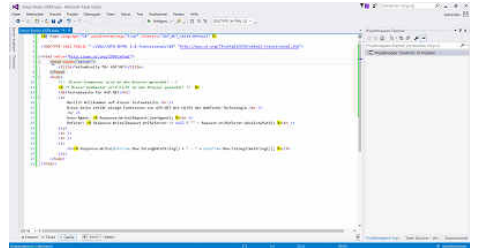


Sie befinden sich hier: [Homepage-Webhilfe](#) » [Software](#) » [Editor](#)



## Visual Studio

Visual Studio ist die professionelle und funktionsreiche Entwicklungsumgebung aus dem Hause Microsoft. Die IDE dient zum Entwickeln von Desktop-Anwendungen, mobilen Apps und Webseiten. Als Programmiersprachen kommen C#, Visual Basic und F# sowie C, C++ und Python zum Einsatz. Visual Studio kann zum Entwickeln von ASP.NET-Websites verwendet werden, wofür das Programm den IIS Express enthält. Seit der Version von 2013 gibt es die Community-Version, welche uneingeschränkt für Privat-Anwender, Schüler, Studenten und kleinere Unternehmen genutzt werden darf und alle notwendigen Funktionen für professionelles Entwickeln enthält. Die erste Version von Visual Studio wurde bereits 1997 veröffentlicht. Visual Studio ist die perfekte Entwicklungsumgebung für .NET-Anwendungen.



## Microsoft Expression Web



Microsoft Expression Web ist eine Entwicklungsumgebung für Webseiten. Es dient zum Editieren von HTML-, CSS-, JavaScript- und PHP-Quellcode. Für Nicht-Programmierer enthält das Programm auch einen WYSIWYG-Editor. Des Weiteren sind die Einbindung von externen Datenquellen und die Verwendung von ASP.NET möglich. Expression Web war bis 2012 nur gegen Bezahlung erhältlich. Die Weiterentwicklung wurde von seiten Microsofts eingestellt und die letzte Version kostenlos zum Download angeboten. Deshalb sollte man bei diesem Programm beachten, dass dies nicht mehr auf dem neusten Stand der aktuellen Technologien ist.

## Adobe Dreamweaver

Adobe Dreamweaver ist ein Editor für HTML, CSS, JavaScript und PHP. Dreamweaver kann unter Windows und Mac OS X eingesetzt werden. Das Programm von Adobe Systems ist kostenpflichtig und wird daher von Privat-Anwendern eher selten verwendet. Zusätzlich zur Quelltext-Verarbeitung bietet die Anwendung einen WYSIWYG-Editor.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: Homepage-Webhilfe » Software » Browser

## Browser

Was ein Browser (eigentlich Webbrowser) ist, muss man wohl kaum erklären. Wir benutzen ihn fast täglich, um Webseiten aufzurufen, Dinge im Internet zu suchen oder zu bestellen und für vieles mehr. Grundsätzlich sollten Sie bedenken, dass Web-Entwickler und -Designer in soviel wie möglichen Browsern (auch im eher unbeliebten Microsoft Internet Explorer) testen sollten. Gerade während der Entwicklungs-Phase sind aber vor allem die Browser Mozilla Firefox und Google Chrome von Vorteil, da Sie starke Tools zur Entwicklung (Konsole, Inspektor, Stil-Bearbeitung, Netzwerk-Analyse und Debugger) und Fehlersuche bieten. Zu beachten ist auch, dass diese beiden Browser die beliebtesten Browser sind und somit auch am meisten verwendet werden.

**Inhalt dieser Seite:**

1. Mozilla Firefox
2. Google Chrome

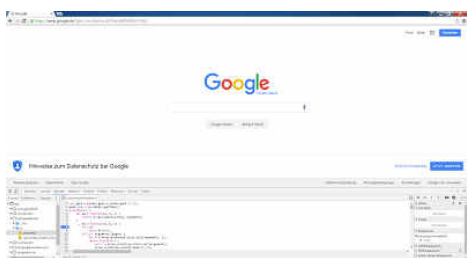
### Mozilla Firefox



Mozilla Firefox ist ein freier Browser der Mozilla Corporation. Die erste Version erschien im Jahre 2002. In Deutschland gilt Firefox als meistgenutzter Browser, weltweit belegt Firefox Platz 2. Firefox ist für Windows, Mac OS X, Linux, iOS und andere Betriebssysteme erhältlich. Über Add-ons und Plugins kann der Browser um weitere Funktionen und Designs erweitert werden. Gerade bei Entwicklern ist Firefox sehr beliebt, da er viele Tools bietet: Hierzu zählen die Konsole zum Ausführen von JavaScript-Befehlen und betrachten von Fehler- oder Warnmeldungen, der Inspektor zum Anschauen und Editieren des HTML-Codes, der Debugger zum Debuggen von JavaScript-Code, die Stilbearbeitung zum Betrachten und Editieren von CSS-Regeln und die Netzwerkanalyse zum Analysieren der über das Netzwerk übertragenen Dateien.

### Google Chrome

Google Chrome ist der Webbrowser von Google, welcher erstmals 2008 veröffentlicht wurde. Eingesetzt werden kann der Browser unter Windows, Mac OS X, Linux, iOS und Android. Google wirbt bei der Vermarktung vor allem durch die Schnelligkeit des Browser. Tatsächlich haben Benchmark-Tests ergeben, dass Chrome schneller als andere Browser sind. Chrome gilt in der Zwischenzeit weltweit als meist genutzter Browser. So wie Firefox, verfügt auch Chrome über einige Tools für Webdesigner. Der Aufbau dieser Funktionen ist ähnlich wie bei Mozilla Firefox.



<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p><b>Web:</b> <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> <b>E-Mail:</b> <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Software](#) » [E-Mail-Client](#)

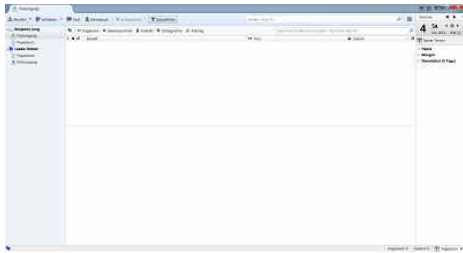
## E-Mail-Client

Ein E-Mail-Client oder einfacher gesagt ein E-Mail-Programm ist ein Programm zum Abrufen und Schreiben von E-Mails. Viele Privatanwender werden gar kein E-Mail-Programm besitzen, sondern lediglich über die „Anwendung“ des jeweiligen Anbieters auf der Website Ihres Providers E-Mails abrufen und schreiben. Doch ein E-Mail-Client hat einige Vorteile: z. B. einfachere Verwaltung und keine Werbung (viele Provider finanzieren sich durch Werbung auf Ihrer Website).

### Inhalt dieser Seite:

1. Mozilla Thunderbird
2. Microsoft Outlook

## Mozilla Thunderbird



Mozilla Thunderbird ist ein weiteres freies Programm, welches von der Mozilla Corporation entwickelt wird. Es ist eines der beliebtesten und meist genutzten Programme, um E-Mails zu lesen und zu schreiben. Erhältlich ist Thunderbird für Windows, Mac OS X und Linux. Die erste Version erschien 2003. 2015 wurde Lightning in Thunderbird integriert, welches das Programm um eine Termin- und Aufgaben-Planung erweitert.

## Microsoft Outlook

Microsoft Outlook ist ein Programm aus der Office-Reihe, welches Funktion zur E-Mail-, Termin-, Kontakt- und Aufgaben-Verwaltung enthält. Im Gegensatz zu Thunderbird ist Outlook kostenpflichtig und kann nur über das Microsoft Office-Paket erworben werden. Auch wenn Outlook einige mehr Funktionen enthält und als meist verwendetes E-Mail-Programm gilt, verwenden viele Mozilla Thunderbird, da es kostenlos erhältlich ist und ebenfalls einfach zu bedienen ist.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

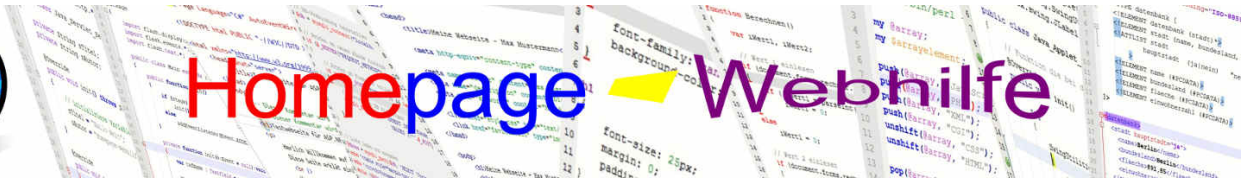
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Software](#) » [FTP-Client](#)

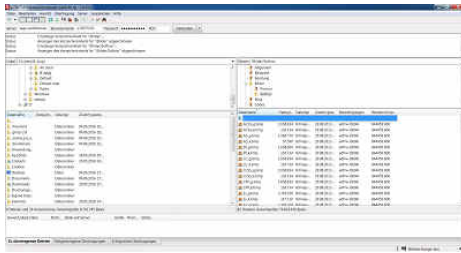
## FTP-Client

Ein FTP-Client ist ein Programm, um Dateien mit Hilfe des FTP-Protokolls zu transferieren. Es dient bei Webentwicklern dazu, Webseiten, Stylesheets, Skripte, Bilder und andere Dateien auf den Webserver und somit auf die Website zu übertragen. Bei FTP-Clients gibt es relativ wenig unterschiedliche Programme, welche auch „oft“ eingesetzt werden.

### Inhalt dieser Seite:

1. FileZilla

## FileZilla



FileZilla ist eine Anwendung von "Tim Koose und Team" und kann unter Windows, Mac OS X und Linux eingesetzt werden. Die erste Version erschien 2001. FileZilla unterstützt neben dem Protokoll FTP, auch das Protokoll SFTP, welches eine verschlüsselte Verbindung zur Datenübertragung nutzt. Der FileZilla Client bietet eine einfache Oberfläche zum Transferieren von Dateien von PC zu Server und von Server zu PC.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

Java EE

XML

# E-Book

## Provider





Sie befinden sich hier: Homepage-Webhilfe » Provider » E-Mail

## E-Mail

Eine E-Mail-Adresse dient zum elektronischen Versenden von „Briefen“. Fast jeder hat eine E-Mail-Adresse, doch die wenigsten machen sich Gedanken über den Anbieter. Wir wollen Ihnen hier ein paar bekannte E-Mail-Diensteanbieter vorstellen: darunter kostenlose und kostenpflichtige Tarife. Die kostenlosen Anbieter wurden von uns getestet und verfügen in den untenstehenden Texten einen Screenshot von der Weboberfläche zum Betrachten des Posteingangs. Falls Sie vorhaben, eine Domain zu bestellen, werden Sie vermutlich mit Ihrem Domain oder Hosting-Angebot auch E-Mail-Adressen erhalten. Am Seitenende finden Sie eine Tabelle mit den wichtigsten Daten zu den verschiedenen Tarifen.

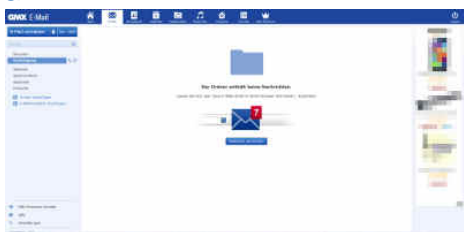
**Inhalt dieser Seite:**

1. 1&1
2. GMX
3. Google Mail
4. Strato
5. T-Online
6. Web.de
7. Yahoo
8. Vergleich

### 1&1

1&1 (rechtlich 1&1 Internet SE) ist ein bekannter Diensteanbieter für verschiedene Produkte im Bereich Internet und Telekommunikation. Hierzu zählen DSL und Mobilfunk sowie Hosting, Domains, Server und E-Mail. Die E-Mail-Angebote von 1&1 sind kostenpflichtig und sind direkt mit einer .de-Domain verbunden. Es existieren zwei verschiedene Pakete, welche zudem noch über die Anzahl der Postfächer angepasst werden können.

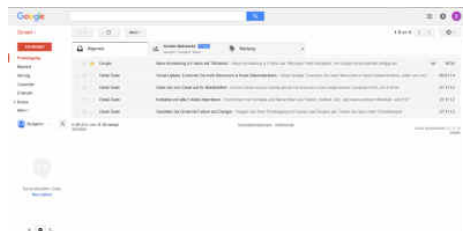
### GMX



GMX ist ein Webportal, welches von der 1&1 Internet SE betrieben wird. Auf dem Webportal sind aktuelle Nachrichten zu finden. Hauptsächlich wird GMX (Global Message eXchange) jedoch für den E-Mail-Dienst genutzt. Hier stehen uns 3 verschiedene Produkte zur Auswahl, wovon das erste kostenlos erhältlich ist und 1 GB Speicherplatz sowie 2 E-Mail-Adressen (@gmx.de, @gmx.net, @gmx.eu und weitere) umfasst.

### Google Mail

Google Mail (kurz Gmail) ist der E-Mail-Dienst von Google. Google stellt diesen Dienst kostenlos zur Verfügung und gilt seit 2012 mit ca. 1 Milliarde Accounts als meist genutzter E-Mail-Dienst. Zur Erstellung eines Google Mail Accounts ist ein Google-Account erforderlich. Dies ist für viele vorteilhaft, da diese dadurch verschiedene Dienste in einem Account haben (z. B. Google +1, Blogger, YouTube, Drive und Android). Google Mail bietet eine E-Mail-Adresse (@googlemail.com bzw. @gmail.com) mit 15 GB Speicherplatz.



### Strato

Strato ist ein Internetdiensteanbieter für Hosting, Domains und Server. Die Strato AG ist eine Tochtergesellschaft der Deutschen Telekom und ist mit ca. 4 Millionen Domains der zweitgrößte Anbieter Europas. Strato bietet wie 1&1 einen kostenpflichtigen E-Mail-Dienst an, welcher direkt mit einer .de-Domain verbunden ist.

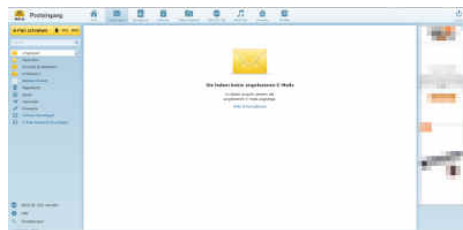
### T-Online



T-Online ist ein Webportal, welches bis 2015 von der Deutschen Telekom AG betreut wurde. Seit November 2015 gehört T-Online zur Ströer Content Group. Der E-Mail-Dienst, welcher auf T-Online zu finden ist, wird weiterhin von der Deutschen Telekom betreut. Der E-Mail-Tarif von T-Online bietet 10 E-Mail-Adressen (@t-online.de) und 10 GB Speicher.

### Web.de

Web.de ist, wie GMX auch, ein Webportal, auf welchem aktuelle Nachrichten zu finden sind. Bei Web.de wird ein E-Mail-Tarif angeboten, welcher kostenlos erhältlich ist und 1 GB Speicherplatz und 1 E-Mail-Adresse (@web.de) enthält. Web.de wurde aufgekauft und gehört in der Zwischenzeit zur 1&1 Internet SE.

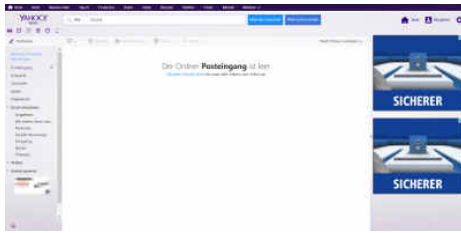


<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislinsen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Provider](#) » [E-Mail](#)

## Yahoo



Yahoo ist eines der bekanntesten Webportale weltweit. Auf dem Webportal sind Nachrichten zu finden und eine Suchleiste für die Yahoo-eigene Suchmaschine. Des Weiteren bietet Yahoo einen kostenlosen E-Mail-Dienst an, welcher 2 E-Mail-Adressen (@yahoo.de) sowie 1 TB Speicherplatz umfasst.

## Vergleich

	Postfächer	Adressen	Speicherplatz	POP3	IMAP	SSL	kostenlos
<b>1&amp;1 Basic</b>	1 oder 20	∞	2 GB pro Postfach	ja	ja	ja	nein
<b>1&amp;1 Buisness</b>	1, 2, 3, 4, 5, 10, 25, 50, 100	∞	50 GB pro Postfach	ja	ja	ja	nein
<b>GMX FreeMail</b>	1	2	1 GB	ja	nein	ja	ja
<b>GMX ProMail</b>	1	10	5 GB	ja	ja	ja	nein
<b>GMX TopMail</b>	1	50	10 GB	ja	ja	ja	nein
<b>Google Mail</b>	1	1	15 GB	ja	ja	ja	ja
<b>Strato Mail</b>	25 Basic	250	2 GB pro Postfach	ja	ja	ja	nein
<b>Strato Mail Pro3</b>	22 Basic und 3 Premium	250	2 GB pro Postfach	ja	ja	ja	nein
<b>T-Online</b>	1	10	1 GB	ja	ja	ja	ja
<b>Web.de</b>	1	1	1 GB	ja	ja	ja	ja
<b>Yahoo</b>	1	2	1 TB	ja	ja	ja	ja

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Provider](#) » [Domain](#)

## Domain

Domain-Namen (z. B. [www.homepage-webhilfe.de](#)) sind die Namen, welche im Internet zur Identifizierung von Websites genutzt werden. Hinter jeder Website steckt in Realität eine IP-Adresse, welche im Hintergrund zur Kommunikation genutzt wird, doch diese kann man sich einfach nicht so gut merken wie DNS-Namen (mal abgesehen davon, dass sich die IP-Adressen öfters mal ändern können). Domain-Namen lösen dieses Problem. Unterschieden wird beim Domain-Kauf durch den letzten Teil (Top-Level-Domain): de, com, eu, net, org, ... Die preislichen Unterschiede zwischen den Anbietern sind hierbei eher gering.

### Inhalt dieser Seite:

1. 1&1
2. DomainFactory
3. Host Europe
4. Strato

## 1&1

1&1 bietet sowohl die klassischen Domains (.de, .eu, .com, .info, .net, .org) sowie weitere „ausgefallene“ Endungen (.gmbh, .berlin, .online, ...) an. Alle Domains von 1&1 enthalten 1 E-Mail-Account mit 2 GB Speicher und 1 SSL-Zertifikat. Zudem können Sie die Domains kostenlos mit Hilfe von Subdomains unterteilen (z. B. [blog.homepage-webhilfe.de](#), [forum.homepage-webhilfe.de](#)).

## DomainFactory

DomainFactory ist ein Tochterunternehmen von Host Europe und bietet neben den klassischen Domains auch viele weitere Endungen an. SSL-Zertifikate und E-Mail-Postfächer sind optional erhältlich.

## Host Europe

Host Europe bietet eine Vielzahl von Domainendungen, darunter natürlich auch die „Top 5“ (.de, .com, .eu, .net, .org), wie sie von Host Europe bezeichnet wird. Wie auch bei DomainFactory sind SSL-Zertifikate und E-Mail-Postfächer in den Domain-Angeboten nicht enthalten.

## Strato

Beim Vergleich der Domain-Anbieter darf natürlich auch Strato nicht fehlen. Strato bietet viele verschiedene Domain-Endungen an. Im Domain-Paket enthalten ist zudem ein 2 GB großes E-Mail-Postfach und Subdomains. Des Weiteren ist die Verwendung von Umlauten in Domain-Namen möglich.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Provider](#) » [Hosting](#)

## Hosting

Unter Hosting bzw. im Bereich der Websites korrekterweise Webhosting genannt, versteht man das Bereitstellen von Webspace (Speicherplatz der Website). Zumeist enthalten Hosting-Pakete zugleich eine Domain-Adresse. Zudem laufen auf den Servern der Webhoster Programme wie PHP und Perl sowie Datenbanksysteme (z. B. MySQL), um dem Betreiber zu ermöglichen, „professionelle und komplexe Websites“ erstellen zu können. FreeHoster sind Anbieter, die ein solches Hosting-Angebot kostenlos zur Verfügung stellen. Diese verfügen meist über deutliche Einschränkungen und / oder Werbung. Am Ende dieser Seite finden Sie wieder eine Tabelle, welche nochmals alle Informationen der verschiedenen Tarife der Anbieter enthalten.

**Inhalt dieser Seite:**

1. 1&1
2. Alfahosting
3. bplaced
4. DomainFactory
5. Strato
6. Vergleich

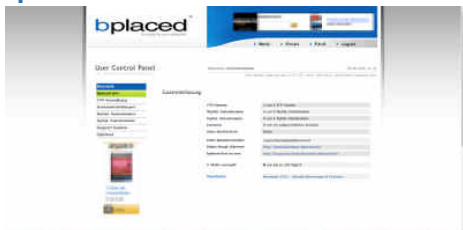
### 1&1

1&1 ist natürlich auch bei den Hosting-Angeboten vertreten. Die 4 verschiedenen Tarife unterscheiden sich vom Speicherplatz, der Domain-Anzahl und der Datenbank-Anzahl. Bereits ab dem 2. Tarif sind Speicherplatz und Datenbank-Anzahl unbegrenzt. Des Weiteren bietet 1&1 als eine der wenigsten auch Windows-Hosting an. Hier kommt also kein Linux-Betriebssystem, sondern ein Windows-Betriebssystem zum Einsatz. Windows-Hosting ist immer dann notwendig, wenn Sie .NET-Anwendungen (ASP.NET) auf dem Webserver betreiben möchten. Beim Windows-Hosting gibt es bei 1&1 nochmals zusätzlich 3 Tarife, die sich von der Domain-Anzahl und der Datenbank-Anzahl unterscheiden. Alle Tarife verfügen über die Unterstützung von PHP und Perl sowie eine unbegrenzte Anzahl an E-Mail-Adressen und die Verschlüsselung per SSL.

### Alfahosting

Alfahosting ist ein günstiger Hosting-Anbieter, bei welchem jedoch der Traffic im 1. Tarif auf 10 GB begrenzt ist. Alfahosting bietet 3 verschiedene Tarife, die sich in Speicherplatz, Datenbank-Anzahl und E-Mail-Adressen-Anzahl unterscheiden. Zudem ist die Unterstützung von PHP nur im Tarif „Starter XL“ und „Starter XXL“ vorhanden. Perl-Unterstützung ist nur im Tarif „Starter XXL“ vorhanden.

### bplaced



bplaced ist wohl hauptsächlich als FreeHoster bekannt. Der FreeHoster bietet zwei unterschiedliche kostenlose Tarife, welche sich in Speicherplatz und Datenbank-Anzahl gegensätzlich unterscheiden. Des Weiteren bietet bplaced noch einen kostenpflichtigen Tarif an. Dieser besitzt mehr Speicherplatz und mehr Datenbanken im Vergleich zu beiden kostenlosen Tarifen. Die Domain-Adressen bei allen Tarifen enden mit .bplaced.net. Als Datenbanken kommen MySQL und PostgreSQL zum Einsatz. Auf bplaced wird keine Werbung geschaltet, was für FreeHoster eher ungewöhnlich ist.

[Zur Test-Website ...](#)

### DomainFactory

Neben Domains bietet DomainFactory auch Hosting-Angebote an. Bei DomainFactory gibt es 5 Tarife, die sich vom Speicherplatz, Datenbank-Anzahl und Domain-Anzahl unterscheiden. Zu beachten ist, dass die ersten zwei Tarife keine Domain enthalten. Die Anzahl der E-Mail-Adressen ist bei allen Tarifen unbegrenzt. Unterstützung für die serverseitigen Programmiersprachen PHP und Perl ist vorhanden. Bei allen Tarifen ist der Traffic unbegrenzt.

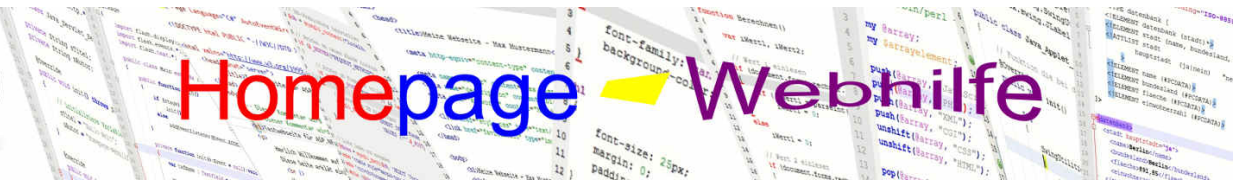
### Strato

Strato bietet 4 verschiedene Hosting-Tarife an, welche alle unter einem Linux-Betriebssystem betrieben werden. Alle Tarife enthalten ein SSL-Zertifikat sowie die Unterstützung von PHP und Perl. Die Tarife unterscheiden sich in Domain-Anzahl, Speicherplatz, Datenbank-Anzahl und E-Mail-Adressen. Als Datenbanksystem kommt MySQL zum Einsatz. Traffic ist bei allen Tarifen von Strato unbegrenzt.

## Vergleich

	Speicher	Traffic	Domains	Datenbanken	E-Mail-Adressen	PHP	Perl	ASP.NET	SSL	kostenlos
1&1 Starter	50 GB	∞	1	2 MySQL	∞	ja	ja	nein	ja	nein
1&1 Unlimited	∞	∞	2	∞ MySQL	∞	ja	ja	nein	ja	nein
1&1 Unlimited Plus	∞	∞	3	∞ MySQL	∞	ja	ja	nein	ja	nein
1&1 Unlimited Pro	∞	∞	4	∞ MySQL	∞	ja	ja	nein	ja	nein
1&1 Unlimited Windows	∞	∞	2	5 MSSQL	∞	ja	ja	ja	ja	nein
1&1 Unlimited Plus Windows	∞	∞	3	10 MSSQL	∞	ja	ja	ja	ja	nein
1&1 Unlimited Pro Windows	∞	∞	4	25 MSSQL	∞	ja	ja	ja	ja	nein
Alfahosting Starter L	250 MB	10 GB	1	-	50	nein	nein	nein	nein	nein
Alfahosting Starter XL	1 GB	∞	1	1 MySQL	100	ja	nein	nein	nein	nein
Alfahosting Starter XXL	2 GB	∞	1	4 MySQL	500	ja	ja	nein	nein	nein
bplaced freestyle	1 GB	∞	1	8 MySQL + 8 PostgreSQL	-	ja	nein	nein	nein	ja

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p><b>Web:</b> <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> <b>E-Mail:</b> <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Provider](#) » [Hosting](#)

<b>bplaced freestyle plus</b>	2 GB	∞	1	2 MySQL + 2 PostgreSQL	-	ja	nein	nein	nein	ja
<b>bplaced pro</b>	4 GB	∞	1	16 MySQL + 16 PostgreSQL	-	ja	nein	nein	nein	nein
<b>DomainFactory Basic</b>	25 GB	∞	-	1 MySQL	∞	ja	ja	nein	nein	nein
<b>DomainFactory Medium</b>	50 GB	∞	-	10 MySQL	∞	ja	ja	nein	nein	nein
<b>DomainFactory Professional</b>	100 GB	∞	1	1000 MySQL	∞	ja	ja	nein	nein	nein
<b>DomainFactory Premium</b>	200 GB	∞	2	∞ MySQL	∞	ja	ja	nein	ja	nein
<b>DomainFactory Ultimate</b>	400 GB	∞	5	∞ MySQL	∞	ja	ja	nein	ja	nein
<b>Strato Starter</b>	30 GB	∞	1	2 MySQL	10000	ja	ja	nein	ja	nein
<b>Strato Basic</b>	60 GB	∞	2	25 MySQL	20000	ja	ja	nein	ja	nein
<b>Strato Plus</b>	120 GB	∞	3	50 MySQL	40000	ja	ja	nein	ja	nein
<b>Strato Pro</b>	200 GB	∞	4	75 MySQL	100000	ja	ja	nein	ja	nein

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

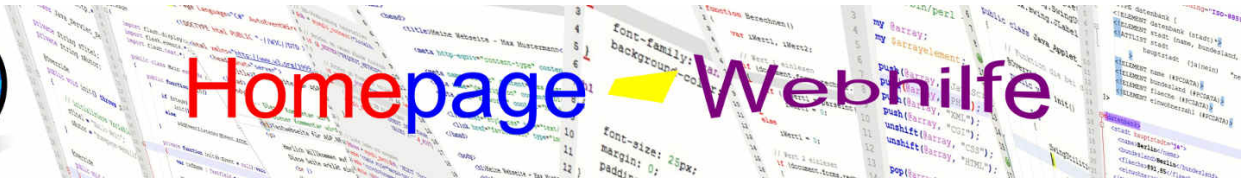
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Provider](#) » [Server](#)

## Server

Server bzw. Server-Hosting sind eine spezielle Art von Hosting. Vorteil von Server-Hosting im Gegensatz zu Webhosting ist, dass beim Server-Hosting meist ein Vollzugriff auf die Server möglich ist. Dies ist dann nützlich, wenn auf dem Webserver individuelle Programme installiert oder spezielle Konfigurationen vorgenommen werden müssen. Bei einem Server-Hosting wird zwischen virtuellen Servern und dedizierten Servern unterschieden. Bei virtuellen Servern handelt es sich um „einen Teil eines Servers“. Hier werden mehrere Server mit Hilfe von Virtualisierung auf einem „echten“ Server betrieben. Dedizierte Server sind direkt ein „echter Server“, weshalb dedizierte Server auch deutlich teurer sind. Auf Grund der hohen Anzahl an verschiedenen Angeboten, wollen wir Ihnen hier lediglich virtuelle Server vorstellen. Der Vergleich am Ende der Seite zeigt die unterstützten Betriebssysteme, die CPU-Kerne, den Arbeitsspeicher und den Speicherplatz der unterschiedlichen Tarife auf. Alle hier vorgestellten Server-Hosting-Angebote sind kostenpflichtig.

### Inhalt dieser Seite:

1. 1&1
2. Alfahosting
3. Host Europe
4. Strato
5. Vergleich

### 1&1

1&1 bietet 4 verschiedene Tarife für virtuelle Server an. Im 1. Tarif wird nur Linux als Betriebssystem unterstützt. Auf allen Servern ist PHP, Perl und das MySQL-Datenbanksystem vorinstalliert. SSD-Speicher sind optional erhältlich. Alle Tarife enthalten bereits eine Domain.

### Alfahosting

Alfahosting bietet neben seinen Webhosting-Angeboten auch Tarife für virtuelle Server an. Hierfür werden 4 verschiedene Tarife angeboten, welche alle sowohl in einer Linux- als auch in einer Windows-Variante erhältlich sind. Die virtuellen Server sind mit SSD-Speichermedien ausgestattet.

### Host Europe

Host Europe bietet 6 verschiedene Tarife für virtuelle Server an. Ab dem 3. Tarif („Advanced“) ist ein SSL-Zertifikat inklusive. Zum Einsatz kommen Markenserver von HP und SSD-Speicher.

### Strato

Strato bietet 5 verschiedene Tarife an, welche alle als Windows- und Linux-Variante erhältlich sind. Als Speichermedien kommen HP 3PAR Speicher zum Einsatz (Kombination von SSD und HDD: oft abgerufene Daten auf SSD, seltener abgerufene Daten auf HDD). Alle V-Server-Angebote von Strato werden mit SSL-Zertifikaten ausgeliefert und besitzen eine Unterstützung für PHP, Perl, Python und MySQL.

## Vergleich

	Betriebssystem	CPU	RAM	Speicherplatz
1&1 Virtual Server M	Linux	1 vCore	1 GB	50 GB
1&1 Virtual Server L	Linux oder Windows	2 vCore	2 GB	150 GB
1&1 Virtual Server XL	Linux oder Windows	4 vCore	4 GB	300 GB
1&1 Virtual Server XXL	Linux oder Windows	6 vCore	6 GB	400 GB
Alfahosting VServer M	Linux oder Windows	1 vCore	2 GB	55 GB
Alfahosting VServer L	Linux oder Windows	2 vCore	4 GB	155 GB
Alfahosting VServer XL	Linux oder Windows	4 vCore	8 GB	260 GB
Alfahosting VServer XXL	Linux oder Windows	8 vCore	16 GB	550 GB
Host Europe Starter	Linux oder Windows	2 vCore	2 GB	100 GB
Host Europe Starter Plus	Linux oder Windows	2 vCore	4 GB	150 GB
Host Europe Advanced	Linux oder Windows	4 vCore	6 GB	200 GB
Host Europe Advanced Plus	Linux oder Windows	4 vCore	8 GB	400 GB
Host Europe Enterprise	Linux oder Windows	6 vCore	16 GB	600 GB
Host Europe Enterprise Plus	Linux oder Windows	10 vCore	32 GB	800 GB
Strato V-Server Level 1	Linux oder Windows	2 vCore	2 GB	300 GB
Strato V-Server Level 2	Linux oder Windows	4 vCore	4 GB	500 GB
Strato V-Server Special Edition	Linux oder Windows	6 vCore	6 GB	600 GB
Strato V-Server Level 3	Linux oder Windows	8 vCore	8 GB	800 GB
Strato V-Server Level 4	Linux oder Windows	16 vCore	16 GB	1000 GB

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

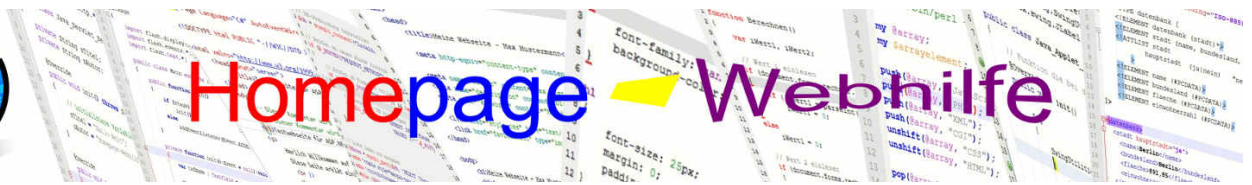
#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Provider](#) » [Homepage-Baukasten](#)

## Homepage-Baukasten

Ein Homepage-Baukasten (oder auch Website-Baukasten) ist ein CMS-System (Content Management System), welches von einem Provider zum Erstellen von Websites, zumeist mittels Webbrowser, zur Verfügung gestellt wird. Vorteil von Homepage-Baukästen ist, dass keinerlei Programmierkenntnisse erforderlich sind und das Zusammenstellen der Inhalte meist mittels Drag'n'Drop erfolgt. Anders wie bei CMS-Systemen wie z. B. WordPress, Joomla und TYPO3 ist keine Installation und Übertragung der Dateien auf den Webserver notwendig. Bei Homepage-Baukästen verfügt der Kunde über keinen direkten FTP-Zugang auf den Webserver. Übertragen von Bildern oder anderen Dateien zur Einbindung auf der Website erfolgt ebenfalls über den Webbrowser. Homepage-Baukästen werden von einigen Providern kostenlos angeboten, andere verlangen wiederum Geld dafür. Die Unterschiede zwischen den Providern und Tarifen sind in den Texten erklärt und in der Tabelle am Seitenende aufgeführt. Alle kostenlose Anbieter wurden von uns getestet.

### Inhalt dieser Seite:

1. 1&1
2. Jimdo
3. Strato
4. Webnode
5. Weebly
6. Wix
7. Vergleich

### 1&1

1&1 bietet mit 1&1 MyWebsite 4 verschiedene Tarife für Homepage-Baukästen an. Alle Tarife enthalten bereits eine Domain und ein SSL-Zertifikat. Die Unterschiede in den Tarifen sind bei der Anzahl von auswählbaren Designs und beim Speicherplatz zu erkennen. Ab dem 3. Tarif sind zusätzlich Tools zum Online-Marketing (SEO-Optimierung, Facebook, Newsletter) mit inbegriffen. Ein Online-Shop ist lediglich im 4. Tarif inklusive, welcher jedoch auf 1000 Produkte eingeschränkt ist.

### Jimdo



Jimdo ist ein Anbieter, welcher vor allem durch seinen kostenlosen Homepage-Baukasten bekannt ist. Zusätzlich existieren jedoch noch zwei weitere Tarife, welche kostenpflichtig sind. Die kostenpflichtigen Tarife enthalten im Gegensatz zum kostenlosen Tarif eine eigene Domain und ein E-Mail-Konto. Der kostenlose Tarif enthält lediglich eine Domain mit der Endung .jimdo.com und zudem in der Fußzeile einen Werbelink mit Icon. Weitere Unterschiede zwischen den Tarifen sind im Speicherplatz, in der Suchmaschinenoptimierung und im Online-Shop zu erkennen.

[Zur Test-Website ...](#)

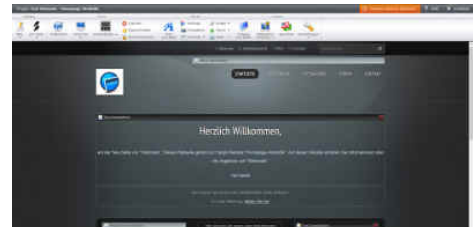
### Strato

Strato bietet einen kostenpflichtigen Tarif als Homepage-Baukasten an. Der Tarif enthält bereits eine Domain, E-Mail-Adressen, ein SSL-Zertifikat und unbegrenzten Traffic. Zudem bietet Strato einen zweiten Tarif an, welcher mit dem ersten Tarif vollständig identisch ist, abgesehen von dem Unterschied, dass dieser Tarif über bessere Suchmaschinen-Optimierungen (SEO) verfügt.

### Webnode

Webnode bietet sowohl einen kostenlosen Tarif an sowie 3 kostenpflichtige Tarife. Die Unterschiede in den Tarifen sind in der Traffic-Begrenzung, im Speicherplatz, in der Anzahl der E-Mail-Adressen und natürlich im Preis zu erkennen. Alle Tarife verfügen standardmäßig nur über die Webnode-eigene Domain-Endung .webnode.com. Bei allen kostenpflichtigen Tarifen besteht jedoch die Möglichkeit, eine Domain-Adresse hinzuzukaufen. Zu beachten ist auch, dass in den ersten zwei Tarifen in der Fußzeile ein kleiner Werbebanner zu finden ist.

[Zur Test-Website ...](#)



### Weebly



Weebly bietet 4 verschiedene Tarife an, wovon der 1. Tarif kostenlos erhältlich ist. Wie auch bei Webnode enthalten alle Pakete standardmäßig nur eine firmeneigene Endung (.weebly.com). Die Verknüpfung mit einer vollständigen Domain-Adresse ist jedoch möglich (Wichtig: Es fallen zusätzliche Gebühren für das Mieten der Domain an.). Der 4. Tarif enthält zudem ein SSL-Zertifikat. Das kostenlose Paket ist im Speicherplatz beschränkt, bei allen weiteren Tarifen existiert keine Speicherplatz-Grenze. Weitere Unterschiede zwischen den Tarifen sind vor allem im Bereich E-Commerce zu erkennen. Des Weiteren ist beim kostenlosen Tarif eine Branding in der Fußzeile zu finden.

[Zur Test-Website ...](#)

### Wix

Wix ist ein Homepage-Baukasten-Angebot der internationalen Firma Wix.com. Wix bietet 4 kostenpflichtige Tarife sowie 1 kostenlosen Tarif an. Der 1. kostenpflichtige Tarif nennt sich „Connect Domain“ und bietet auch nicht mehr, als er vom Namen vermuten lässt, an. Im Gegensatz zum kostenlosen Tarif ist nämlich lediglich die Verbindung mit einer bereits bestehenden Domain (muss extra bezogen werden) möglich. Die Tarife unterscheiden sich im Hinblick auf Speicherplatz und Traffic (Bandbreite). Ein Online-Shop ist lediglich im letzten und teuersten Tarif („eCommerce“) enthalten. Das Firmen-Branding in der Fußzeile kann erst ab dem 3. Tarif („Combo“) entfernt werden. Die Domain von Wix trägt das Kürzel .wix.com, ist jedoch noch zusätzlich mit einem Ordnernamen verbunden. Wix gilt in der Zwischenzeit als einer der modernsten Provider für Homepage-Baukästen. Das Bearbeiten und Hinzufügen von Inhalten ist eine Mischung aus einfacher Bedienung und präziser Platzierung.

[Zur Test-Website ...](#)



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

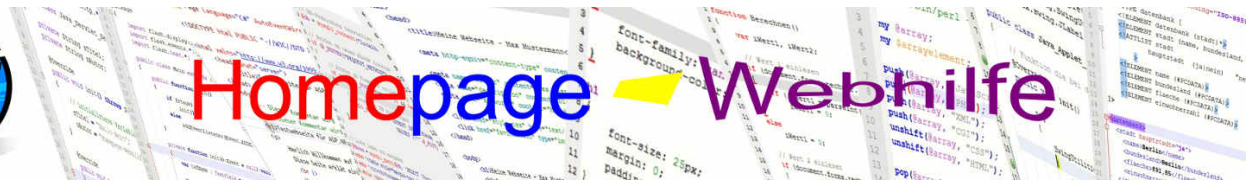
#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Provider](#) » [Homepage-Baukasten](#)

## Vergleich

	Traffic	Speicherplatz	Domains	E-Mail-Adressen	Werbung	SSL	kostenlos
1&1 MyWebsite Personal	∞	10 GB	1	200	nein	ja	nein
1&1 MyWebsite Business Basic	∞	50 GB	1	200	nein	ja	nein
1&1 MyWebsite Business Plus	∞	50 GB	1	200	nein	ja	nein
1&1 MyWebsite Business Pro	∞	50 GB	1	200	nein	ja	nein
Jimdo Free	∞	500 MB	-	-	ja	nein	ja
Jimdo Pro	∞	5 GB	1	1	nein	nein	nein
Jimdo Business	∞	∞	1	20	nein	nein	nein
Strato Homepage-Baukasten Pro	∞	∞	1	500	nein	ja	nein
Webnode Kostenlos	1 GB	100 MB	-	-	ja	nein	ja
Webnode Mini	3 GB	500 MB	-	1	ja	nein	nein
Webnode Standard	10 GB	2 GB	-	20	nein	nein	nein
Webnode Profi	∞	5 GB	-	100	nein	nein	nein
Weebly Kostenlos	∞	500 MB	-	-	ja	nein	ja
Weebly Starter	∞	∞	-	-	nein	nein	nein
Weebly Pro	∞	∞	-	-	nein	nein	nein
Weebly Business	∞	∞	-	-	nein	ja	nein
Wix Free	1 GB	500 MB	-	-	ja	nein	ja
Wix Connect Domain	1 GB	500 MB	-	-	ja	nein	nein
Wix Combo	2 GB	3 GB	-	-	nein	nein	nein
Wix Unlimited	∞	10 GB	-	-	nein	nein	nein
Wix eCommerce	10 GB	20 GB	-	-	nein	nein	nein

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

Java EE

XML

# E-Book

## HTML





Sie befinden sich hier: Homepage-Webhilfe » HTML » Einführung

## Einführung



HTML (Hypertext Markup Language) ist eine Auszeichnungssprache und dient als Basis aller Webseiten. Egal ob Webseiten mit Sprachen wie PHP, Perl oder ASP.NET dynamisch erstellt werden, oder mit Sprachen wie JavaScript aktiv verändert werden, letztendlich existiert bzw. entsteht immer ein HTML-Dokument.

HTML unterstützt von Haus aus sehr wenige Formatierungsmöglichkeiten, weshalb hierfür externe Dokumente notwendig sind (sogenannte Stylesheets), mit welchen eine Seite formatiert werden kann. Die Auszeichnungssprache HTML dient also nur dazu, den Inhalt zu strukturieren und zu gliedern. Als Stylesheet-Sprache kommt [CSS](#) zum Einsatz, wozu Sie auf unserer Website ebenfalls ein Tutorial finden.

HTML-Dokumente werden vom Webserver an den Webbrowser unverändert gesendet, dadurch kann jeder den HTML-Code von anderen Webseiten betrachten. Natürlich kann auch jeder den HTML-Code Ihrer Website betrachten, sofern sich die Website frei zugänglich im World Wide Web befindet. Der HTML-Code wird von jedem Browser selbst interpretiert und verarbeitet, was zur Folge hat, dass nicht immer alle Inhalte in jedem Browser gleich aussehen.

HTML-Code kann theoretisch mit jedem Texteditor geschrieben werden, da der Code nicht kompiliert werden muss, jedoch sollte jeder einen professionelleren Editor nutzen, welcher Syntax-Highlighting und Autovervollständigung anbietet. Dadurch können wir unseren HTML-Code nicht nur schneller erstellen, sondern auch einige Syntax-Fehler vermeiden. Eine Vorstellung verschiedener Editoren finden Sie in unserem [Software-Bereich](#). HTML-Dateien haben üblicherweise die Dateiendung htm oder html.

Inhalt dieser Seite:
1. Geschichte
2. Syntax
3. Aufbau

## Geschichte

Die Geschichte von HTML beginnt gemeinsam mit der Geschichte des World Wide Webs (W3) bereits im Jahre 1989. HTML wurde als Auszeichnungssprache von Tim Berners-Lee entwickelt und diente schon damals zum Gliedern von Inhalten. Durch das ebenfalls von Berners-Lee entwickelte HTTP-Protokoll wurden diese Dokumente durch einen Webserver an den Browser gesendet. Die erste HTML-Spezifikation erschien Ende 1992. 1994 wurde von Tim Berners-Lee das World Wide Web Consortium (W3C) gegründet, welches sich mit der Spezifikation der Technologien rund um das World Wide Web beschäftigt.

Natürlich wurde die Sprache HTML mehrmals erweitert und verbessert. Eine sehr lange verwendete Version war HTML 4.01, welche bereits Ende 1999 erschien. Auch heute wird noch auf einigen Websites HTML 4.01 eingesetzt. Eine Abwandlung von HTML war XHTML, welches im Gegensatz zu HTML nicht auf SGML, sondern auf XML basieren sollte. Die Entwicklung von XHTML wurde 2009 jedoch von seitens des W3Cs eingestellt, da der Fokus vollständig auf HTML5 gelegt werden sollte. Seit Oktober 2014 ist HTML5 die aktuelle und empfohlene Version von HTML. HTML5 ist eine vollständig eigene Sprache, welche jedoch wieder (so wie HTML4 auch) auf SGML basiert.

## Syntax

Um ein HTML-Dokument strukturieren zu können, gibt es in HTML sogenannte **Elemente**. Die meisten Elemente verfügen hierbei über einen sogenannten **Start- und End-Tag**. Alle Tags beginnen immer mit der öffnenden spitzen Klammer < (kleiner-als) und enden mit der schließenden spitzen Klammer > (größer-als). Zwischen den spitzen Klammern wird der Name des Tags angegeben. Beim End-Tag muss zusätzlich zwischen der öffnenden spitzen Klammer und dem Tag-Name ein Schrägstrich / (Slash) notiert werden. Zwischen dem Start- und End-Tag kann weiterer **Inhalt** notiert werden. Dabei kann es sich um ein weiteres Element (oder dazu mehrere), aber auch um reinen Text handeln. Ein Element besteht also aus 3 Teilen: Start-Tag, Inhalt und End-Tag. Das folgende Beispiel zeigt den `p`-Tag (dazu später mehr) mit Text als Inhalt:

```
1 | <p>Hier steht ein Text.</p>
```

Hier ein weiteres Beispiel bei welchem in einem `div`-Tag mehrere `p`-Tags untergeordnet sind:

```
1 | <div>
2 |   <p>Hier steht ein Text.</p>
3 |   <p>Hier steht ein Text.</p>
4 |   <p>Hier steht ein Text.</p>
5 | </div>
```

Bei einer solchen Verschachtelung wird auch gerne von einer Baumstruktur gesprochen. Dieser Begriff kommt unter anderem von der Auszeichnungssprache XML, denn auch hier entspricht der logische Aufbau des Dokuments einer **Baumstruktur**. Der Vergleich der Struktur eines XML- oder HTML-Dokuments mit einem Baum kommt von den Verzweigungen eines Baums: Es gibt ein Hauptelement (Stamm), dieses verzweigt sich in Elemente (Äste) und kann sich nun in weitere Elemente (Äste) oder in Texte (Blätter) verzweigen.

Neben den bisher kennengelernten klassischen Elementen, welche aus Start-Tag, Inhalt und End-Tag bestehen, gibt es auch noch **einteilige Elemente** (auch leere Elemente genannt). Einteilige Elemente verfügen nur über einen Tag (den Start-Tag) und keinen Inhalt. Ein Beispiel ist das `br`-Element, welches wir ebenfalls später kennenlernen werden:

```
1 | <br>
```

Hierbei gilt jedoch zu beachten, dass der obige Code nur in den alten HTML-Versionen sowie in HTML5 gültig ist, nicht jedoch in XHTML. In XHTML müssten wir vor dem >-Zeichen einen Schrägstrich notieren. Wenn wir unseren Code XML-kompatibel halten wollen, dann sollten wir den Schrägstrich auch in HTML5 notieren. Die unten gezeigte Variante mit dem Schrägstrich ist nicht nur gültiges XHTML, sondern auch gültiges HTML5. Alle Beispiele auf unserer Website sind vom Syntax XML-kompatibel. Ein weiterer Vorteil dieser Variante ist, dass der Code einfacher lesbar ist, da man einteilige und mehrteilige Elemente klar erkennen und auseinanderhalten kann. Deshalb empfehlen wir Ihnen die folgende (zweite) Variante:

```
1 | <br />
```

Zusätzlich zu den Elementen gibt es noch sogenannte **Attribute**. Attribute werden innerhalb des Start-Tags von Elementen angegeben. Elemente können keine, ein oder mehrere Attribut(e) besitzen. Attribute bestehen ebenfalls aus 3 Teilen: Attributname, Gleichheitszeichen und Wert. Der Wert sollte dabei in doppelten Anführungszeichen notiert werden. Zwischen Attributname und Elementname sowie zwischen Attributen müssen bzw. sollten Leerzeichen notiert werden. Das folgende Beispiel zeigt ein `p`-Element mit dem Attribut `id` und dem Wert `einleitung`:

```
1 | <p id="einleitung">Hier steht ein Text.</p>
```

Einige Attribute wie z. B. das `disabled`-Attribut für `input`-Elemente können in verkürzter Weise notiert werden. Hierzu folgendes Beispiel:

```
1 | <input type="text" disabled />
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--





Sie befinden sich hier: Homepage-Webhilfe » HTML » Einführung

Auch hier ist zu beachten, dass die Verkürzung von Attributen in XHTML nicht erlaubt ist. Jedoch empfehlen wir auch hier die Verkürzung nur zu verwenden, wenn Sie sicher sind, dass das HTML-Dokument nicht von XML-Programmen verarbeitet wird. Unsere Beispiele verwenden die Attributs-Verkürzung nicht. Folgender Code ist mit obigem Beispiel von der Funktion identisch:

```
1 | <input type="text" disabled="disabled" />
```

**Kommentare** werden zwar in HTML-Dokumenten seltener verwendet, jedoch ist es während der Entwicklungs-Phase ganz praktisch, wenn Programmteile einfach auskommentiert werden können. HTML-Kommentare werden vom Webbrowser nicht verarbeitet und dienen eigentlich zu Dokumentationszwecken. Ein HTML-Kommentar beginnt mit den Zeichen <!-- und endet mit den Zeichen -->.

```
1 | <!-- Dies ist ein Kommentar! -->
```

**Übrigens:** Zwar schreibt die HTML5-Spezifikation nicht vor, ob Elemente oder Attribute klein oder groß geschrieben werden müssen, trotzdem hat es sich eingebürgert, Element- und Attributnamen klein zu schreiben. An diese ungeschriebene Regel halten wir uns und sollten auch Sie sich halten, denn Kleinbuchstaben sind wesentlich leichter und angenehmer zu lesen, da Großbuchstaben einfach zu dominant sind.

**Wichtig:** Die HTML-Elemente (sofern diese sowohl über einen Start-Tag als auch über einen End-Tag verfügen) müssen in dieser Reihenfolge geschlossen werden wie diese geöffnet werden. Der folgende Syntax ist ungültig:

```
1 | <i>Dies ist <b>ein</i> Programmcode!</b>
```

Der obere Code müsste für gültiges HTML wie folgt ersetzt werden:

```
1 | <i>Dies ist <b>ein</b></i><b> Programmcode!</b>
```

Wie Sie in den nächsten Code-Beispielen feststellen werden, ordnen wir Unter-elemente eines Elements immer mit Leerzeichen oder Tabs unter. Dies schreibt die HTML-Spezifikation zwar nicht vor, ist jedoch gängige Praxis und sorgt für eine **gute Struktur** sowie einen **übersichtlichen Code**. Die meisten Editoren verwenden 4 Leerzeichen bzw. einen Tab für die Unterordnung von Elementen.

### Aufbau

Die Basisstruktur (Baumstruktur) eines HTML-Dokuments ist immer gleich. Jedes HTML-Dokument verfügt über das Wurzelement `html`. Als **Wurzelement** wird das 1. Element bezeichnet, welches sich in der 1. Ebene befindet. Alle anderen Elemente oder Texte müssen sich innerhalb dieses Wurzelementes bzw. der darin verschachtelten Elemente befinden.



```
1 | <html>
2 |
3 | </html>
```

Nachdem wir nun unser Wurzelement erstellt haben, benötigen wir noch 2 weitere Elemente für unsere Basisstruktur: `head` und `body`. Beide Elemente werden innerhalb der `html`-Tags notiert. Der **Head-Bereich** stellt später, wie der Name schon sagt, den Kopfbereich dar. Dieser ist für Benutzer im Browser nicht sichtbar, ist jedoch wenn es um das Thema Suchmaschinenoptimierung geht sehr wichtig. Dazu jedoch mehr im [nächsten Thema](#). Der **Body-Bereich** (Inhaltsbereich) dient zum Platzieren unseres eigentlichen Webseiten-Inhalts. Hier werden Inhalte wie Texte, Überschriften, Bilder, Listen, Tabellen u. v. m. platziert.

```
1 | <html>
2 |   <head>
3 |
4 |   </head>
5 |
6 |   <body>
7 |
8 |   </body>
9 | </html>
```

Um unserem HTML-Dokument eine Kennung zu geben, um welche Version von HTML oder auch XHTML es sich handelt, benötigen wir den sogenannten **Doctype**. Der Doctype (eigentlich Document Type Definition bzw. Dokumenttypdefinition, kurz DTD) steht immer am Anfang der HTML-Datei und sieht so ähnlich aus, wie ein Start-Tag eines Elementes: Das Wort `doctype` mit einem vorangestellten Ausrufezeichen wird innerhalb der spitzen Klammern notiert. Zwischen dem Schlüsselwort `doctype` und der schließenden spitzen Klammer `>` müssen wir noch ein Leerzeichen und das Wort `html` notieren. Dies sieht dann so aus:

```
1 | <!doctype html>
```

In den älteren HTML-Versionen sowie in XHTML war die Definition des Dotypes komplexer, da hier eine URL zu einer **DTD-Datei** (Datei, in welcher die Regeln für den SGML- oder XML-Aufbau niedergeschrieben sind) angegeben werden musste (dazu mehr im Kapitel [XML](#)). Für HTML5 gibt es keine DTD-Datei, weshalb die Notierung des Dokumententyps nur noch aus Kompatibilitätsgründen erfolgen muss. Ein Doctype eines HTML 4.01-Dokumentes sieht wie folgt aus:

```
1 | <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

Der folgende Code entspricht dem Doctype für XHTML 1.1:

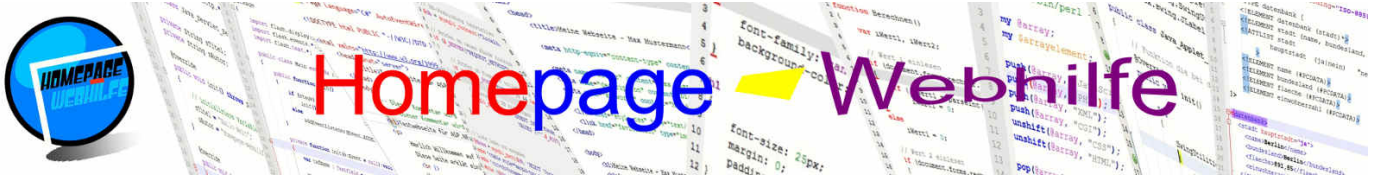
```
1 | <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Über die verschiedenen Dotypes müssen Sie sich jedoch nicht wirklich Gedanken machen, da wir ausschließlich HTML5-Code schreiben und somit immer den „einfachen Doctype“ notieren.

Aber nun zurück zu unserer ersten HTML-Seite. Den Code den wir bisher geschrieben haben, sieht wie folgt aus:

```
1 | <!doctype html>
2 |
3 | <html>
4 |   <head>
```

Footer area with navigation links: Über uns, Community, Nachschlagewerk, Benjamin Jung contact info, and website details.



Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Einführung](#)

```

5
6     </head>
7
8     <body>
9
10    </body>
11 </html>

```

Würden wir diesen in eine HTML-Datei schreiben und mit einem Browser öffnen, so würden wir eine leere Seite sehen. Dies soll sich jetzt ändern.

Im Head-Bereich werden hauptsächlich die Elemente `title`, `meta` und `link` platziert. Fürs Erste benötigen wir nur das `title`-Element. Zwischen den 2 `title`-Tags notieren wir den **Titel der Seite**. Dieser wird in der Browserleiste bzw. in der Zeile des aktuellen Tabs angezeigt.

```
1 | <title>Meine erste Webseite</title>
```

Für den Body-Bereich gibt es sehr viele unterschiedliche Elemente (`h1`, `p`, `div`, `img`, ...), welche wir im Laufe des HTML-Kurses vorstellen werden. Für unsere erste Webseite verwenden wir ausschließlich das `p`-Element, in welchem wir einen Text notieren.

```
1 | <p>Willkommen auf meiner ersten Webseite!</p>
```

Nun können wir unseren Code zusammenfügen. Das Ergebnis des Beispiels können Sie sich anschauen, indem Sie auf das Vorschau-Icon unterhalb des Codes klicken. Zuerst jedoch der fertige Code unserer ersten Webseite:

```

1 | <!doctype html>
2
3 | <html>
4 |   <head>
5 |     <title>Meine erste Webseite</title>
6 |   </head>
7
8 |   <body>
9 |     <p>Willkommen auf meiner ersten Webseite!</p>
10 |   </body>
11 | </html>

```



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

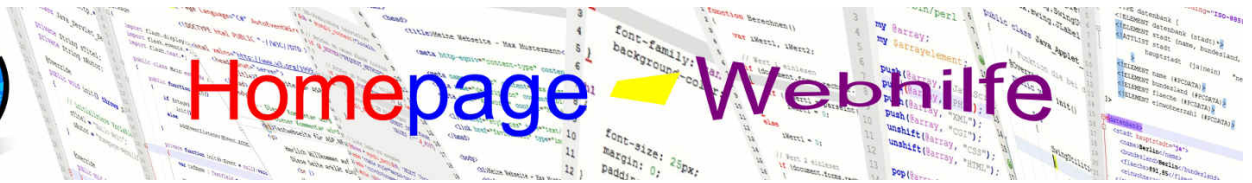
#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Head-Bereich](#)

## Head-Bereich

Der Head-Bereich dient zum Definieren von Merkmalen über die enthaltenen Daten. Dabei enthält jedoch der Head-Bereich nicht die eigentlichen Daten selbst, denn diese sind im Body-Bereich zu finden. Der Head-Bereich befindet sich durch das `head`-Element direkt im `html`-Element. Neben den verschiedenen Metadaten im Head-Bereich, sind dort auch Einbindungen von externen Daten wie Stylesheets und Skripte zu finden.

### Inhalt dieser Seite:

1. Titel
2. Meta-Angaben
3. Einbindung externer Daten

### Titel

Der Titel einer Seite wird ebenfalls innerhalb des Head-Bereichs definiert. Hierfür notieren wir den Titel innerhalb der `title`-Tags. Der Titel ist eigentlich der einzige für den Endbenutzer sichtbare Teil des Head-Bereichs. Hier nochmals ein Beispiel für einen Titel in HTML:

```
1 | <title>Meine erste Webseite</title>
```

### Meta-Angaben

Als Metadaten (oder auch Metainformationen) werden, wie bereits erwähnt, Informationen und Merkmale anderer Daten (zumeist die, welche im Body-Bereich zu finden sind) bezeichnet. Um diese Merkmale festzulegen, gibt es in HTML das leere Element `meta`. Um Metainformationen festzulegen, benötigen wir neben dem `meta`-Element die Attribute `name` und `content`. `name` legt dabei den Typ der Metainformation fest, wohingegen das `content`-Attribut für den Inhalt zuständig ist. Wichtig zu wissen ist, dass keine der Meta-Angaben für ein gültiges (valides) HTML-Dokument notwendig sind. Die Angaben aller Metadaten sind also optional.

Gerade im Bereich **Suchmaschinenoptimierung** (SEO, Search Engine Optimization) sind die Meta-Angaben `description` und `keywords` sehr wichtig. Die Seitenbeschreibung, welche mittels `description` definiert wird, sollte ungefähr 100 bis 150 Zeichen lang sein und wird bei den Suchmaschinen zumeist unterhalb des Webseiten titels und der URL angezeigt (siehe Bild). Ist die Beschreibung zu lang, so schneiden die Suchmaschinen die Beschreibung ab, was oft unschön ist. Grundsätzlich sollte die Beschreibung klar formuliert werden und Suchende dazu animieren, Ihre Webseite aufzurufen. Hier nun das erste Beispiel zu einer Meta-Angabe:

[Homepage-Webhilfe](#)  
[www.homepage-webhilfe.de/](http://www.homepage-webhilfe.de/)  
 Sie wollen Programmiersprachen für Webseiten lernen? Oder wollen Sie Tipps zu Homepages bekommen? Hier bei Homepage-Webhilfe sind Sie genau richtig.

```
1 | <meta name="description" content="Hier finden Sie alle Informationen über die Auszeichnungssprache HTML: vom Aufbau, über Listen und Tabellen bis hin zu Videos" />
```

Die Meta-Angabe `keywords` dient zum Festlegen von Schlüsselwörtern. Unter anderem mit diesen Schlüsselwörtern, können die Benutzer einer Suchmaschine Ihre Webseite finden. Schlüsselwörter (auch Stichwörter genannt) müssen durch Kommas getrennt werden. Für eine bessere Lesbarkeit wird nach dem Komma zumeist ein Leerzeichen notiert.

```
1 | <meta name="keywords" content="HTML, Head, Body, Überschrift, Text, Link, Liste, Tabelle, Formular" />
```

Mit dem Typ `author` können wir den Autor und somit die Person, welche für den Inhalt verantwortlich ist, festlegen:

```
1 | <meta name="author" content="Max Mustermann" />
```

Des Weiteren können wir mit dem Typ `publisher` die Person oder Firma festlegen, welche den Webaufritt herausgegeben hat. Bei privaten und selbst erstellten Webseiten haben die Meta-Angaben für den Autor und Herausgeber meist den gleichen Wert.

```
1 | <meta name="publisher" content="Max Mustermann" />
```

Um die Steuerung von Webcrawlern festzulegen, können wir im `name`-Attribut des `meta`-Elements den Wert `robots` eintragen. **Webcrawler** sind Programme, welche die Aufgabe haben, das Internet zu durchsuchen und sich dabei von Link zu Link durchzuarbeiten. Dadurch ist eine Erfassung eines Großteils des Internets möglich. Als Wert für das `content`-Attribut kommen hauptsächlich die Schlüsselwörter `index` und `follow` sowie die gegenteiligen Schlüsselwörter `noindex` und `nofollow` zum Einsatz. Daraus ergeben sich nun theoretisch 4 verschiedene Möglichkeiten.

Die Kombination `index, follow` bedeutet, dass die aktuelle Seite in den Seitenindex der Suchmaschine aufgenommen (fachsprachlich **indexiert**) werden soll und die darin enthaltenen Links vom Webcrawler verfolgt (also ebenfalls aufgerufen und geprüft) werden sollen. Dies ist die gängigste Notation, wenn es darum geht, dem Webcrawler explizit mitzuteilen, dass diese Webseite vollständig verarbeitet werden soll.

```
1 | <meta name="robots" content="index, follow" />
```

Als gegensätzliche Notation zu `index, follow` gibt es die Kombination `noindex, nofollow`. Damit teilen wir dem Webcrawler mit, die aktuelle Webseite weder zu indexieren, noch die dort enthaltenen Links zu verfolgen. Dies eignet sich ideal für Seiten, welche für Besucher nicht relevant sind bzw. nicht zugänglich sein sollten (z. B. ein Admin-Bereich).

```
1 | <meta name="robots" content="noindex, nofollow" />
```

Durch die Anweisung `noindex, follow` wird dem Webcrawler mitgeteilt, die aktuelle Seite zwar nicht in den Seitenindex aufzunehmen, jedoch die auf der Seite enthaltenen Links weiter zu verfolgen.

```
1 | <meta name="robots" content="noindex, follow" />
```

Als 4. und letzte Möglichkeit ergibt sich die Kombination `index, nofollow`. Dadurch wird die Seite in den Index aufgenommen, die dort enthaltenen Links jedoch nicht verfolgt.

```
1 | <meta name="robots" content="index, nofollow" />
```

Eine weitere wichtige Meta-Angabe ist die Angabe der **Zeichenkodierung**. Hierfür benötigen wir ebenfalls das `meta`-Element, jedoch nicht das `name` und `content` Attribut. Die Angabe der Zeichenkodierung erfolgt mittels des `charset`-Attributs. Gültige Werte sind z. B. `UTF-8` oder `ISO-8859-1`. Die Angabe dieser Meta-Angabe ist ebenfalls keine Pflicht, sollte jedoch in keinem HTML-Dokument fehlen. Die HTML5-Spezifikation schreibt vor, dass sofern die Angabe vorhanden ist, die Kodierungs-Angabe innerhalb der ersten 1024 Bytes der Seite liegen muss. Der folgende Code legt die Zeichenkodierung UTF-8 fest:

```
1 | <meta charset="UTF-8" />
```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

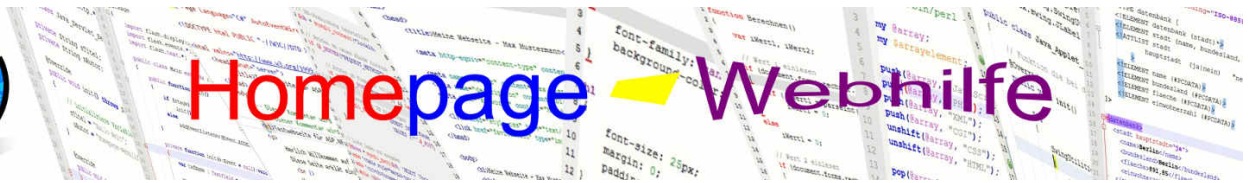
#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
 Krummstraße 9/3  
 73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
 E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Head-Bereich](#)

## Einbindung externer Daten

Um externe Daten bzw. Dateien einzubinden gibt es in HTML das `link`-Element. Auch dieses ist leer bzw. einteilig und wird im Head-Bereich platziert. Über das `rel`-Attribut wird angegeben, um was für eine Ressource oder vereinfacht gesagt, um was für eine Datei es sich handelt. Dieses Attribut ist erforderlich und muss bei Verwendung des `link`-Elements immer angegeben werden. Über das Attribut `href` (Hyper Reference) können wir die URL zu der Datei angeben. Bei einigen Ressourcen, wie z. B. Icons oder Stylesheets, sollte zusätzlich das `type`-Attribut verwendet werden, in welchem der MIME-Typ der jeweiligen Datei angegeben werden muss.

Die folgende Tabelle zeigt die verschiedenen Werte, welche im `rel`-Attribut eingesetzt werden können (links) und eine kurze Beschreibung dazu (rechts):

<b>icon</b>	Link zu einem Favicon (Icon, welches in der Browserzeile angezeigt wird)
<b>stylesheet</b>	Link zu einem Stylesheet (Dokument, welches Style-Angaben wie z. B. CSS enthalten)
<b>alternate</b>	Link zu einer alternativen Version des Dokuments (Druckversion, Backup-Version, aber auch Versionen in anderen Inhalten wie RSS-Feeds)
<b>prev</b>	Link zur vorherigen Seite (z. B. bei Kapiteln oder Tutorials, bei welchen es mehrere Seiten gibt, welche zusammengehören)
<b>next</b>	Link zur nächsten Seite (z. B. bei Kapiteln oder Tutorials, bei welchen es mehrere Seiten gibt, welche zusammengehören)
<b>search</b>	Link zur "Suchmaschine" der Website
<b>author</b>	Link zu einem Dokument über den Autor (z. B. "Über uns"-Seite)
<b>license</b>	Link zu einem Dokument über Copyright- und Lizenz-Informationen
<b>help</b>	Link zu einem Dokument, welches eine Hilfe enthält

Hier nun einige Code-Beispiele zu den oben aufgelisteten Link-Typen:

```

1 | <link rel="icon" href="/favicon.ico" type="image/x-icon" />
1 | <link rel="stylesheet" href="/layout.css" type="text/css" />
1 | <link rel="alternate" href="/feed.rss" type="application/rss+xml" />
1 | <link rel="prev" href="/HTML/einleitung.php" />
1 | <link rel="next" href="/HTML/text.php" />
1 | <link rel="search" href="/suche.html" />
1 | <link rel="author" href="/Ueber-Uns/" />
1 | <link rel="license" href="https://creativecommons.org/licenses/by-sa/3.0/" />
1 | <link rel="help" href="/Hilfe/" />
    
```

Eine Ausnahme ist die Einbindung externer Skripte (wie z. B. JavaScript-Dateien). Hier benötigen wir nicht das `link`-Element, sondern das `script`-Element. Dieses verfügt über das Attribut `type`, in welchem wir, wie beim `type`-Attribut des `link`-Elements, den MIME-Typ festlegen müssen (für JavaScript ist dies `text/javascript`). Mit Hilfe des Attributs `src` legen wir die URL zu der Datei fest. Das `script`-Element ist zweiteilig und kann sowohl im Head- als auch im Body-Bereich notiert werden.

```

1 | <script type="text/javascript" src="/JS/Google-Analytics.js"></script>
    
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

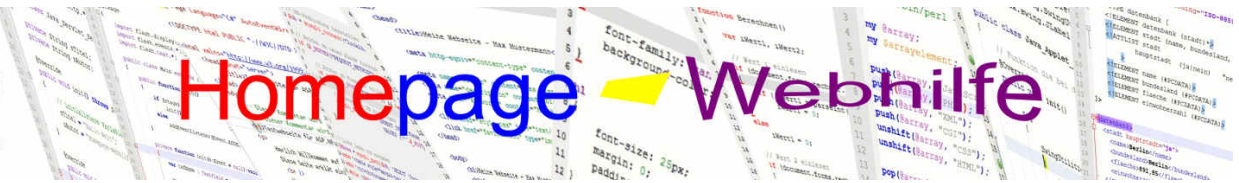
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Überschriften, Absätze und Texte](#)

## Überschriften, Absätze und Texte

Überschriften, Absätze und Texte sind neben dem Layout der zentrale Bestandteil einer HTML-Seite. Hierfür gibt es in HTML verschiedene Elemente, welche für die **Strukturierung** oder auch für die **Hervorhebung** von Textstellen verwendet werden.

Zuerst sollten wir jedoch den Unterschied zwischen den sogenannten Block- und Inline-Elementen klären. **Block-Elemente** benötigen immer die vollständige Breite des Bereichs (bei unseren ersten Beispielen bedeutet dies, dass die Elemente immer die komplette Breite des Fensters benötigen). Dies kann natürlich mittels CSS verändert werden. **Inline-Elemente** benötigen hingegen von der Breite immer nur so viel Platz wie dessen Inhalt. Block-Elemente werden dadurch immer dann eingesetzt, wenn es um Blöcke, Bereiche oder Absätze geht. Inline-Elemente werden hingegen oft für Schriftformatierungen oder ähnliches verwendet. Dadurch, dass Block-Elemente standardmäßig die komplette Breite des Fensters einnehmen, könnte man sagen, dass Block-Elemente einen Zeilenumbruch erzeugen. Dies ist bei den Inline-Elementen natürlich nicht der Fall. Alle Code-Beispiele stellen, sofern nicht anders angegeben, ab diesem Thema lediglich den Bereich zwischen den body-Tags dar.

### Inhalt dieser Seite:

1. Überschriften
2. Absätze
3. Zeilenumbrüche
4. Schriftauszeichnungen
5. Präformatierter Text
6. Akronyme
7. Zitate
8. Trennlinien

### Überschriften

Überschriften dienen dazu, Seiten und Abschnitte zu betiteln und zu unterteilen. Um Überschriften in HTML zu definieren, gibt es die zweiteiligen Elemente `h1`, `h2`, `h3`, `h4`, `h5` und `h6`. Zwischen Start- und End-Tag wird die jeweilige Überschrift notiert. Die Elemente verfügen alle standardmäßig über unterschiedliche Schriftgrößen, wovon `h1` die größte Schrift hat und `h6` die kleinste. Die Größe kann und sollte mittels CSS verändert werden.

```

1 <h1>Überschrift h1</h1>
2 <h2>Überschrift h2</h2>
3 <h3>Überschrift h3</h3>
4 <h4>Überschrift h4</h4>
5 <h5>Überschrift h5</h5>
6 <h6>Überschrift h6</h6>
    
```

Neben der Größe werden die verschiedenen Überschriften-Elemente hauptsächlich zur **Gliederung** in verschiedene Ebenen eingesetzt. Dabei stellt `h1` die oberste bzw. erste Ebene dar. Deshalb ist es gängig, dass `h1` lediglich einmal pro Seite verwendet wird, da jedes Buch ja auch nur einen Titel hat. Ein Buch und somit auch eine Website kann jedoch mehrere Untertitel (`h2`), "Unter-Untertitel" (`h3`), etc. haben. Hier ein Beispiel für eine solche Gliederung:

```

1 <h1>Programme</h1>
2 <h2>Browser</h2>
3 <h3>Microsoft Internet Explorer</h3>
4 <h3>Mozilla Firefox</h3>
5 <h3>Google Chrome</h3>
6 <h2>E-Mail-Clients</h2>
7 <h3>Microsoft Outlook</h3>
8 <h3>Mozilla Thunderbird</h3>
9 <h2>FTP-Clients</h2>
10 <h3>FileZilla</h3>
    
```

### Absätze

Absätze sind der zentrale Bestandteil, wenn es um Texte geht. Mit dem `p`-Element können Sie Absätze in HTML definieren. Das `p`-Element ist zweiteilig und der Text des Absatzes wird zwischen den Tags notiert. Im Beispiel wird Ihnen auffallen, dass zwischen den Absätzen mehr Abstand ist, als zwischen den einzelnen Zeilen. Auch dies kommt von den Voreinstellungen der Browser. Später können wir über CSS diese Abstände anpassen. Der Buchstabe `p` steht für *paragraph*.

```

1 <p>Absatz 1: ...</p>
2 <p>Absatz 2: ...</p>
3 <p>Absatz 3: ...</p>
    
```

### Zeilenumbrüche

Um eine Zeile **manuell umzubrechen**, gibt es das leere Element `br`. `br` (Abkürzung für *break*) kann innerhalb von Absätzen aber auch außerhalb verwendet werden. Um in einem Text eine Leerzeile zu erzeugen, benötigen wir zwei `br`-Elemente, da wir mit einem `br`-Element lediglich einen Umbruch erzeugen.

```

1 <p>
2   Hier steht Text ...
3   <br /><br />
4   Hier steht noch mehr Text ...
5   <br />
6   Und noch mehr Text ...
7 </p>
    
```

### Schriftauszeichnungen

Als Schriftauszeichnung wird die **Hervorhebung** (fachsprachlich auch Auszeichnung) einzelner Teile eines Textes bezeichnet. Schriftauszeichnungen sind nützlich, da dadurch wichtige Informationen (z. B. Schlagwörter) in den „Vordergrund“ gerückt werden können



<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Überschriften, Absätze und Texte](#)



(zumeist durch Fettdruck) und im Gegenzug dazu, jedoch auch unwichtige oder nebensächliche Informationen in den „Hintergrund“ gerückt werden können (z. B. durch kursiven Text). Die Verwendung von einfachen Schriftauszeichnungen ist **komplett ohne CSS** in reinem HTML realisierbar. Hervorhebungen von Textbestandteilen sollten zwar verwendet werden, jedoch sollte man damit auch **sparsam** umgehen. Ein Text, bei welchem die Hälfte oder mehr in fetter Schrift geschrieben ist, wirkt schnell aggressiv und unfreundlich.

Um auf Textbestandteile Schriftauszeichnungen mittels HTML anzuwenden, gibt es die zweiteiligen Elemente `b`, `i`, `u` und `s`. Zwischen dem jeweiligen Start- und End-Tag wird der zu hervorhebende Text notiert. Mit dem `b`-Element können Sie Texte **fett** hervorheben (*Fettdruck*) und mit dem `i`-Element können Sie **kursiven** Text erzeugen. Das `u`-Element versieht den Text mit einem **Unterstrich** (Linie unterhalb des Textes). Genutzt werden sollte das Element laut HTML5-Spezifikation, um (chinesische) Eigennamen oder falsch geschriebene Wörter hervorzuheben. Das `u`-Element wird daher selten verwendet, vor allem auch deshalb, da Benutzer einen Text mit Unterstrich zumeist als Link interpretieren. Mit dem Element `s` können wir einen Text **durchstreichen**. Die HTML5-Spezifikation beschreibt, dass dieses Element genutzt werden soll, um nicht mehr aktuelle Informationen (die aber nicht von der Seite entfernt werden) als „falsch“ bzw. „nicht mehr gültig“ zu markieren.

```

1 <b>fetter Text</b>
2 <br />
3 <i>kursiver Text</i>
4 <br />
5 <u>unterstrichener Text</u>
6 <br />
7 <s>durchgestrichener Text</s>

```



## Präformatierter Text

Als präformatierter Text wird Text bezeichnet, welcher auf eine bestimmte Art und Weise vorformatiert ist. Bei einem präformatierten Text wird der Text im Browser so angezeigt wie dieser im Editor eingegeben wurde.

So ist Ihnen bestimmt schon aufgefallen, dass wenn Sie in Ihrem Editor innerhalb eines Textes einen **Zeilenumbruch** notieren, dieser im Browser nicht erscheint. Die Anzeige ist also so, als hätten wir keinen Zeilenumbruch in der Datei notiert. Dies ist vom Regelwerk von HTML so vorgeschrieben, weshalb wir für die Erzeugung eines Zeilenumbruchs das `br`-Element benötigen. Aber warum ist das so? Stellen Sie sich vor, Sie schreiben einen längeren Text und notieren diesen in Ihrer HTML-Datei, die Zeilenumbrüche sollen aber vom Browser automatisch erstellt werden (z. B. auch abhängig von der Fenstergröße). In diesem Fall müssten Sie also Ihren ganzen Text in eine Zeile schreiben, was natürlich völlig unübersichtlich wäre.

Neben den Zeilenumbrüchen gibt es in HTML aber noch ein anderes Problem: Mehrere **Leerzeichen oder Tabs** werden immer nur als ein einziges Leerzeichen angezeigt. Auch dies ist von den HTML-Regeln festgelegt und hierfür gibt es ebenfalls einen Grund. Dabei beziehen wir uns nochmal auf unser vorheriges Beispiel. Nachdem wir nun eingesehen haben, dass es keinen Sinn macht, einen ganzen Text in eine Zeile zu schreiben, brechen wir unseren Text im Editor um. Unser `p`-Element, welches den Text enthält, ist jedoch nicht das Wurzelement und so sind das Element und der Text dementsprechend um einige Tabs oder Leerzeichen eingerückt. Dies machen wir ja, wie bereits am Anfang des Tutorials erwähnt, um einen strukturierten Code zu haben. Diese Einrückung des Textes hätte jetzt jedoch negative Auswirkung, wenn es die genannte Regelung in HTML nicht geben würde, sodass ungewollt mehrere Leerzeichen oder sogar Tabs angezeigt werden würden.

Doch was hat dies nun mit präformatiertem Text zu tun? Für einige Zwecke (z. B. **Logfiles oder Programmier-/Codeausschnitte**) ist es ungünstig, wenn der komplette Text überarbeitet werden müsste, sodass die dort vorhandenen Leerzeichen und Zeilenumbrüche auch tatsächlich so angezeigt werden. Dies gilt vor allem dann, wenn dieser „besondere Text“ später mittels einer serverseitigen Sprache (wie PHP) dynamisch erstellt oder aus einer Datei gelesen werden soll. Deshalb gibt es in HTML die Elemente `pre` und `code`. Innerhalb dieser werden alle Inhalte so angezeigt wie sie tatsächlich sind. `pre` ist ein Block-Element und wird für mehrzeiligen Text verwendet. `code` wird dagegen für einzeiligen Text verwendet, weshalb es ein Inline-Element ist und somit in den Textfluss mit eingebunden werden kann. Werden innerhalb des `code`-Elements Zeilenumbrüche notiert, werden diese nicht angezeigt (siehe Beispiel). Anders ist dies beim `pre`-Element.

```

1 <pre>Ein Code mit dem pre-Element
2   von HTML</pre>
3 <code>Ein Code mit dem code-Element
4   von HTML</code>

```



## Akronyme

Ein Akronym ist eine Form von **Abkürzung**, bei welcher die ersten Buchstaben der einzelnen Wörter zur Bildung der Abkürzung genutzt werden. Akronyme werden gerne in Texten genutzt, sodass diese nicht zu lange werden und auch einfacher zu lesen sind. Mit dem HTML-Element `abbr` können wir Akronyme markieren und die dazugehörige ausgeschriebene Schreibweise hinterlegen. Die Definition bzw. Bedeutung des Akronyms wird im `title`-Attribut des `abbr`-Elements angegeben. Browser zeigen die Bedeutung der Akronyme beim Darüberfahren mit der Maus an.

```

1 <p>Auf dieser Seite gibt es Tutorials zu <abbr title="HyperText Markup Language">HTML</abbr> und <abbr title="Cascading Style Sheet">CSS</abbr>.</p>

```



## Zitate

Um Zitate in HTML einzubetten, gibt es die Elemente `q`, `blockquote` und `cite`. Das `q`-Element wird für **einzeilige Zitate** verwendet, weshalb es ein Inline-Element ist und somit in den Textfluss eingebaut werden kann. Das Element `blockquote` ist ein Block-Element und wird somit für **komplexere oder mehrzeilige Zitate** verwendet. Innerhalb des Elements können Absätze (`p`-Element), aber auch Überschriften (`h1`-`h6`) und andere Elemente notiert werden. Dabei ist es geläufig, das Zitat nicht direkt im `blockquote`-Element, sondern in einem `p`-Element zu platzieren.

Sowohl das `q`-Element, als auch das `blockquote`-Element, verfügen über das `cite`-Attribut, in welchem die **Referenz in Form einer URL** zu dem Zitat angegeben werden kann. Dieses wird zwar nicht angezeigt, kann jedoch von Webcrawlern oder anderen computergesteuerten Systemen verwendet werden. Das `cite`-Element dient ähnlich wie das `cite`-Attribut zur Angabe der Referenz des Zitats oder allgemein gesagt des Werks (z. B. bei Verwendung mit einem Bild).

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



## Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Überschriften, Absätze und Texte](#)

Innerhalb des Elements kann ein Text aber auch eine URL (z. B. auch in Verbindung mit einem Link) notiert werden.

Brower zeigen ein Zitat mit dem `q`-Element normalerweise in doppelten Anführungszeichen an. `blockquote`-Elemente werden zumeist mit einem Abstand von 40px zur linken und rechten Seite angezeigt. Der Inhalt von `cite`-Elementen wird dagegen kursiv dargestellt.

Hier nun das Beispiel mit Verwendung eines `q`- und `cite`-Elements innerhalb eines kurzen Textes:

```
1 | <p>Das Zitat <q>Dies ist ein kleiner Schritt für einen Menschen, aber ein gewaltiger Sprung für die Menschheit.</q> kommt von  
  | <cite>Neil Armstrong, Apollo 11</cite>.</p>
```



Und hier das zweite Beispiel, welches das `blockquote`-Element nutzt:

```
1 | <blockquote>  
  | <p>Dies ist ein kleiner Schritt für einen Menschen, aber ein gewaltiger Sprung für die Menschheit.</p>  
  | <cite>- Neil Armstrong, Apollo 11</cite>  
4 | </blockquote>
```



## Trennlinien

Um eine Trennlinie (horizontal) per HTML zu erstellen, gibt es das leere Element `hr` (*horizontal rule*). Das `hr`-Element ist ein Blockelement und wird zur Trennung von verschiedenen Inhalten genutzt, welche nichts miteinander zu tun haben.

```
1 | <p>Hier steht Text ...</p>  
  | <hr />  
  | <p>Hier steht noch mehr Text ...</p>
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Links](#)

## Links

Links (eigentlich Hyperlinks) sind **Verweise**, um Sprünge auf andere Seiten oder innerhalb der Seite zu ermöglichen. Solche Verweise erstellen wir in HTML mit dem Element `a`. Innerhalb der Tags wird der anzuzeigende Text angezeigt. Der eigentliche Verweis wird im `href`-Attribut festgelegt.

```
1 | <a href="VERWEIS">TEXT</a>
```

Auch Links verfügen standardmäßig über eine besondere Darstellung. Dabei ist zwischen „normalen Links“, besuchten Links und aktiven Links zu unterscheiden. „Normale Links“ sind vom Benutzer bisher unbesucht und werden in blauer Farbe und mit einer Linie unter der Schrift angezeigt. Besuchte Links werden in lila und ebenfalls mit einer Linie angezeigt. Als aktive Links werden Links bezeichnet, die ausgewählt aber noch nicht aufgerufen sind. Wenn Sie also die linke Maustaste auf einem Link gedrückt halten, ist dieser aktiv. Sobald Sie die Taste loslassen, wird die aufzurufende Seite aufgerufen und der Link gilt ab sofort als besucht und nicht mehr als aktiv. Aktive Links werden unterstrichen und in roter Farbe angezeigt.

## URL-Links

URL-Links sind Verweise, welche auf **andere Seiten** verweisen. Diese können Bestandteil unserer Webseite sein oder einen Verweis auf eine andere Seite darstellen. Bei **internen Links**, also Links innerhalb unserer Website, können wir relative aber auch absolute Pfadangaben verwenden. **Externe Links** müssen neben dem Pfad- und Dateinamen (falls vorhanden) das Protokoll (z. B. http) und die Domain-Adresse (alternativ auch IP-Adresse) enthalten. Die Angabe des `target`-Attributs in Kombination mit dem Wert `_blank` sorgt dafür, dass die Seite in einem neuen Fenster oder Tab geöffnet wird. Das folgende Beispiel zeigt sowohl einen internen Link (Datei `maillinks.html`, welche sich im gleichen Verzeichnis befindet) als auch einen externen Link (Domain `www.google.de`):

```
1 | <a href="maillinks.html">Zum E-Mail-Link-Beispiel ...</a>
2 | <br />
3 | <a href="https://www.google.de/" target="_blank">Zu Google (Neues Fenster) ...</a>
```



## E-Mail-Links

E-Mail-Links sind Links, welche nicht auf eine Seite im World Wide Web verweisen, sondern auf eine **E-Mail-Adresse**. Browser öffnen beim Anklicken eines E-Mail-Links zumeist den auf dem Computer installierten E-Mail-Client (z. B. Mozilla Thunderbird, Microsoft Outlook). Die E-Mail-Adresse für einen E-Mail-Link wird ebenfalls im `href`-Attribut angegeben. Um dem Browser mitzuteilen, dass es sich um einen E-Mail-Link handelt, müssen wir jedoch vor der E-Mail-Adresse den Begriff `mailto`, gefolgt von einem Doppelpunkt, notieren. Dies sieht dann bspw. so aus:

```
1 | <a href="mailto:kontakt@homepage-webhilfe.de">Schreiben Sie uns eine E-Mail ...</a>
```



## Anker

Links zu Anker ermöglichen den Sprung innerhalb einer Seite. Hierfür ist ein **Anker** (dort wo wir hinspringen wollen) und ein **Anker-Link** (von dort wo wir springen wollen) notwendig.

Ein Anker kann seit HTML5 an jedem Element definiert werden. Hierfür müssen in dem Element, wo wir hinspringen wollen, das Attribut `id` festlegen. Als Wert für `id` ist ein frei definierbarer Name (**Anker-Name**) zulässig, der jedoch für diese Seite eindeutig sein muss und Leerzeichen und einige andere Sonderzeichen (z. B. #) nicht enthalten darf.

```
1 | <h1 id="oben">Webseiten-Titel</h1>
```

Ein Anker-Link wird ebenfalls im `href`-Attribut angegeben und enthält als Wert das `#`-Zeichen gefolgt von dem Anker-Namen.

```
1 | <a href="#oben">zum Seitenanfang</a>
```

Auch Anker-Links auf **externe Seiten** sind möglich. Hierfür notieren wir die URL gefolgt von dem `#`-Zeichen und dem Anker-Namen.

```
1 | <a href="text.php#navUmbruch">Zum Thema Zeilenumbrüche</a>
```

Der folgende Code zeigt ein etwas komplexeres Beispiel mit Anker und Anker-Links. Die Beispiel-Ansicht wurde noch um einige weitere Absätze erweitert.

```
1 | <h1 id="oben">Blindtext</h1>
2 | <p id="abs1">...</p>
3 | <p id="abs2">...</p>
4 | <p id="abs3">...</p>
5 | <br />
6 | <a href="#oben">Zurück zum Seitenanfang ...</a><br />
7 | <br />
8 | <a href="#abs1">Zu Absatz 1</a><br />
9 | <a href="#abs2">Zu Absatz 2</a><br />
10 | <a href="#abs3">Zu Absatz 3</a><br />
11 | <br />
12 | <a href="https://www.homepage-webhilfe.de/HTML/links.php#navAnker">Zum Thema ...</a>
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislöcher



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Bilder](#)

## Bilder

### Inhalt dieser Seite:

1. Bilder mit Verweisen

Bilder können in eine HTML-Seite mittels des `img`-Elements eingebunden werden. Das Element ist inhaltsleer und enthält einige Attribute, um Informationen über das einzubindende Bild zu spezifizieren.

Das `src`-Attribut gibt die **URL** zu der Bilddatei an. Hier kann der Link zu einem Bild vom Typ **JPG, PNG, BMP, SVG** und anderen Typen angegeben werden. Über die `width`- und `height`-Attribute können wir die **Größe** des Bildes festlegen. Werden beide Angaben weggelassen, so wird das Bild in Originalgröße angezeigt. Bei der Angabe des Attributs `width` oder `height` wird der fehlende Wert automatisch berechnet. Dabei wird das Seitenverhältnis beibehalten. Das Attribut `title` gibt den **Titel** des Bildes an, welcher beim Darüberfahren mit der Maus angezeigt wird. Über das `alt`-Attribut können wir den **Alternativtext** des Bildes festlegen. Dieser wird angezeigt, wenn das Bild nicht geladen werden kann (z. B. wenn das Bild nicht vorhanden ist). Das `alt`- und `title`-Attribut wird unter anderem auch von Suchmaschinen verarbeitet. Die Angabe der Attribute `src` und `alt` sind Pflicht. Alle anderen Attribute sind optional.

```
1 | 
```



## Bilder mit Verweisen

Wollen wir Bilder einbinden und dort einen **Link** platzieren, so können wir den `img`-Tag (Abkürzung für *image*) innerhalb eines `a`-Elements notieren. Doch dadurch können wir nur dem gesamten Bild einen einzigen Link zuweisen. Was machen wir nun, wenn das Bild **mehrere klickbare Bereiche** haben soll oder nur ein bestimmter Bereich als Link dienen soll? Für solche Zwecke gibt es das `map`-Element, mit welchem wir sogenannte *image-maps* definieren können. `map`-Elemente besitzen das `name`-Attribut, welches angegeben werden muss. Der dortige Name ist frei definierbar, muss jedoch eindeutig für die Seite sein. Das Attribut dient zur Verbindung der *image-map* mit dem `img`-Element. Innerhalb des `map`-Elements werden `area`-Elemente platziert, welche wir im nächsten Abschnitt vorstellen werden.

```
1 | <map name="ImageMap">
2 |
3 | </map>
```

Das leere `area`-Element wird zur Definition eines (klickbaren) Bereichs in einem Bild verwendet. `area`-Elemente werden innerhalb des `map`-Elements notiert und können unterschiedliche Formen haben. Die **Form** wird dabei mit dem `shape`-Attribut angegeben. Die **Koordinaten** der Form werden im `coords`-Attribut angegeben.

Um eine **rechteckige Form** zu definieren, brauchen wir den Wert `rect` im `shape`-Attribut. Im `coords`-Attribut wird zuerst die Ecke oben links angegeben und anschließend die Ecke unten links. Die X- und Y-Werte werden mit einem Komma voneinander getrennt.

```
1 | <area shape="rect" coords="x1,y1,x2,y2" />
```

Wollen wir einen **kreisförmigen Bereich** definieren, so benötigen wir den Wert `circle` im `shape`-Attribut. Das `coords`-Attribut enthält die Koordinaten des Kreis-Mittelpunkts gefolgt vom Kreisradius. Auch hier werden die Werte durch Kommas getrennt.

```
1 | <area shape="circle" coords="x,y,r" />
```

Die 3. und letzte Form ist das **Polygon** (Vieleck). Der Wert für das `shape`-Attribut ist hier `poly`. Das `coords`-Attribut enthält alle Punkte des Vielecks, wovon immer zuerst der X-Wert und anschließend der Y-Wert angegeben werden muss. Alle Werte müssen, wie bei den anderen Formen auch, durch Kommas getrennt werden.

```
1 | <area shape="rect" coords="x1,y1,x2,y2,x3,y3,...,xn,yn" />
```

Um unserem `area`-Element nun noch einen Link hinzuzufügen, gibt es, wie beim `a`-Element, das `href`-Attribut. Wird dieses angegeben, so muss auch das `alt`-Attribut angegeben werden, welches wie beim `img`-Element einen **Alternativtext** enthalten sollte. Auch die Angabe eines Titels mittels des `title`-Attributs ist möglich. Das `title`-Attribut steht übrigens auch bei allen anderen HTML-Elementen zur Verfügung.

Um unserem Bild die *image-map* zuzuweisen, benötigen wir im `img`-Element das `usemap`-Attribut. Hier wird das #-Zeichen, gefolgt von dem im `name`-Attribut des `name`-Elements notierten Namen, angegeben.

```
1 | 
2 | <map name="BannerMap">
3 |   <area shape="circle" coords="50,50,45" href="https://www.homepage-webhilfe.de/" alt="Logo" title="Startseite" />
4 |   <area shape="rect" coords="180,25,700,75" href="https://www.homepage-webhilfe.de/" alt="Schriftzug" title="Startseite" />
5 | </map>
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » HTML » Listen

## Listen

Listen ermöglichen eine einfache und strukturierte Art **Informationen oder Begriffe zu sammeln und aufzulisten**. HTML bietet 3 verschiedene Listen an, welche wir uns in diesem Thema genauer anschauen wollen.

### Inhalt dieser Seite:

1. Aufzählungslisten
2. Nummerierte Listen
3. Beschreibungslisten

## Aufzählungslisten

Eine Aufzählungsliste ist der einfachste, jedoch auch meist genutzte, Typ von Listen. Eine Liste wird in HTML mit dem `ul`-Element (*unordered list*) definiert. Um dieser Liste nun noch **Listenpunkte** hinzuzufügen, gibt es das ebenfalls zweiteilige Element `li`, welches innerhalb der `ul`-Tags notiert wird. Innerhalb der `li`-Tags wird der anzuzeigende Text angegeben. Eine Aufzählungsliste ist meist um 40px eingerückt (links) und enthält standardmäßig einen gefüllten Kreis für die einzelnen Punkte als **Aufzählungszeichen**. Diese Voreinstellungen können mittels CSS geändert werden.

```

1 <ul>
2   <li>Frühstück</li>
3   <li>Mittagessen</li>
4   <li>Kaffeetrinken</li>
5   <li>Abendessen</li>
6 </ul>
    
```



## Nummerierte Listen

Nummerierte Listen sind vom Grundaufbau mit Aufzählungslisten zu vergleichen, jedoch besitzen nummerierte Listen keinen Aufzählungspunkt, sondern **eine Nummer, einen Buchstaben oder ein römisches Zeichen**. Auf Grund dieses Unterschieds wird hier nicht das `ul`-Element, sondern das `ol`-Element (*ordered list*) verwendet. Die `li`-Elemente werden hier ebenfalls benötigt.

Standardmäßig werden nummerierte Listen mit Nummern als Aufzählungszeichen angezeigt. Um dies zu ändern, gibt es das Attribut `type`. Mögliche Werte sind: 1 (Nummern), A (Großbuchstaben), a (Kleinbuchstaben), I (große römische Zeichen) und i (kleine römische Zeichen). Um den **Startwert** der nummerierten Liste zu ändern, gibt es das Attribut `start`. Mit Hilfe des Attributs `reversed` und dem Wert `reversed` können wir die Nummerierung umdrehen (absteigend sortiert).

```

1 <ol>
2   <li>Frühstück</li>
3   <li>Mittagessen</li>
4   <li>Kaffeetrinken</li>
5   <li>Abendessen</li>
6 </ol>
7
8 <ol start="5">
9   <li>Frühstück</li>
10  <li>Mittagessen</li>
11  <li>Kaffeetrinken</li>
12  <li>Abendessen</li>
13 </ol>
14
15 <ol reversed="reversed">
16  <li>Frühstück</li>
17  <li>Mittagessen</li>
18  <li>Kaffeetrinken</li>
19  <li>Abendessen</li>
20 </ol>
21
22 <ol type="a">
23  <li>Frühstück</li>
24  <li>Mittagessen</li>
25  <li>Kaffeetrinken</li>
26  <li>Abendessen</li>
27 </ol>
    
```



## Beschreibungslisten

Beschreibungslisten sind der 3. Typ von Listen, welche vom Aufbau etwas anders sind als die zwei vorherigen. Beschreibungslisten werden über das `dl`-Element (*description list*) definiert. In diesem werden nun `dt`-Elemente (*description term*) und `dd`-Elemente (*description data*) untergeordnet. `dt`-Elemente stellen den zu **erläuternden Begriff** dar, wo hingegen der Inhalt im `dd`-Element die **Beschreibung des zu erläuternden Begriffs** enthält. `dd`-Elemente werden hierbei von Browsern mit 40px Abstand zur linken Seite angezeigt. `dt`- und `dd`-Elemente werden im Webbrowser untereinander angezeigt. Geeignet sind solche Listen z. B. für Glossare oder Formelsammlungen.

```

1 <dl>
2   <dt>HTML</dt>
3   <dd>HyperText Markup Language</dd>
4   <dt>CSS</dt>
5   <dd>Cascading Style Sheet</dd>
6   <dt>PHP</dt>
7   <dd>PHP Hypertext Processor</dd>
8   <dt>ASP</dt>
    
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Listen](#)

```
9 | <dd>Active Server Pages</dd>
10 | </dl>
```



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Tabellen](#)

## Tabellen

Tabellen werden in HTML mit dem `table`-Element definiert. Innerhalb der Tabelle können wir mit Hilfe von `tr`-Elementen (*table row*) **Tabellenzeilen** hinzufügen. Mit `td`-Elementen (*table data*) können wir **Tabellenzellen** erstellen. `td`-Elemente werden innerhalb von `tr`-Elementen platziert. Darauf ist zu achten, dass alle Spalten **gleich viele Zellen** enthalten. An Stelle des `td`-Elements kann auch das `th`-Element (*table header*) verwendet werden. Dieses stellt eine **Kopfzelle** dar, welche z. B. als Überschrift für eine Tabellenspalte genutzt werden kann. Kopfzellen werden, im Gegensatz zu normalen Zellen, fett gedruckt und dessen Text wird zentriert. Natürlich kann auch diese Voreinstellung mittels CSS geändert werden. Alle weiteren Formatierungen müssen (falls diese gewünscht sind) ebenfalls mit Hilfe CSS durchgeführt werden.

```

1 <table>
2   <tr>
3     <th>Name</th>
4     <th>Durchwahl</th>
5   </tr>
6   <tr>
7     <td>Hr. Pfeifer</td>
8     <td>15</td>
9   </tr>
10  <tr>
11    <td>Fr. Schneider</td>
12    <td>30</td>
13  </tr>
14  <tr>
15    <td>Hr. Müller</td>
16    <td>20</td>
17  </tr>
18  <tr>
19    <td>Fr. Koch</td>
20    <td>32</td>
21  </tr>
22 </table>
    
```

**Inhalt dieser Seite:**

1. Überschrift
2. Kopf, Inhalt und Fuß
3. Zellen zusammenführen



## Überschrift

Über das `caption`-Element können wir unserer Tabelle eine Überschrift geben. Dieses Element muss direkt nach dem öffnenden Tag des `table`-Elements folgen und darf pro Tabelle nur einmal vorkommen. Jedoch ist es auch erlaubt, das `caption`-Element wegzulassen. Zwischen den Tags des `caption`-Elements notieren wir die anzuzeigende Überschrift. Die Überschrift erstreckt sich dabei über die komplette Breite der Tabelle bzw. über alle Spalten.

```

1 <table>
2   <caption>Durchwahlliste</caption>
3   <tr>
4     <th>Name</th>
5     <th>Durchwahl</th>
6   </tr>
7   <tr>
8     <td>Hr. Pfeifer</td>
9     <td>15</td>
10  </tr>
11  <tr>
12    <td>Fr. Schneider</td>
13    <td>30</td>
14  </tr>
15  <tr>
16    <td>Hr. Müller</td>
17    <td>20</td>
18  </tr>
19  <tr>
20    <td>Fr. Koch</td>
21    <td>32</td>
22  </tr>
23 </table>
    
```



## Kopf, Inhalt und Fuß

Um eine Tabelle in die drei Hauptbestandteile (Kopf, Körper und Fuß) zu teilen, können wir die Elemente `thead`, `tbody` und `tfoot` verwenden. Alle Elemente müssen dabei am Anfang der Tabelle notiert werden (direkt nach dem `table`-Tag oder falls vorhanden dem `caption`-Element) und müssen in folgender Reihenfolge auftreten: `thead`, `tfoot`, `tbody`. Die reale Anzeihe der Tabelle erfolgt jedoch nach der Reihenfolge `thead`, `thead` und `tfoot`. Dabei kann natürlich das `thead`- oder `tfoot`-Element weggelassen werden, wenn dieses nicht benötigt wird. Die `tr`-Elemente werden bei Verwendung dieser Gliederung nicht mehr direkt innerhalb der `table`-Tags platziert, sondern zwischen den `thead`-, `tbody` und `tfoot`-Tags.

```

1 <table>
2   <thead>
3     <tr>
    
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » HTML » Tabellen

```

4      <th>Name</th>
5      <th>Durchwahl</th>
6    </tr>
7  </thead>
8  <tfoot>
9    <tr>
10     <td>Amt</td>
11     <td>0</td>
12   </tr>
13 </tfoot>
14 <tbody>
15 <tr>
16   <td>Hr. Pfeifer</td>
17   <td>15</td>
18 </tr>
19 <tr>
20   <td>Fr. Schneider</td>
21   <td>30</td>
22 </tr>
23 <tr>
24   <td>Hr. Müller</td>
25   <td>20</td>
26 </tr>
27 <tr>
28   <td>Fr. Koch</td>
29   <td>32</td>
30 </tr>
31 </tbody>
32 </table>
    
```



## Zellen zusammenführen

Wie in Programmen wie z. B. Microsoft Excel ist es auch mittels der HTML-Tabellen möglich, **Zellen vertikal oder horizontal zu verbinden**. Hierfür müssen wir in der Zelle (td- oder th-Element), welche mit anderen verbunden werden soll, das Attribut `rowspan` oder `colspan` notieren. `rowspan` (*row = Zeile*) gibt an, über wie viele Zeilen sich die Zelle erstrecken soll. `colspan` (*col = column = Spalte*) gibt hingegen an, über wie viele Spalten sich die Zelle erstreckt. Das Wort *span* in `colspan` und `rowspan` kommt aus der englischen Sprache und heißt so viel wie „spannen“. Man könnte also sagen, mit Hilfe der zwei Attribute wird eine Zelle überspannt. Der Standardwert von `colspan` und `rowspan` ist 1. Die Angabe der Attribute mit dem Wert 1 ist nicht notwendig und sorgt nur für einen längeren HTML-Code.



**Wichtig** ist, dass wenn wir z. B. eine Tabelle mit 3 Spalten haben und sich die 1. Zelle über 2 Zellen erstrecken soll (`colspan="2"`), dass sich dann lediglich zwei `td`-Elemente innerhalb des `tr`-Elements befinden dürfen. Das Beispiel und die nebenstehende Grafik wird die Erklärung des Textes verdeutlichen.

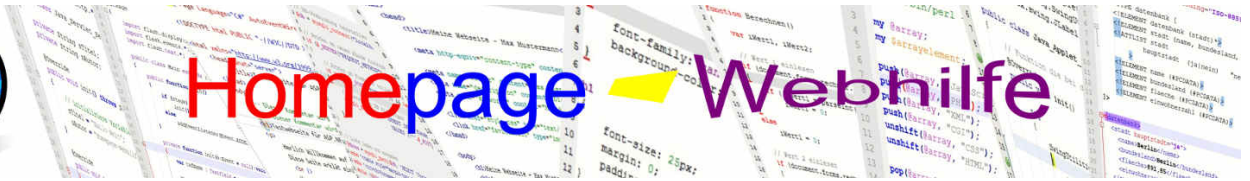
```

1 <table>
2   <tr>
3     <th>Name</th>
4     <th>Durchwahl</th>
5   </tr>
6   <tr>
7     <td>Hr. Pfeifer</td>
8     <td>15</td>
9   </tr>
10  <tr>
11    <td>Fr. Schneider</td>
12    <td>30</td>
13  </tr>
14  <tr>
15    <td rowspan="2">Hr. Müller</td>
16    <td>20</td>
17  </tr>
18  <tr>
19    <td>32</td>
20  </tr>
21  <tr>
22    <td colspan="2"><i>Keine 0 vorwählen!</i></td>
23  </tr>
24 </table>
    
```



Früher wurden Tabellen unter anderem auch dazu verwendet, Webseiten zu designen. Diese Technik gilt als veraltet und sollte nicht mehr eingesetzt werden, da das Design einer Seite vollständig über CSS erfolgen soll.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » HTML » Strukturierung

## Strukturierung

Um Informationen wie Bilder und Texte zu strukturieren und zu gruppieren, gibt es in HTML einige Elemente, welche wir Ihnen hier vorstellen wollen. Dabei können wir grundsätzlich zwischen Gruppierung und Strukturierung unterscheiden.

### Inhalt dieser Seite:

1. Gruppierung
2. Seitenstrukturierung

## Gruppierung

Um Inhalte zu gruppieren, gibt es das `fieldset`-Element (*fieldset = Feldgruppe*). Diesem sollte zum einen das `legend`-Element (Überschrift des `fieldset`-Elements) untergeordnet werden (welches zumeist am Anfang notiert wird) und weitere Elemente wie z. B. `p`, `table` oder auch Text enthalten. Der durch das `fieldset`-Element entstandene Block enthält standardmäßig automatisch einen Rahmen. Der Inhalt des `legend`-Elements wird dabei im Rahmen oben links angezeigt, wobei der Rahmen für die Breite des Inhalts unterbrochen wird. Das `fieldset`-Element ist eigentlich dazu gedacht, ein [Formular](#) visuell abzusetzen und dessen Bestandteile zu gruppieren. Die Benutzung von anderen Inhalten in `fieldset`-Elementen ist jedoch ebenfalls möglich.

```

1 <fieldset>
2   <legend>Titel</legend>
3   <p>Text ...</p>
4 </fieldset>
    
```

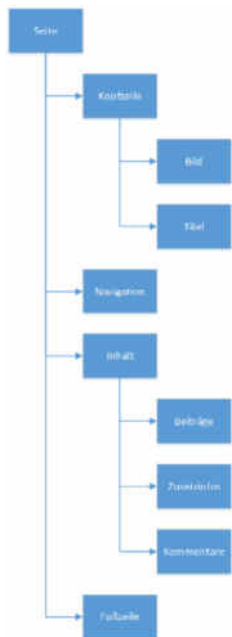


## Seitenstrukturierung

Mit der Einführung von HTML5 wurden auch einige neue Elemente zur Strukturierung einer Seite in den Element-Vorrat von HTML hinzugefügt. Der Grund für die Einführung war einfach: Bisher mussten Bestandteile einer Website immer mittels den `div`-Elementen erstellt werden. Um diese später mittels CSS zu erkennen, wurde den Elementen das Attribut `id` mit einem passenden frei definierbaren Namen vergeben (darauf gehen wir im Thema CSS genauer ein). Ein kurzes Beispiel hierzu:

```

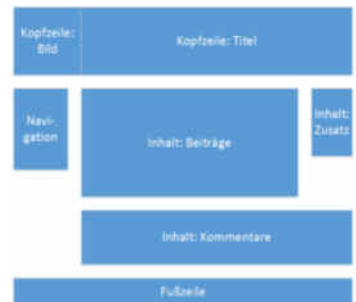
1 <div id="kopfzeile">
2
3 </div>
    
```



`div`-Elemente benötigen immer den kompletten Platz, da es Block-Elemente sind. `span`-Elemente sind hingegen Inline-Elemente und werden für Formatierungen verwendet. Beide Elemente gelten in ihrem jeweiligen Anzeige-Charakteristikum als **Universal-Elemente**. Mit den neuen Strukturierungselementen von HTML soll die Anzahl der benötigten `div`-Elemente sinken. Dadurch ist auch der HTML-Code **leichter zu lesen** und die einzelnen Bestandteile sind **leichter zu finden** (z. B. wenn man mit fremdem HTML-Code arbeiten muss).

Das Hauptproblem der neuen Elemente ist jedoch, dass es **keine klare Vorgabe** gibt, wie die Elemente zu verwenden sind, weshalb alle Beispiel-Codes in Bezug auf die Strukturierungselemente etwas anders sind. Wir haben Ihnen folgende Tabelle aufgestellt, welche die HTML-Strukturierungselemente und ihre Bedeutung enthält:

<b>article</b>	Bereich für einen Artikel eines Dokuments (ideal für Beiträge bei einem Blog)
<b>aside</b>	Bereich für einen nebenstehenden Abschnitt
<b>footer</b>	Fußbereich (z. B. für die Fußzeile)
<b>header</b>	Kopfbereich (z. B. für die Kopfzeile)
<b>main</b>	Bereich für den Hauptbestandteil der Website (zumeist Inhaltsbereich)
<b>nav</b>	Bereich für eine Navigation
<b>section</b>	Bereich für einen Abschnitt eines Dokuments



Der folgende Code zeigt nun eine von vielen Möglichkeiten, wie die Elemente `header`, `nav`, `main`, `aside`, `article`, `section` und `footer` benutzt werden können. Der Aufbau des Dokuments ist dabei in der Grafik links zu sehen. Ein mögliches resultierendes Ergebnis durch eine Formatierung mit CSS ist auf der rechten Seite zu sehen.

```

1 <header>
2   
3   <h1>Homepage-Webhilfe</h1>
4 </header>
5
6 <nav>
7   <ul>
8     <li><a href="https://www.homepage-webhilfe.de/">Startseite</a></li>
9     <li><a href="https://www.homepage-webhilfe.de/HTML/">HTML</a></li>
10    <li><a href="https://www.homepage-webhilfe.de/CSS/">CSS</a></li>
11    <li><a href="https://www.homepage-webhilfe.de/PHP/">PHP</a></li>
12  </ul>
13 </nav>
14
    
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » HTML » Strukturierung

```

15 <main>
16   <p>...</p>
17
18   <aside>
19     <h3>Wussten Sie schon?</h3>
20     <p>...</p>
21   </aside>
22
23   <article>
24     <h2>Beitrag 1</h2>
25     <p>...</p>
26   </article>
27
28   <article>
29     <h2>Beitrag 2</h2>
30     <p>...</p>
31   </article>
32
33   <article>
34     <h2>Beitrag 3</h2>
35     <p>...</p>
36   </article>
37
38   <section>
39     <h2>Kommentare</h2>
40     <p>...</p>
41     <p>...</p>
42     <p>...</p>
43   </section>
44 </main>
45
46 <footer>
47   <i>Copyright 2016 by Homepage-Webhilfe</i>
48 </footer>

```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Formulare](#)

## Formulare

Formulare dienen dazu, **Eingabefelder** für Besucher zur Verfügung zu stellen und die darin enthaltenen Informationen weiterzuverarbeiten bzw. an eine **andere Datei zu schicken**. Formulare werden in HTML mit dem `form`-Element definiert.

Innerhalb des `form`-Elements können wir einige Attribute festlegen, wovon jedoch keines davon angegeben werden muss. Mit dem Attribut `action` können wir festlegen, an welche Datei (Angabe als URL) die sogenannten Formulare Daten gesendet werden sollen. Wird das Attribut nicht angegeben, so werden die Formulare Daten an die aktuelle Datei geschickt (also an die Datei, welche das Formular enthält). Das Attribut `method` legt fest, über welche **HTTP-Methode** die Daten versendet werden sollen. Gültige Werte sind `get` und `post`, wovon `get` die Voreinstellung ist. Bei der GET-Methode werden die Formulare Daten an die URL angehängt (`?name=wert&name2=wert2`). Bei der POST-Methode hingegen befinden sich die Formulare Daten innerhalb des HTTP-Protokolls und somit außerhalb der **URL**. Dadurch sind die Daten sozusagen nicht sichtbar (jedoch nicht verschlüsselt!). Werden in einem Formular kritische Formulare Daten, wie z. B. Passwörter verwendet, sollte das Formular nicht mit der GET-Methode übertragen werden. Des Weiteren eignet sich die GET-Methode nur für kleinere Datenmengen, denn die Länge einer URL ist begrenzt. Das Attribut `enctype` legt fest, wie die Daten, also mit welcher Kodierung die Daten, im HTTP-Protokoll versendet werden sollen. Die Angabe ist für uns jedoch vorerst nicht relevant.

Der folgende Code zeigt eine einfache Definition eines Formulars:

```
1 | <form action="auswertung.php" method="post">
2 |
3 | </form>
```

### Inhalt dieser Seite:

1. Text-Eingabe
2. Numerische Eingabe
3. Kontrollkästchen
4. Auswahlgruppen
5. Auswahllisten
6. Buttons
7. Dateiuploads
8. Versteckte Felder

## Text-Eingabe

Nachdem wir nun unser Formular definiert haben, brauchen wir natürlich noch Felder. Die Verarbeitung des Formulars erfolgt normalerweise über eine **serverseitige Programmiersprache** wie PHP oder ASP.NET. Das heißt also, dass wir am Ende dieses Themas bereits Formulare erstellen können, jedoch noch nicht wissen, wie wir diese verarbeiten. Sollten Sie also Formulare nicht benötigen, können Sie dieses Thema auch vorerst überspringen. Falls Sie bereits eine serverseitige Programmiersprache erlernt haben, dann können Sie natürlich auch jetzt schon die Daten des Formulars verarbeiten.

Doch nun genug Theorie: Nun wollen wir das erste Feld in unser Formular integrieren. Hierfür benötigen wir das `input`-Element. Das `input`-Element ist inhaltsleer und kann über verschiedene Attribute angepasst werden. Um zu erkennen, um was für ein **Eingabefeld** es sich handelt, gibt es das `type`-Attribut. Der dazugehörige Wert für ein Eingabefeld für „reinen Text“ ist `text`. Handelt es sich um ein Textfeld, bei welchem die Daten „verschlüsselt“ angezeigt werden sollen (zumeist bei **Passwörtern**), so ist der Wert `text` durch `password` zu ersetzen.

Um Formularfelder später weiter verarbeiten zu können, brauchen wir das Attribut `name`. Über den dort vergebenen Namen können wir das **Feld identifizieren**. Enthält ein `input`-Element kein `name`-Attribut, so wird dessen Wert beim Versand (dazu später mehr) nicht übermittelt. Das Attribut `value` enthält den Wert des Eingabefelds (vordefinierter Wert) und kann ohne Weiteres weggelassen werden. Hier nun das erste Beispiel:

```
1 | <input type="text" name="Vorname" value="Peter" />
```

Für Textfelder können wir mit Hilfe des `maxlength`-Attributs angeben, wie viele Zeichen maximal eingegeben werden dürfen. Fehlt diese Angabe, so ist die Länge „unbegrenzt“.

```
1 | <input type="text" name="Vorname" value="Peter" maxlength="40" />
```

Weitere Attribute, welche für alle Felder eingesetzt werden können, sind u. a. `readonly`, `disabled` und `required`. Durch das Attribut `readonly` und den gleichnamigen Wert, können wir dem Feld die Eigenschaft zuweisen, dass das Feld **nicht editiert** werden kann. Das Attribut `disabled` und der Wert `disabled` **deaktiviert das Eingabefeld** und hat im Großen und Ganzen die gleiche Wirkung wie das `readonly`-Attribut. Der Unterschied zwischen dem `disabled`- und `readonly`-Attribut ist, dass Felder mit dem `readonly`-Attribut noch markiert werden können und auch weiterhin übermittelt werden. Enthält das Feld das `disabled`-Attribut, so kann das Feld weder markiert werden noch wird dessen Wert übermittelt. Das ebenfalls optionale `required`-Attribut mit dem dazugehörigen Wert `required` markiert das Feld als **Pflichtfeld**. Dadurch wird vor dem Absenden des Formulars geprüft, ob das Feld leer ist. Ist dies der Fall, so erscheint eine Fehlermeldung (oft als Tooltip), das Feld wird eingerahmt und der Versandvorgang wird abgebrochen. Hier nun ein komplexeres Beispiel:

```
1 | <form>
2 |   Vorname: <input type="text" name="Vorname" value="Peter" disabled="disabled" />
3 |   <br />
4 |   Nachname: <input type="text" name="Nachname" value="Meyer" readonly="readonly" />
5 |   <br />
6 |   Benutzername: <input type="text" name="Benutzername" maxlength="30" required="required" />
7 |   <br />
8 |   Passwort: <input type="password" name="Passwort" maxlength="30" />
9 | </form>
```

Wenn Sie sich das Beispiel angeschaut haben, wird Ihnen bestimmt aufgefallen sein, dass die im Beispiel vorhandenen Eingabefelder **einzeilig** sind. Doch es gibt auch die Möglichkeit, ein **mehrzeiliges Textfeld** zu erstellen: Hierfür benötigen wir jedoch nicht das `input`-Element, sondern das `textarea`-Element. Das `textarea`-Element ist zweiteilig. Zwischen den Tags kann, sofern vorhanden, ein vordefinierter Text notiert werden. Das `value`-Attribut gibt es hier also nicht. Die Attribute `name`, `maxlength`, `readonly`, `disabled` und `required` gibt es, wie beim `input`-Element, hier ebenfalls. Die Bedeutung der Attribute ist dabei die gleiche.

```
1 | <form>
2 |   Nachricht:
3 |   <textarea name="Nachricht" maxlength="200">Hier steht die Nachricht ...</textarea>
4 | </form>
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Formulare](#)

## Numerische Eingabe

Numerische Eingabefelder werden wie einzeilige Textfelder mit dem `input`-Element definiert. Als Typ muss hier `number` angegeben werden. Numerische Eingabefelder verfügen zusätzlich am Ende des Feldes über **Buttons zum Verringern und Vergrößern** des Wertes. Die Eingabe der Werte für den Benutzer erfolgt dabei dezimal. Über die Attribute `min` und `max` kann zusätzlich der Bereich der erlaubten Zahl eingeschränkt werden. Beide Attribute sind optional, wovon es auch erlaubt ist, nur eines davon anzugeben.

```
1 <form>
2   Geburtsjahr: <input type="number" name="Geburtsjahr" value="2000" min="1900" max="2100" />
3 </form>
```



## Kontrollkästchen

Kontrollkästchen sind die kleinen quadratischen Boxen, die durch Anklicken mit einem **Häkchen** markiert werden. Ein weiteres Anklicken führt dazu, dass das Häkchen wieder entfernt wird. Als Typ im `type`-Attribut muss hierzu `checkbox` angegeben werden. Der im `value`-Attribut angegebene Wert wird dabei nur übertragen, wenn das Häkchen gesetzt ist. Ist das Kontrollkästchen (*englisch checkbox*) nicht ausgewählt, so wird auch das Feld nicht übertragen. Über das Attribut `checked` können wir das Kästchen bereits vorauswählen. Da Kontrollkästchen über **keine sichtbare Beschriftung** verfügen, wird oft hinter dem `input`-Element ein Text notiert. Dadurch kann der Endbenutzer das Kästchen zuordnen.

```
1 <form>
2   <input type="checkbox" name="Newsletter" value="ja" checked="checked" /> Ja, ich möchte den Newsletter erhalten.
3 </form>
```



## Auswahlgruppen

Auswahlgruppen (*englisch radio buttons*) sind eine Gruppe von runden Feldern, welche es ermöglichen, **eine von mehreren Möglichkeiten** auszuwählen. Hierfür muss im `input`-Element das `type`-Attribut auf `radio` gesetzt werden. Um mehrere sogenannte **Optionsfelder** (oft auch einfach nur Radiobuttons) zusammenzufassen, müssen die zusammengehörende Felder über den gleichen Wert im `name`-Attribut verfügen. Der im `value`-Attribut notierte Wert unterscheidet sich hingegen und dient ebenfalls nur zum Versand und nicht zu Anzeige. Wie Kontrollkästchen können auch Optionsfelder mit dem `checked`-Attribut vorselektiert werden. Hierzu folgendes typisches Beispiel, welches Sie in ähnlicher Form auf vielen Websites finden werden:

```
1 <form>
2   Anrede: <input type="radio" name="Anrede" value="Herr" /> Herr <input type="radio" name="Anrede" value="Frau"
3   checked="checked" /> Frau
4 </form>
```



## Auswahllisten

Auswahllisten ermöglichen, ähnlich wie Auswahlgruppen, die Auswahl von ein oder auch mehreren Werten aus einer **Liste**. Bei Auswahllisten benötigen wir nicht das `input`-Element, sondern die Elemente `select` und `option`.

`select` definiert dabei die Liste. Hier kann wie beim `select`-Element, das Attribut `name` oder auch `disabled` ebenfalls notiert werden. Das `option`-Element definiert einen Eintrag der Auswahlliste. Das Element ist zweiteilig und wird innerhalb der `select`-Tags notiert. Die Attribute `value` und `disabled` stehen uns dabei auch im `option`-Element zur Verfügung. Der Wert des `value`-Attributs entspricht nicht dem anzuzeigenden Wert, sondern lediglich dem zu versendenden Wert. Der anzuzeigende Wert wird zwischen den Tags des `option`-Elements angegeben. Wird das `value`-Attribut weggelassen, so wird sowohl zur Anzeige als auch zur Übertragung der Wert zwischen den `option`-Tags verwendet. Die Vorselektierung eines `option`-Elements erfolgt nicht mit dem `checked`-Attribut, sondern mit dem `selected`-Attribut.

Standardmäßig ist in einer Auswahlliste nur die Auswahl eines Wertes möglich. Möchten Sie jedoch dem Benutzer ermöglichen, mehrere Werte auszuwählen zu können (**Mehrfachauswahl**), so können Sie dies mit Hilfe des Attributs `multiple` und dem Wert `multiple` innerhalb des `select`-Tags erlauben. Das Attribut `size` kann dazu genutzt werden, die Anzahl der anzuzeigenden Werte für die Liste festzulegen. Die Standardeinstellung ist 1. Verwendet wird das `size`-Attribut meist nur bei Auswahllisten, bei welchen eine Mehrfachauswahl möglich ist.

```
1 <form>
2   E-Book:
3   <select name="EBook">
4     <option value="Einf">Einführung</option>
5     <option value="ProgClnt" selected="selected">Programmierung Client</option>
6     <option value="ProgServ">Programmierung Server</option>
7     <option value="Fort">Fortgeschrittenes</option>
8   </select>
9   <br />
10  Vorkenntnisse:
11  <select name="Prog" size="5" multiple="multiple">
12    <option>HTML</option>
13    <option>CSS</option>
14    <option>JavaScript</option>
15    <option>PHP</option>
16    <option>Perl</option>
17  </select>
18 </form>
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Formulare](#)

## Buttons

Buttons (oder auch Schaltflächen genannt) erlauben dem Benutzer, eine bestimmte **Aktion auszuführen**. Hierfür besitzen Formulare 2 fest definierte Buttons: der Zurücksetzen-Button und der Sende-Button.

Der **Zurücksetzen-Button** setzt, wie der Name schon sagt, alle Werte im Formular wieder zurück. Hierbei wird nicht der im HTML-Code definierte Wert des `value`-Attributs wiederhergestellt, sondern das Eingabefeld wird lediglich „geleert“. Um einen solchen Button zu definieren, benötigen wir das `input`-Element mit dem Typ `reset`.

Der **Sende-Button** führt den Versand des Formulars aus. Dabei wird das Formular zuerst geprüft (z. B. ob Felder ausgefüllt wurden, welche mit dem `required`-Attribut versehen sind) und anschließend die Daten an die im `action`-Attribut angegebene Datei (oder falls nicht angegeben, an die gleiche Datei) gesendet. Beim Drücken des Sende-Buttons wird hierfür die neue Seite geladen bzw. die gleiche Seite erneut geladen. Der Sende-Button wird ebenfalls über das `input`-Element definiert. Als Wert im `type`-Attribut muss hier `submit` gesetzt werden. Bei beiden Buttons entspricht der im `value`-Attribut festgelegte Wert dem anzuzeigenden Text innerhalb des Buttons.

Weitere **universelle Buttons** können über den Wert `button` im Type-Attribut spezifiziert werden. Diese Buttons haben ohne eine Skriptsprache wie JavaScript keine Wirkung. Das Klicken eines solchen Buttons, ohne ein JavaScript-Ereignis zu hinterlegen, führt weder zum Versand des Formulars, noch zum erneuten laden der Seite, noch zum Zurücksetzen irgendwelcher Werte.

Wollen wir innerhalb eines Buttons nicht nur einen einfachen Text anzeigen, sondern z. B. ein Bild, so können wir das `input`-Element durch ein `button`-Element ersetzen. Auch hier gibt es das `type`-, `name`- und `disabled`-Attribut. Das `value`-Attribut ist beim `button`-Element ebenfalls vorhanden, wird jedoch nicht zur Anzeige verwendet, sondern legt lediglich den zu versendenden Wert des Buttons fest. Das `button`-Element ist zweiteilig, wodurch es möglich ist, zwischen den Tags einen HTML-Code (z. B. ein `img`-Tag für ein Bild) zu notieren.

**Übrigens:** Zumeist ist es nicht notwendig, den Inhalt (Text) eines Buttons zu versenden. Deshalb wird bei Buttons oft das `name`-Attribut weggelassen, denn dadurch wird dessen Wert nicht übermittelt.

```

1 <form>
2   Nachname: <input type="text" name="Nachname" />
3   <br />
4   <input type="reset" value="Zurücksetzen" />
5   <input type="submit" value="Senden" />
6   <br />
7   <input type="button" value="Kaufen" />
8   <input type="button" value="Ausleihen" />
9   <br />
10  <button type="button" title="Startseite aufrufen">
11    
12  </button>
13 </form>

```



## Dateiuploads

Um den Upload einer Datei zu ermöglichen, müssen wir ein `input`-Element mit dem Typ `file` notieren. Möchten Sie den Typ der auswählbaren Dateien einschränken, so können Sie im Attribut `accept` entweder den MIME-Typ oder die Dateiendung mit vorangestelltem Punkt des zu erlaubenden Dateityps angeben.

Bei Dateiuploads muss das Formular zwingend per **POST-Methode** übertragen werden. Zudem muss im `form`-Tag das Attribut `enctype` mit dem Wert `multipart/form-data` angegeben werden. Die Standardeinstellung des `enctype`-Attributs ist `application/x-www-form-urlencoded`. Dabei werden alle Namen und Werte im URL-Format versendet (bsp. Vorname=Peter&Nachname=Meyer). Leerzeichen werden in Pluszeichen umgewandelt und Sonderzeichen werden in Hex-Werte umgewandelt (+ » %2B). Wird der Wert `multipart/form-data` verwendet, so werden die Datei und die restlichen Felder im HTTP-Protokoll durch sogenannte Boundaries getrennt. Der genaue Aufbau des Protokolls würde dieses Thema jedoch sprengen. Das `enctype`-Attribut ist nur für Formulare mit der POST-Methode relevant.

```

1 <form method="post" enctype="multipart/form-data">
2   <input type="file" name="Datei" />
3   <br />
4   <input type="submit" value="Senden" />
5 </form>

```



## Versteckte Felder

Versteckte Eingabefelder werden in HTML mit dem `input`-Element und dem Typ `hidden` notiert. Solche Eingabefelder dienen dabei vor allem als **Hilfsmittel**, um einen Formular-Versand mit, für den Benutzer nicht sichtbaren, Informationen zu erweitern. Oft werden solche versteckte Eingabefelder auch in Zusammenhang mit JavaScript-Funktionen verwendet.

```

1 <form>
2   Vorname: <input type="hidden" name="Vorname" value="Kai" />
3   <br />
4   Nachname: <input type="text" name="Nachname" />
5   <br />
6   <input type="submit" value="Senden" />
7 </form>

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Medien-Einbindung](#)

## Medien-Einbindung

HTML bietet neben der Einbindung von Bildern mittels des `img`-Tags auch die Möglichkeit, andere Medien-Dateien wie z. B. Audio-, Video- oder Flash-Dateien einzubinden. Die dafür notwendigen Elemente und Attribute stellen wir Ihnen hier vor.

### Inhalt dieser Seite:

1. Fremdseiten
2. Audio
3. Video
4. Flash

### Fremdseiten

Bevor wir uns mit der Einbindung von den „richtigen“ Medien-Dateien beschäftigen, wollen wir Ihnen noch die sogenannten **iFrames** vorstellen. iFrames ermöglichen es, andere Dateien (zumeist andere HTML-Dokumente) innerhalb eines Kästchens anzuzeigen und somit in die aktuelle Seite zu integrieren. Die einzubindende Datei kann dabei auf der gleichen Website oder auch auf einer externen Website (Fremdseite) liegen. iFrames werden durch das zweiteilige Element `iframe` notiert. Das Attribut `src` gibt die URL zu der anzuzeigenden Datei an. Der Text zwischen den `iframe`-Tags legt einen Text fest, welcher angezeigt werden soll, wenn der Browser keine iFrames unterstützt. Alle modernen Browser unterstützen jedoch iFrames. Die Höhe und Breite eines iFrames kann mit Hilfe der Attribute `height` und `width` festgelegt werden. Die Werte im `height`- und `width`-Attribut müssen in Pixel (ohne Einheit) angegeben werden.

```
1 | <iframe src="absaetze.html" width="600" height="400">Ihr Browser unterstützt kein iFrames.</iframe>
```

### Audio

Um Audio-Dateien in eine Seite mittels HTML einzubinden, gibt es das `audio`-Element. Innerhalb des `audio`-Elements werden sowohl `source`-Elemente sowie optional ein Text (wird angezeigt, falls der Browser die Einbindung von Audio-Dateien nicht unterstützt) notiert. Das `audio`-Element verfügt über einige Attribute zum Festlegen von Einstellungen. Deren Werte entsprechen dabei immer dem Attribut-Namen bzw. könnten für gültiges HTML5 auch weggelassen werden. Wird das `autoplay`-Attribut notiert, so wird die Musik **automatisch abgespielt**. Wollen wir dem Benutzer eine **Steuerung** (Pause, Spulen und Lautstärke) für die Musik anzeigen, so müssen wir das `controls`-Attribut notieren. Das `loop`-Attribut gibt an, ob die Musik in einer **Endlosschleife** wiedergegeben werden soll. Ein weiteres Attribut ist `muted`, wird dieses angegeben, so ist der Ton stummgeschaltet. Wird die Steuerung angezeigt, so kann die Stummschaltung auch wieder durch den Benutzer deaktiviert werden.

Das `source`-Element gibt die Quelle der abzuspielenden Datei an. `source`-Elemente sind einteilig und verfügen über die Attribute `src` und `type`. Das `src`-Attribut gibt die URL der Datei an. Mit dem `type`-Attribut geben wir den MIME-Typ der Datei an. Die folgende Tabelle zeigt die aktuell möglichen Dateitypen sowie deren MIME-Typ und Browser-Unterstützung:

Dateityp	MIME-Typ	Firefox	Chrome	Opera	Safari	IE / Edge
MP3	audio/mpeg	ja	ja	ja (bei Opera Mini, nein)	ja	ja (ab IE9)
WAV	audio/wav	ja	ja	ja (bei Opera Mini, nein)	ja	ja (ab Edge 13)
OGG	audio/ogg	ja	ja	ja (bei Opera Mini, nein)	nein	nein

Aus aktuellem Stand ist daher ersichtlich, dass die Verwendung von MP3-Dateien die sicherste Variante zur Einbindung von Audio-Dateien ist. Trotzdem haben wir uns im folgenden Beispiel dazu entschieden, alle unterstützten Dateiformate einzubinden. Dies ist möglich, indem wir die Datei mehrmals auf unserem Server in unterschiedlichen Formaten ablegen und innerhalb des `audio`-Elements mehrere `source`-Elemente notieren. Natürlich ist es auch möglich, nur ein oder zwei `source`-Elemente anzugeben.

```
1 | <audio autoplay="autoplay" controls="controls">
2 |   <source src="/Musik/Beispiele/Tonleiter.mp3" type="audio/mpeg" />
3 |   <source src="/Musik/Beispiele/Tonleiter.ogg" type="audio/ogg" />
4 |   <source src="/Musik/Beispiele/Tonleiter.wav" type="audio/wav" />
5 | </audio>
```

### Video

Die Einbindung von Video-Dateien ist mit der Einbindung von Audio-Dateien zu vergleichen. Für Videos benötigen wir das `video`-Element. Die Attribute `autoplay`, `controls`, `loop` und `muted` gibt es hier ebenfalls. Deren Bedeutung entspricht der gleichen wie beim `audio`-Element. Zusätzliche Attribute für das `video`-Element sind `width`, `height` und `poster`. `width` und `height` legen die Breite und Höhe des anzuzeigenden **Video-Players** fest. Über das `poster`-Attribut kann die URL zu einem Bild angegeben werden, welches vor dem Abspielen des Videos oder während dem Laden angezeigt wird. Innerhalb der `video`-Tags werden ebenfalls wieder `source`-Elemente sowie bei Bedarf ein Text, welcher angezeigt werden soll, wenn der Browser keine Videos unterstützt, angezeigt. Die möglichen Formate für das `video`-Element sowie deren MIME-Typ und Browserunterstützung sind in folgender Tabelle aufgestellt:

Dateityp	MIME-Typ	Firefox	Chrome	Opera	Safari	IE / Edge
MP4	video/mp4	ja	ja	ja (bei Opera Mini, nein)	ja	ja (ab IE9)
WEBM	video/webm	ja	ja	ja (bei Opera Mini, nein)	nein	ja (ab Edge 14)
OGG	video/ogg	ja	ja	ja (bei Opera Mini, nein)	nein	nein

Hier scheint aus aktuellem Stand die Verwendung von MP4 am besten zu sein, da es von den meisten aktuellen Browsern unterstützt wird. Wie auch beim Audio-Beispiel haben wir in folgendem Beispiel-Code alle Dateitypen eingebunden:

```
1 | <video controls="controls" width="480" height="270" poster="/Bilder/Logo/Logo.png">
2 |   <source src="/Video/Beispiele/Event-Banner.mp4" type="video/mp4" />
3 |   <source src="/Video/Beispiele/Event-Banner.webm" type="video/webm" />
4 |   <source src="/Video/Beispiele/Event-Banner.ogg" type="video/ogg" />
5 | </video>
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Medien-Einbindung](#)



## Flash

Um Flash-Anwendungen in eine Webseite zu integrieren, gibt es das `embed`-Element. Das `embed`-Element ist ein universelles Element und dient allgemein zur Einbindung externer Daten und Medien. So ist es z. B. auch möglich, Videos per `embed`-Tag einzubinden. Hier ist jedoch Vorsicht in Bezug auf die Browserunterstützung geboten. Die Einbindung von Flash-Anwendungen per `embed`-Tag wird dagegen von allen aktuellen Browsern unterstützt. Das `embed`-Element ist einteilig und verfügt über 4 Attribute: `src`, `type`, `width` und `height`. `src` gibt die **Quelle der Datei** in Form einer URL an. Als Wert des `type`-Attributs ist der **MIME-Typ** der Datei anzugeben. Für Flash-Anwendungen ist dies `application/x-shockwave-flash`. `width` und `height` dienen zum Festlegen der **Breite und Höhe**. Die Angabe darf auch hier ebenfalls nur in Pixeln (ohne Einheit) erfolgen.

```
1 | <embed type="application/x-shockwave-flash" src="/Flash/Beispiele/Rechteckanimation.swf" width="220" height="250" />
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

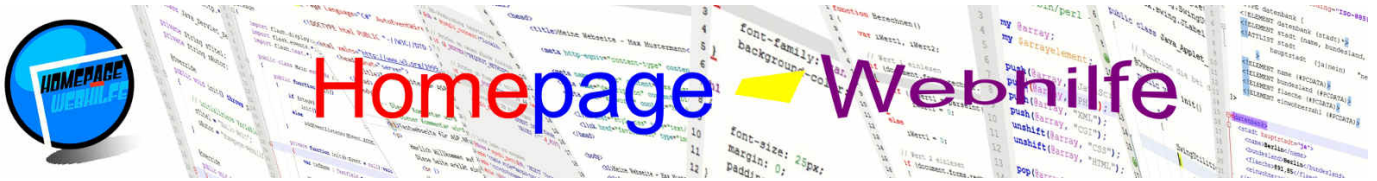
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [HTML](#) » [Abschluss](#)

## Abschluss

Nun sind wir auch schon am Ende dieses Kapitels angekommen. Doch zuletzt wollen wir Ihnen, als vielleicht angehender Webseiten-Betreiber oder Web-Programmierer, noch ein paar Dinge auf den Weg geben.

HTML ist komplexer als man denkt und leider unterstützen nicht immer alle Browser alle Elemente und Attribute. So ist es immer hilfreich, wenn man eine **Referenz** zur Hand hat, in welcher man nachschlagen kann. Eine solche Referenz findet man bspw. bei [W3Schools](#). W3Schools bietet eine sogenannte Tag-Referenz sowie eine Attribut-Referenz. Als weitere Referenz können natürlich auch unsere Karteikarten verwendet werden.

Für die meisten geht es nach dem Erlernen der Sprache HTML weiter mit anderen Sprachen. Hier sollte als erstes CSS gelernt werden. Mit Hilfe von CSS ist es möglich, Ihr HTML-Dokument zu „stylen“: Das Festlegen von **Layout, Schrift und Farbe** ist dabei zentrale Aufgabe von CSS. Ein [Tutorial zu CSS](#) finden Sie ebenfalls auf unserer Website.

Nachdem Sie Ihre HTML-Seiten erstellt und designt haben, kommen oft noch clientseitige Programmiersprachen wie [JavaScript](#) und [ActionScript](#) (für Flash-Anwendungen) und serverseitige Programmiersprachen wie [PHP](#), [Perl](#) oder [ASP.NET](#) zum Einsatz, mit welchen es möglich ist, aktive und dynamische Webdokumente zu erstellen.

Wir hoffen Ihnen hat dieses HTML-Tutorial gefallen. Sollten Sie **Fragen, Anregungen, Wünsche oder auch Kritik** haben, so können Sie uns diese gerne in unserem Forum, per Beratungs- oder Kontaktformular oder direkt per E-Mail zukommen lassen.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

Java EE

XML

# E-Book

## CSS





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Einführung](#)

## Einführung



CSS (Cascading Style Sheets) ist eine Stylesheet-Sprache, welche zur Formatierung von HTML und XML eingesetzt werden kann. Der Haupteinsatz findet sich hierbei jedoch in Verbindung mit HTML.

CSS ermöglicht neben „einfachen Formatierungen“ wie Text- oder Layout-Formatierungen auch Transformationen, Effekte und sogenannte Media Queries, mit welchen es möglich ist, unterschiedliche Formatierungen auf verschiedene Endgeräte anzuwenden.

CSS-Dateien werden unverändert an den Browser gesendet und können daher von jedem gelesen werden.

Da CSS-Skripte reine Text-Dateien sind und vom Browser interpretiert und verarbeitet werden, benötigen wir um CSS-Dateien zu erstellen und zu bearbeiten lediglich einen Text-Editor. Natürlich empfehlen wir Ihnen einen [Editor mit Syntax-Highlighting](#) zu verwenden.

### Inhalt dieser Seite:

1. Geschichte
2. Syntax
3. Farben
4. Einheiten
5. Platzierung
6. Beispiel

## Geschichte

CSS wurde entworfen, um eine Trennung zwischen Inhalt und Design herzustellen. In älteren HTML-Versionen gab es unter anderem das `font`-Element, mit welchem Schriftfarbe, Schriftgröße und Schriftart festgelegt werden konnte. Des Weiteren wurden früher Tabellen verwendet, um Layouts mittels CSS zu erstellen. Ein solches Schema widerspricht dem **MVC-Konzept** (*Model View Controller*).

Die ersten Ausarbeitungen und Entwicklungen des CSS-Vorgängers CHSS (Cascading HTML Style Sheet) fanden bereits in den Jahren 1994 statt. Ende 1996 wurde die erste Spezifikation von CSS namens *CSS Level 1* vom W3C (World Wide Web Consortium) veröffentlicht.

Die 2. Version von CSS (offiziell *CSS Level 2*) wurde im Mai 1998 veröffentlicht. Mitte 2011 wurde *CSS Level 2 Revision 1* veröffentlicht, welche seit 2014 von fast allen Webbrowsern so gut wie vollständig unterstützt wird.

Parallel zur Entwicklung von CSS2 wurde im Jahre 2000 mit der Entwicklung von CSS3 begonnen. CSS3 enthält neben der Weiterentwicklung von CSS2-Bestandteilen auch neue Funktionen wie z. B. Transformationen, Schatten, Transparenz und Media Queries.

Mittels CSS wird das MVC-Konzept wieder hergestellt bzw. aufrechterhalten. So könnte man am Beispiel von Webseiten sagen, dass der HTML-Code das Modell (M = Model) und der CSS-Code die Darstellung (V = View) ist.

## Syntax

Der Syntax von CSS ist ziemlich einfach aufgebaut. Als erstes notieren wir immer einen oder mehrere **Selektoren**. Selektoren erlauben (wie der Name schon sagt) das Selektieren von HTML-Elementen. Dadurch können wir also mittels CSS auf HTML-Dokumente „zugreifen“. CSS bietet verschiedene Arten von Selektoren: Element-Selektoren, Klassen, IDs und Attribut-Selektoren. Auf diese werden wir im [nächsten Thema](#) genauer eingehen. Nach dem Selektor folgt immer ein **Block**, welcher in geschweifte Klammern gesetzt werden muss. Hier ein schematischer Aufbau:

```
1 | selektor
2 | {
3 |
4 | }
```

Um z. B. einen „Block“ für mehrere Elemente oder einen „Block“ nur für Elemente, die einem anderen Element untergeordnet sind anzuwenden, gibt es sogenannte **Kombinatoren**. Diese werden wir ebenfalls im nächsten Thema vorstellen.

Innerhalb des Blocks können nur sogenannte **Eigenschaften** platziert werden. Eigenschaften ermöglichen das Festlegen verschiedener Einstellungen. Diese Einstellung kann sich auf Schrift, Layout und anderes auswirken. Die Liste der verfügbaren Eigenschaften ist lang. Die wichtigsten und meist genutzten Eigenschaften werden Sie innerhalb dieses CSS-Kurses kennenlernen. Nach dem Namen der Eigenschaft muss ein Doppelpunkt `:` notiert werden. Im Anschluss wird der dazugehörige **Wert** bzw. die dazugehörigen Werte notiert. Um die Eigenschaft abzuschließen, müssen wir noch am Ende ein Semikolon `;` notieren. Hier nochmals ein weiterer schemenhafter CSS-Code:

```
1 | selektor
2 | {
3 |   eigenschaft1: wert1;
4 |   eigenschaft2: wert2;
5 |   eigenschaft3: wert3;
6 | }
```

Die in den zwei obigen Codes enthaltenen Leerzeichen und Zeilenumbrüche sind nur „Kosmetik“. Sie dienen also nur zur Verbesserung der **Lesbarkeit**. Da Sie mit Ihren CSS-Dateien immer wieder arbeiten werden, sollten Sie hier (sowie im HTML-Code auch) auf eine übersichtliche Darstellung und einen guten Aufbau achten.

Zur besseren Lesbarkeit dienen auch **Kommentare**. Kommentare beginnen in CSS mit `/*` und enden mit `*/`. Kommentare können sich dabei über mehrere Zeilen erstrecken. Genutzt werden Kommentare rein zu Dokumentations- und Testzwecken. Bitte bedenken Sie, dass alle Personen die Zugang zu Ihrer Website haben, auch den Quelltext und somit auch die Kommentare Ihrer CSS-Dateien lesen können.

```
1 | /* einzeiliger Kommentar */
2 | /* ein mehrzeiliger
3 |    Kommentar */
```

## Farben

Bei diversen Eigenschaften von CSS ist es notwendig, eine Farbe anzugeben. Für die Angabe einer Farbe gibt es 6 verschiedene Möglichkeiten. Zum einen ist die Angabe eines **Farbnamens** in englischer Sprache möglich. Die CSS-Spezifikation enthält über 100 vordefinierte Farbnamen. Hierzu zählen gängige Namen wie `red`, `green`, `lime`, `blue`, `orange`, `white`, `black` und viele mehr. Die besseren Editoren (z. B. [Adobe Brackets](#)) enthalten diese Farbliste, weshalb diese dort ganz einfach mittels Autovervollständigung abgerufen werden können.

```
1 | red
```

Um Farben im **RGB-Farbraum** anzugeben, gibt es das Schlüsselwort `rgb`. Hinter diesem muss ein rundes Klammernpaar notiert werden. Innerhalb der Klammer werden die drei Werte des Farbraums (Rotanteil, Grünanteil, Blauanteil) angegeben. Die Werte sind dabei Dezimalzahlen zwischen 0 und 255. Eine Einheit wird hier

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Einführung](#)

nicht angegeben.

```
1 | rgb(107, 234, 45)
```

Wollen wir zusätzlich zu den Farbanteilen noch die **Deckkraft** angeben, so verwenden wir das Schlüsselwort `rgba`. Hier muss dann noch zusätzlich ein 4. Wert angegeben, welcher zwischen 0.0 und 1.0 liegen muss. 1.0 bedeutet dabei die volle Deckkraft. Wird das Schlüsselwort `rgb` verwendet, so besitzt die Farbe immer die volle Deckkraft.

```
1 | rgba(107, 234, 45, 0.7)
```

An Stelle der Notation mit dem `rgb`-Schlüsselwort kann der Farbwert auch **hexadezimal** angegeben werden. Das dafür vorgesehene Format ist `#RRGGBB`. Da die einzelnen Bestandteile eines RGB-Wertes zwischen 0 und 255 in Dezimalangabe liegen, befindet sich der Wertebereich in hexadezimaler Schreibweise zwischen 00 und FF.

```
1 | #6BEA2D
```

Viele Designer arbeiten nicht mit dem RGB-Farbraum, sondern mit dem **HSL-Farbraum**, da man sich die Farben besser vorstellen kann. Der HSL-Farbraum setzt sich aus dem Farbwert (Hue), der Sättigung (Saturation) und der Helligkeit (Lightness) zusammen. Der Farbwert wird in Grad angegeben (ohne Einheit). Die Sättigung und die Helligkeit werden in Prozent (inkl. Prozentzeichen) angegeben. Um HSL-Werte in CSS zu notieren, benötigen wir das Schlüsselwort `hsl`, ein rundes Klammerspaar und die drei Werte, welche durch Komma getrennt werden.

```
1 | hsl(207, 82%, 35%)
```

Auch HSL-Farbangaben können noch um die Deckkraft erweitert werden. Hierzu benötigen wir das Schlüsselwort `hsla` an Stelle von `hsl`. Auch hier wird die Deckkraft als 4. Parameter mit einem Wert zwischen 0.0 und 1.0 angegeben.

```
1 | hsla(207, 82%, 35%, 0.4)
```

## Einheiten

Für einige CSS-Eigenschaften (Positionierung, Rahmen, Größe uvm.) muss zusätzlich zu einer Zahl eine Einheit angegeben werden, mit welcher spezifiziert wird, in welchem „Format“ die Zahl angegeben wurde. In CSS wird zwischen absoluten und relativen Einheiten unterschieden. Alle Zahlen können dabei Ganzzahlen, aber auch **Zahlen mit Nachkommastellen** sein. Bei einigen Eigenschaften (z. B. bei der Positionierung von Elementen) ist es auch möglich, **negative Zahlen** zu notieren. Hierfür muss direkt vor der Zahl ein Minuszeichen / Bindestrich notiert werden.

Bei **absoluten Einheiten** wird sozusagen sichergestellt, dass die **Darstellung auf allen Endgeräten gleich** ist. Als Einheiten stehen uns Millimeter `mm`, Zentimeter `cm`, Inch `in`, Pixel `px`, Points `pt` und Pica `pc` zur Verfügung.

```
1 | 100mm
```

```
1 | 10cm
```

```
1 | 30in
```

```
1 | 16px
```

```
1 | 24pt
```

```
1 | 8pc
```

Einige Personen behaupten, dass Pixel keine absolute Einheit ist. Dies liegt daran, dass Pixel je nach Größe des Bildpunktes unterschiedlich groß sein können. Trotzdem hat es sich durchgesetzt, dass Pixel als absolute Einheit gilt.

**Relative Einheiten** beziehen sich auf bestimmte andere Angaben (z. B. das übergeordnete Element). Die Benutzung von relativen Einheiten hat in den letzten Jahren zugenommen. Dies liegt auch daran, dass relative Einheiten für **responsives Webdesign** praktischer sind. Dennoch werden für Schriftgrößen und ähnliches weiterhin gerne absolute Einheiten verwendet. Als relative Einheiten stehen uns Prozent `%`, `em` und `rem` zur Verfügung. Die `em`-Einheit gibt den Faktor zur Multiplikation mit der Schriftgröße an. Verwendet wird die Einheit oftmals für die Zeilenhöhe. Die `rem`-Einheit ist fast identisch mit der `em`-Einheit. Der einzige Unterschied ist, dass sich die Multiplikation nicht auf die Schriftgröße des aktuellen Elements, sondern auf die Schriftgröße des Wurzel-Elements (`html`-Element) bezieht.

```
1 | 60%
```

```
1 | 1.3em
```

```
1 | 1.6rem
```

**Übrigens:** Wenn Sie einer Eigenschaft den Zahlenwert `0` zuweisen möchten, ist es gängige Praxis, die Einheit wegzulassen. Laut CSS-Spezifikation ist diese Vereinfachung erlaubt.

**Übrigens:** Im [Kapitel Weiterführendes](#) finden Sie einen Umrechner für die absoluten Maßeinheiten.

## Platzierung

Nun haben wir bereits grundlegendes über unseren späteren CSS-Code gelernt. Doch wo wird der Code eigentlich platziert? Hierfür gibt es insgesamt 3 Möglichkeiten, wovon nicht immer alle für jeden Einsatzzweck geeignet sind.

Die gängigste Variante ist, den CSS-Code in einer **separaten Datei** zu platzieren. Die Dateien verfügen dabei standardmäßig über die Endung `.css`. Nun benötigen wir noch einen Link in unserer HTML-Datei, um die HTML-Datei mit der CSS-Datei zu verbinden. Hierfür benötigen wir das `link`-Element mit dem `href`- und `type`-Attribut. Das `href`-Attribut gibt die URL zur Datei an. Das `type`-Attribut enthält den MIME-Typ, welcher bei CSS `text/css` ist. Für größere Projekte kann es auch nützlich sein, mehrere CSS-Dateien zu erstellen. Wollen wir mehrere Dateien in eine HTML-Seite einbinden, müssen wir einfach das `link`-Element mehrmals notieren.

```
1 | <link rel="stylesheet" href="layout.css" type="text/css" />
```

Und so könnte der schematische Inhalt der CSS-Datei `layout.css` aussehen:

```
1 | selektor
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Einführung](#)

```

2 | {
3 |   eigenschaft1: wert1;
4 |   eigenschaft2: wert2;
5 |   eigenschaft3: wert3;
6 | }

```

Der Vorteil der obigen Methode ist, dass wir eine CSS-Datei für mehrere HTML-Seiten verwenden können. Wollen wir nun das Design unserer Website anpassen (z. B. Schriftart, Farbe etc.), müssen wir nur die CSS-Datei editieren.

Es kann aber auch sein, dass es bestimmte CSS-Regeln gibt, welche nur für eine spezielle HTML-Seite gelten soll. In einem solchen Fall müssen wir nicht unbedingt eine extra CSS-Datei anlegen, sondern können den **CSS-Code im HTML-Code** notieren. Der CSS-Code muss hierfür innerhalb des `style`-Elements angegeben werden. Das `style`-Element muss über das `style`-Attribut verfügen (wie das `link`-Attribut) und muss innerhalb des Head-Bereichs notiert sein.

```

1 | <style type="text/css">
2 |   selektor
3 |   {
4 |     eigenschaft1: wert1;
5 |     eigenschaft2: wert2;
6 |     eigenschaft3: wert3;
7 |   }
8 | </style>

```

Um einem **einzelnen HTML-Element** eine oder mehrere CSS-Eigenschaften zuzuweisen, können wir innerhalb aller darstellbaren HTML-Elemente das `style`-Attribut verwenden. Als Wert werden CSS-Eigenschaften und deren Werte notiert. Auch hier gilt das bekannte Format mit Doppelpunkt zwischen Eigenschaft und Wert und Semikolon am Ende einer Eigenschaft. Es fällt also lediglich der Selektor und die geschweiften Klammern weg.

```

1 | <p style="eigenschaft1: wert1; eigenschaft2: wert2; eigenschaft3: wert3;">...</p>

```

Alternativ dazu ist es möglich, ein einzelnes Element mittels ID per CSS zu selektieren. Dazu im nächsten Thema mehr.

## Beispiel

Nachdem wir bisher immer nur mit schematischem Code gearbeitet haben, wollen wir uns nun den ersten fertigen Code anschauen. Der folgende CSS-Code selektiert alle `h1`-Elemente und weist diesen eine blaue (englischer Farbname `blue`) Schriftfarbe (Eigenschaft `color`) zu.

```

1 | h1
2 | {
3 |   color: blue;
4 | }

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

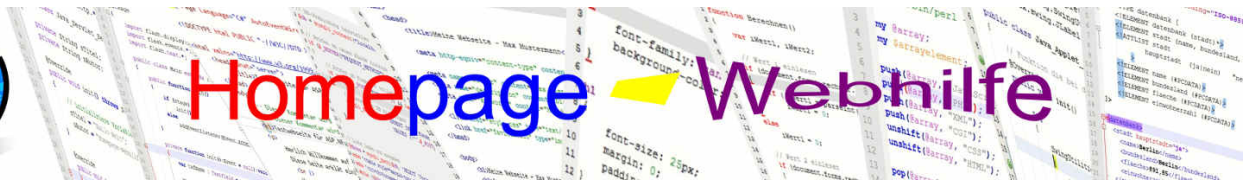
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Selektoren](#)

## Selektoren

Selektoren dienen dazu, bestimmte Elemente Ihres HTML-Dokuments zu selektieren (auszuwählen) und auf diese CSS-Eigenschaften anzuwenden. Unterschieden wird hier zwischen 3 verschiedenen Arten von Selektoren, welche wir Ihnen hier vorstellen werden. Am Ende dieses Themas werden Sie zudem noch die Kombinatoren kennenlernen, welche es ermöglichen, den Selektor noch genauer zu spezifizieren.

### Inhalt dieser Seite:

1. Element-Selektoren
2. Klassen und ID's
3. Attribut-Selektoren
4. Kombinatoren
5. Vererbung und Gewichtung

### Element-Selektoren

Die einfachste Art von Selektoren sind die Element-Selektoren. Der Name von Element-Selektoren entspricht immer dem Namen des (HTML-)Elements. Wir können hier also z. B. alle Elemente des Typs `h1`, `h2`, `h3`, `p`, `article`, `section` oder `nav` auswählen, indem wir lediglich den Namen des HTML-Elements als Selektor angeben. Das folgende Beispiel weist den Überschriften `h1`, `h2` und `h3` unterschiedliche Schriftfarben zu. Dies geschieht mittels der `color`-Eigenschaft, welche wir im Thema [Schrift](#) nochmals genauer erklären werden.

```

1 | h1
2 | {
3 |     color: blue;
4 | }
5 |
6 | h2
7 | {
8 |     color: red;
9 | }
10 |
11 | h3
12 | {
13 |     color: lime;
14 | }
```

**Übrigens:** Wenn Sie alle Elemente selektieren möchten, können Sie das Sternzeichen `*` als Selektor verwenden. Eingesetzt wird dieser Selektor z. B. um eine Schriftart für die ganze Seite zu setzen. Eine Verwendung des `*`-Selektors in Zusammenhang mit Kombinatoren ist ebenfalls möglich.

### Klassen und ID's

Mit der obigen Variante ist es jedoch nicht möglich, ein **einzelnes Element** oder eine **Gruppe von Elementen** zu selektieren. Stellen Sie sich vor, Sie haben mehrere `h1` Überschriften, aber nur eine davon soll per CSS über eine spezielle Formatierung designet werden. Ein anderes Beispiel wäre, dass Sie mehrere `h2`-Überschriften haben und nur ein paar dieser `h2`-Überschriften sollen bestimmte CSS-Regeln zugewiesen bekommen. Für solche Fälle gibt es sogenannte **Klassifizierungen** und **Identifikationen** (abgekürzt Klassen und ID's).

Um ein Element zu klassifizieren, benötigen wir das `class`-Attribut. Dieses wird innerhalb des Start-Tags unseres HTML-Elements notiert. Soll ein Element **mehrere Klassen** besitzen, so werden diese über Leerzeichen getrennt. Der Name der Klasse ist frei definierbar, darf jedoch keinen Punkt und kein Leerzeichen enthalten und muss für dieses Dokument eindeutig sein. Normalerweise werden für Klassennamen nur Buchstaben und Unterstriche verwendet, da es andernfalls Kompatibilitäts-Probleme mit manchen (u. a. älteren) Browsern geben kann. Eine Klassifizierung kann auf **mehrere Elemente** angewendet werden. Hierzu bekommen einfach alle Elemente, die zu dieser Klasse gehören, das `class`-Attribut mit dem Namen der Klasse zugewiesen. Der Typ der Elemente muss dabei nicht zwingend gleich sein. In CSS werden Klassen-Selektoren direkt mit dem Namen und einem vorangestellten Punkt `.` angegeben.

Mit Hilfe der Identifikation kann ein **einzelnes Element** direkt „angesprochen“ werden. Eine Identifikation erfolgt in HTML über das `id`-Attribut, welches Sie vielleicht bereits aus dem HTML-Kurs (Thema [Links, Anker](#)) kennen. Die Namensregeln sind hierbei gleich wie bei Klassen. Jedes Element darf jedoch maximal eine ID besitzen. Des Weiteren ist darauf zu achten, dass ein Identifikationsname nur **einmal pro Dokument** verwendet werden darf. Die Angabe einer ID als Selektor in CSS erfolgt mit einem `#`-Zeichen und dem Namen der Identifikation.

```

1 | <h1 id="titel">Überschrift 1</h1>
2 | <h2 class="untertitel">Überschrift 1.1</h2>
3 | <h2>Überschrift 1.2</h2>
4 | <h2 class="untertitel">Überschrift 1.3</h2>
5 | <h1>Überschrift 2</h1>
6 | <h2>Überschrift 2.1</h2>
7 | <h2 class="untertitel">Überschrift 2.2</h2>
8 | <h2 class="untertitel">Überschrift 2.3</h2>
9 | <h2 class="untertitel">Überschrift 2.4</h2>
10 |
11 | #titel
12 | {
13 |     color: blue;
14 | }
15 |
16 | .untertitel
17 | {
18 |     color: red;
19 | }
```

**Übrigens:** Oftmals könnten wir Formatierungen von Identifikationen direkt im `style`-Attribut des HTML-Elements angeben. Auf Grund des MVC-Konzepts wird jedoch das Verfahren mit ID's bevorzugt. Das `style`-Attribut sollte nur verwendet werden, wenn es sich um eine einzelne Formatierung, eine Ausnahme oder einen Test handelt.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Selektoren](#)

## Attribut-Selektoren

Bisher konnten wir nur Elemente über den Element-Namen oder über den „Umweg“ mit Klassen und ID's selektieren. Doch es ist auch möglich, **Elemente mittels dessen Attribute zu selektieren**. Das Attribut wird hierfür in eckige Klammern [ ] notiert. Innerhalb dieser muss der Name des Attributs notiert werden. Es ist natürlich auch möglich, Elemente mit einem bestimmten Attribut und einem **bestimmten Wert** zu selektieren. Alle verfügbaren Attribut-Selektoren sind in folgender Tabelle aufgestellt:

[name]	Selektiert Elemente mit dem name-Attribut.
[name="wert"]	Selektiert Elemente mit dem name-Attribut, bei welchen der Wert "wert" entspricht.
[name~="wert"]	Selektiert Elemente mit dem name-Attribut, welche im Wert das Wort "wert" enthalten.
[name = "wert"]	Selektiert Elemente mit dem name-Attribut, welche im Wert mit dem Wort "wert" beginnen.
[name^="wert"]	Selektiert Elemente mit dem name-Attribut, welche im Wert mit den Zeichen "wert" beginnen.
[name\$="wert"]	Selektiert Elemente mit dem name-Attribut, welche im Wert mit den Zeichen "wert" enden.
[name*="wert"]	Selektiert Elemente mit dem name-Attribut, welche im Wert die Zeichen "wert" enthalten.

Und hier nun noch ein Beispiel mit Attribut-Selektoren:

```

1 <h1 title="Überschrift 1">Beispiele</h1>
2 <a href="sektorenelemente.html" target="_blank">Zum Beispiel "Element-Selektoren" ...</a><br />
3 <a href="sektorenklassenid.html" target="_blank">Zum Beispiel "Klassen und IDs" ...</a><br />
4 <br />
5 <p title="Klicken Sie auf den Link, um zurück zur Erklärung zu kommen.">
6   Link zum Thema:
7   <a href="https://www.homepage-webhilfe.de/CSS/selektoren.php">hier klicken</a>
8 </p>
9
1 [title]
2 {
3   color: red;
4 }
5
6 [target="_blank"]
7 {
8   color: green;
9 }
    
```



## Kombinatoren

Kombinatoren erlauben es, Selektoren noch genauer zu spezifizieren. So können Sie z. B. mit Kombinatoren bestimmte Elemente (per Typ oder Klasse) selektieren, welche sich in einem anderen Element befinden. Eine Aufstellung der Kombinationen finden Sie in folgender Tabelle:

a b	Leerzeichen: Selektiert alle b-Elemente, welche sich innerhalb eines a-Elements befinden.
a > b	Größer-als: Selektiert alle b-Elemente, welche sich direkt innerhalb eines a-Elements befinden.
a, b	Komma: Selektiert alle a- und b-Elemente.
a ~ b	Tilde: Selektiert alle b-Elemente, welche sich nach einem a-Element befinden.
a + b	Plus: Selektiert alle b-Elemente, welche sich direkt nach einem a-Element befinden.

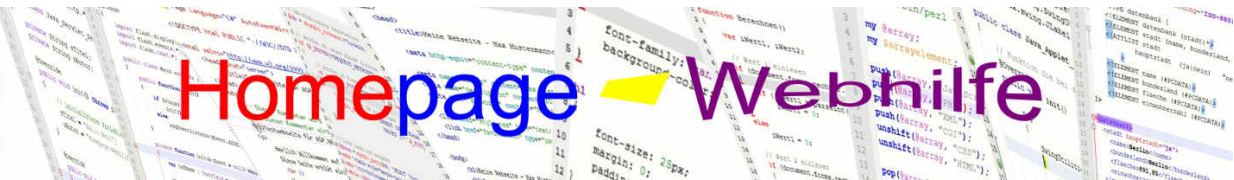
Das folgende Beispiel zeigt die Verwendung der Kombinatoren `und` und `>`:

```

1 <div>
2   <p>...</p>
3   <article>
4     <p>...</p>
5     <p>...</p>
6   </article>
7 </div>
8
9
1 div p
2 {
3   color: blue;
4 }
5
6 div > p
7 {
8   color: red;
9 }
    
```



<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Selektoren](#)

**Wichtig:** Beim obigen Beispiel ist die Reihenfolge der Notation der zwei CSS-Regelblöcke entscheidend, denn die Regel `div p` trifft auch auf alle `p`-Elemente innerhalb des `div`-Elements zu und somit auch auf die `p`-Elemente innerhalb des `article`-Elements. Durch die Regel `div > p` wird jedoch die erste Regel für alle `p`-Elemente außerhalb des `article`-Elements überschrieben.

Das folgende Beispiel zeigt die Verwendung der Kombinatoren `,`, `+` und `~`:

```

1 <main>
2 <h1>Überschrift 1</h1>
3 <article>
4 <h2>Überschrift 1.1</h2>
5 <h3>Überschrift 1.1.1</h3>
6 <h3>Überschrift 1.1.2</h3>
7 </article>
8 <article>
9 <h2>Überschrift 1.2</h2>
10 <h3>Überschrift 1.2.1</h3>
11 <h3>Überschrift 1.2.2</h3>
12 </article>
13 <article>
14 <h2>Überschrift 1.3</h2>
15 <h3>Überschrift 1.3.1</h3>
16 <h3>Überschrift 1.3.2</h3>
17 <h3>Überschrift 1.3.3</h3>
18 </article>
19 </main>
20 <article>
21 <h1>Überschrift 2</h1>
22 <h3>Überschrift 2.1.1</h3>
23 <h3>Überschrift 2.1.2</h3>
24 </article>
1 h1, h2
2 {
3   color: blue;
4 }
5
6 h2 + h3
7 {
8   color: red;
9 }
10
11 h1 ~ h3
12 {
13   color: lime;
14 }
    
```



### Vererbung und Gewichtung

Zum Schluss dieses Themas müssen wir uns noch mit einem theorielastigen Teil beschäftigen, denn bei CSS gibt es noch die sogenannte Vererbung und Gewichtung.

Die **Vererbung** legt fest, dass Formatierungen, die auf ein Element angewendet werden, nicht nur auf das Element selbst angewendet wird, sondern auch auf dessen Kind-Elemente. Vererbt werden natürlich nicht alle CSS-Eigenschaften, sondern hauptsächlich **Schriftformatierungen**.

Als **Gewichtung** wird der Rang bezeichnet, welche die einzelnen CSS-Regeln haben. So haben z. B. Formatierungen, welche im `style`-Attribut eines Elements angegeben werden Vorrrechte und überschreiben somit CSS-Regeln, welche innerhalb der `style`-Tags oder in einer CSS-Datei notiert sind.

Eigenschaften, welche für bestimmte Elemente gelten (Element-Selektoren), können durch Eigenschaften von Klassen oder ID's überschrieben werden. Stellen Sie sich vor, Sie haben ein `p`-Element. Alle `p`-Elemente verfügen über die Schriftfarbe blau, da Sie diese mittels der `color`-Eigenschaft in einem Element-Selektor für das `p`-Element festgelegt haben. Nun haben Sie eine Klasse mit dem Namen `schwarz`, in welcher die Schriftfarbe auf schwarz gesetzt wird. Wird die Klasse nun bei einem der `p`-Elemente notiert, so wird die ursprüngliche Schriftfarbe von blau durch schwarz ersetzt.

Verfügt ein Element sowohl über eine Klasse als auch über eine ID, so haben die Eigenschaften der ID gegenüber den Eigenschaften der Klasse Vorrang.

Bei der Gewichtung spielt zusätzlich noch die **Notationsreihenfolge** eine entscheidende Rolle. Haben wir also z. B. zwei Klassen in welchen beide die Schriftfarbe notiert ist und ein `p`-Element, welches beide Klassen zugewiesen bekommen hat, dann „gewinnt“ die zuletzt notierte Schriftfarbe (also die der 2. Klasse).

Zum Schluss muss noch erwähnt werden, dass CSS-Regeln, welche **im Browser implementiert** sind (z. B. die Schriftfarbe von Links), explizit überschrieben werden müssen. Es reicht also nicht z. B. nur dem `p`-Element, in welchem wir das `a`-Element platziert haben, eine Schriftfarbe zuzuweisen. Diese Schriftfarbe wird dann zwar für das `p`-Element übernommen, nicht jedoch für die enthaltenen Links.

**Wichtig:** Es gibt auch die Möglichkeit, Selektoren zu **überqualifizieren** (*overqualified*), d. h. dass z. B. zusätzlich zu einem Klassennamen auch noch ein Elementname angegeben wird. Dies hat beim genannten Beispiel zur Folge, dass die Klasse auf ein bestimmtes Element eingeschränkt wird. Wir könnten dadurch also auch für einen Klassennamen zwei komplett unterschiedliche CSS-Regeln entwerfen, wovon sich die eine z. B. auf `h1`-Elemente und die andere auf `p`-Elemente beziehen. Bei der Notation müssen wir vor dem Klassennamen das Element angeben. Hinter dem Elementname folgt dann direkt der Punkt und der Name der Klasse. Hier ein kurzes Beispiel:

```

1 h1.text
2 {
    
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Selektoren](#)

```

3     color: red;
4 }
5
6 p.text
7 {
8     color: blue;
9 }
    
```

Das Gleiche ist natürlich auch mit Identifikationen möglich:

```

1 h1#text
2 {
3     color: red;
4 }
5
6 p#text
7 {
8     color: blue;
9 }
    
```

Eine Verwendung von überqualifizierten Selektoren im Zusammenhang mit Attribut-Selektoren ist ebenfalls denkbar:

```

1 h1[title]
2 {
3     color: red;
4 }
5
6 p[title]
7 {
8     color: blue;
9 }
    
```

Die Kombination der Selektoren beim Überqualifizieren ist dabei auf ganz verschiedene Art und Weisen möglich. So wäre z. B. auch folgender Syntax gültig:

```

1 h1[title]#text
2 {
3     color: red;
4 }
5
6 p[title].text
7 {
8     color: blue;
9 }
    
```

Eine Überqualifizierung sollte jedoch immer nur dann eingesetzt werden, wenn sie wirklich von Nöten ist, da das Überqualifizieren leichte **Performance-Einbrüche** mit sich bringt.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Schrift](#)

## Schrift

Nachdem wir uns nun mit den wichtigsten Grundlagen von CSS beschäftigt haben, können wir nun damit anfangen, unsere ersten Formatierungen durchzuführen. Die ersten Formatierungen, welche wir uns nun genauer anschauen wollen, sind Schriftformatierungen.

### Inhalt dieser Seite:

1. Farbe
2. Schriftart
3. Schriftgröße
4. Schriftstile
5. Ausrichtung
6. Textschatten

### Farbe

Die Eigenschaft zum Festlegen der Schriftfarbe ist `color`. Als Wert für die `color`-Eigenschaft sind alle Varianten zur Angabe der Farbe (Name, RGB, RGBA, HSL, HSLA) möglich.

```

1  h1
2  {
3      color: blue;
4  }
5
6  p
7  {
8      color: red;
9  }
```



### Schriftart

Die Schriftart wird in CSS über die Eigenschaft `font-family` festgelegt. Dabei ist darauf zu achten, dass nur wenige Schriftarten als „Web Safe“ (websicher) gelten. Die geläufigsten Schriftarten, welche von den meisten Betriebssystemen unterstützt werden, sind Arial, Georgia, Helvetica, Tahoma, Times und Verdana. Schriftarten, welche Leerzeichen enthalten (z. B. Arial Black und Times New Roman), müssen in doppelte oder einfache Anführungszeichen gesetzt werden.

```

1  h1
2  {
3      font-family: Arial;
4  }
5
6  p
7  {
8      font-family: Times;
9  }
```



Für Websites sollten wir zudem nicht nur eine, sondern mehrere Schriftarten angeben. Dadurch gibt es einen sogenannten „Fallback“, d. h. wenn der Browser die erste Schriftart nicht finden kann, so versucht er die zweite zu verwenden, fehlt auch diese, so verwendet er die dritte usw.. Zudem empfiehlt es sich, als letzte Schriftart den Namen `serif` (für Serifen-Schriftarten) oder `sans-serif` (für „normale“ Schriftarten) anzugeben. Hierdurch hat der Browser die Möglichkeit, die Standard-Serifen-Schriftart oder die Standard-Schriftart zu verwenden, falls keine der angegebenen Schriftarten zur Verfügung steht. Dies sieht dann z. B. so aus:

```

1  h1
2  {
3      font-family: Arial, Helvetica, sans-serif;
4  }
```

### Schriftgröße

Die Schriftgröße wird über die Eigenschaft `font-size` festgelegt. Hier ist die Angabe in allen von CSS unterstützten Einheiten erlaubt. Zudem ist laut CSS-Spezifikation die Angabe der Schlüsselwörter `medium`, `small`, `x-small`, `xx-small`, `smaller`, `large`, `x-large`, `xx-large` und `larger` möglich. Diese sollten jedoch weitestgehend vermieden werden, da Browser diese Angaben unterschiedlich interpretieren können und es somit nicht gewährleistet ist, dass die Schriftgröße in allen Browsern gleich ist.

```

1  h1
2  {
3      font-size: 40px;
4  }
5
6  p
7  {
8      font-size: 20px;
9  }
```



### Schriftstile

CSS erlaubt die Angabe verschiedener Schriftstile.

Die Eigenschaft `font-weight` legt die **Gewichtung** der Schrift fest. Als Werte sind `normal` und `bold` möglich. Die CSS-Spezifikation enthält noch weitere Werte für diese Eigenschaft, welche jedoch von den meisten Browsern nicht unterstützt werden. Über den Wert `bold` erhalten wir das gleiche Ergebnis, wie wenn wir den gewünschten Text in einem `b`-Element platzieren würden.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Schrift](#)

`font-variant` ist die Eigenschaft, mit welcher wir die Schrift als **Kapitälchen** (Kleinbuchstaben in Form von Großbuchstaben) darstellen können. Hierfür muss der Eigenschaft lediglich der Wert `small-caps` zugewiesen werden. Der Wert `normal` deaktiviert die Funktion und ist zudem der Standardwert der Eigenschaft.

Über die Eigenschaft `word-spacing` und `letter-spacing` lässt sich der **Abstand zwischen Wörtern oder Zeichen** ändern. Die Angabe erfolgt als Zahl mit einer der verfügbaren Einheiten. Über den Wert `normal` lässt sich die Standardeinstellung wieder herstellen. Der Abstand zwischen Wörtern ist laut CSS-Spezifikation 0.25em.

Möchten wir einen Text **kursiv** darstellen (sowie mit dem `i`-Element), dann können wir die Eigenschaft `font-style` und den Wert `italic` verwenden. Der Wert `normal` ist hier ebenfalls verfügbar und ist als Standardeinstellung anzusehen und sorgt dafür, dass der Text nicht kursiv dargestellt wird.

Mit der Eigenschaft `text-decoration` können wir unserem Text eine „Dekoration“ hinzufügen. Die „Dekoration“ erfolgt in Form eines unterstrichenen (`underline`), überstrichenen (`overline`) oder durchgestrichenen (`line-through`) Textes. Mit Hilfe des Werts `none` wird die „Dekoration“ entfernt. Verwendung findet diese Eigenschaft vor allem bei Links.

Als Letztes wollen wir noch die Eigenschaft `text-transform` vorstellen, welche es erlaubt, die Buchstaben eines Textes **umzuformatieren**. Der Wert `uppercase` stellt alle Zeichen als Großbuchstaben dar, der Wert `lowercase` hingegen als Kleinbuchstaben. Mit Hilfe des Werts `capitalize` werden alle Anfangsbuchstaben eines Worts in Großbuchstaben dargestellt. Die Standardeinstellung ist `none`, welche den Text normal und somit ohne Transformation darstellt.

```

1 | h1
2 | {
3 |     font-weight: normal;
4 |     text-transform: uppercase;
5 |     text-decoration: underline;
6 | }
7 |
8 | p
9 | {
10 |     font-style: italic;
11 |     font-variant: small-caps;
12 |     letter-spacing: 1px;
13 |     word-spacing: 10px;
14 | }
```



### Ausrichtung

Um einen **Text auszurichten**, gibt es in CSS die Eigenschaft `text-align`. Als Werte sind `left` (linksbündig), `right` (rechtsbündig), `center` (zentriert) und `justify` (Blocksatz) möglich.

Die Eigenschaft `text-indent` ermöglicht es, die **erste Zeile einzurücken**. Hierfür muss als Wert der Eigenschaft eine Zahl in Verbindung mit einer der bekannten Einheiten angegeben werden.

Mit Hilfe der Eigenschaft `white-space` kann die Darstellung von Leerzeichen und Umbrüchen gesteuert werden. Der Wert `normal` ist die Standardeinstellung und fasst alle Leerzeichen zu einem zusammen. Des Weiteren werden vom Browser automatisch Zeilenumbrüche durchgeführt. Der Wert `nowrap` führt dazu, dass keine automatischen Zeilenumbrüche durchgeführt werden. Mehrere Leerzeichen werden jedoch weiterhin durch ein einzelnes Leerzeichen ersetzt. Als weitere Werte stehen `pre`, `pre-line` und `pre-wrap` zur Verfügung. Bei `pre` und `pre-wrap` werden die Leerzeichen vom Browser beibehalten - bei `pre-line` nicht. Wird der Wert `pre-line` oder `pre-wrap` verwendet, so wird der Text umgebrochen, sobald es notwendig ist (wie bei `normal`). Der Wert `pre` verhält sich in Anbetracht auf Umbrüche so, dass ein Umbruch erst erfolgt, sobald dieser explizit notiert wurde. Des Weiteren ist darauf zu achten, dass `pre`, `pre-line` und `pre-wrap` bei Zeilenumbrüchen auf Umbrüche im HTML-Code reagieren. Die Werte verhalten sich also alle ähnlich wie das `pre`-Element.

Die Eigenschaft `line-height` gibt die **Zeilenhöhe** an. Die Angabe erfolgt hierbei oftmals in der em-Einheit. Natürlich sind aber auch Angaben in Pixel oder anderen Einheiten zulässig. Mit einem Wert von 1.2em und einer Schriftgröße von 15px, hätten wir einen Zeilenabstand von 3px.

```

1 | h1
2 | {
3 |     text-align: center;
4 | }
5 |
6 | p
7 | {
8 |     text-indent: 25px;
9 |     line-height: 1.5em;
10 | }
11 |
12 | b
13 | {
14 |     white-space: nowrap;
15 | }
```



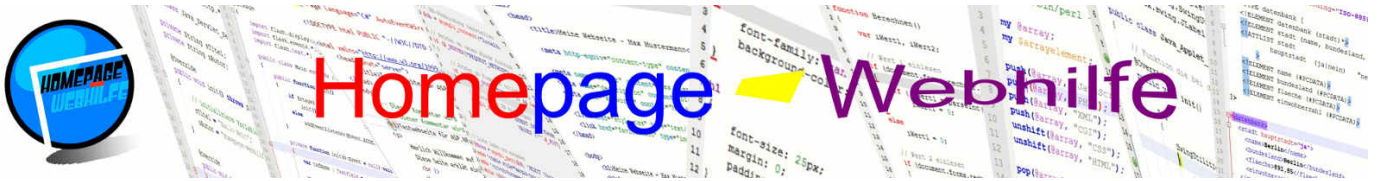
### Textschatten

Um einem Text einen Schatten hinzuzufügen, gibt es die Eigenschaft `text-shadow`. Als Wert müssen 2 bis 4 Parameter angegeben werden. Der erste Parameter legt die horizontale Position und der zweite Parameter legt die vertikale Position fest. Der dritte Parameter gibt den sogenannten „Blur Radius“ an, bei welchem es sich um den Radius für das **Weichzeichnen** handelt. Der vierte und letzte Parameter gibt die Farbe an. Hier sind neben Farbnamen natürlich auch wieder RGB- und HSL-Angaben möglich. Möchten Sie einem Text mehrere Schatten zuweisen, so können Sie nach den 2 bis 4 Parametern ein Komma notieren. Nach dem Komma werden dann wieder 2 bis 4 Parameter angegeben, welche sich jedoch dann auf den zweiten Schatten beziehen.

```

1 | h1
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Schrift](#)

```
2 {  
3   text-shadow: 6px 2px lime;  
4 }  
5  
6 p  
7 {  
8   text-shadow: 2px 2px 5px red;  
9 }
```



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Hintergrund](#)

## Hintergrund

Bei Hintergründen in CSS unterscheidet man grundsätzlich zwischen Hintergrundfarbe und Hintergrundbild. Hintergründe können auf alle Elemente angewendet werden.

### Inhalt dieser Seite:

1. Farbe
2. Bild

### Farbe

Um einem Element eine Hintergrundfarbe zu geben, gibt es die Eigenschaft `background-color`. Hier können Farben in allen bekannten Varianten angegeben werden.

```

1  h1
2  {
3      background-color: red;
4  }
5
6  p
7  {
8      background-color: lime;
9  }
```



### Bild

Um einem Element ein Hintergrundbild zuzuweisen, gibt es die Eigenschaft `background-image`. Als Wert wird das Schlüsselwort `url` gefolgt von einem runden Klammersymbol angegeben. Innerhalb der Klammern wird die URL des Bilds in Anführungszeichen angegeben.

```

1  body
2  {
3      background-image: url("/Bilder/Logo/Logo.png");
4  }
```



Um das Hintergrundbild noch genauer zu spezifizieren, gibt es einige Eigenschaften, welche wir Ihnen hier nun vorstellen werden. Über die Eigenschaft `background-size` lässt sich die **Größe des Bilds** festlegen. Als erstes wird die Breite und als zweites die Höhe angegeben. An Stelle einer Zahl mit Einheit kann auch das Schlüsselwort `auto` notiert werden. Dadurch wird der Wert automatisch ermittelt. Wird nur der 1. Wert angegeben, so entspricht der 2. Wert automatisch dem Wert `auto`.

Normalerweise wird das Hintergrundbild horizontal und vertikal **wiederholt**. Dies kann über die Eigenschaft `background-repeat` geändert werden. Als Werte sind `repeat` (horizontale und vertikale Wiederholung), `repeat-x` (horizontale Wiederholung), `repeat-y` (vertikale Wiederholung) und `no-repeat` (keine Wiederholung) möglich.

Mit Hilfe der Eigenschaft `background-position` lässt sich die **Positionierung** des Startbilds (1. Bild) angeben. Hier ist eine relative, aber auch eine absolute Positionierung möglich. Zudem kann eine Positionierung mittels den Schlüsselwörtern `left`, `right` und `center` für die horizontale Positionierung und `top`, `bottom` und `center` für die vertikale Positionierung erfolgen. Bei der Notation muss zuerst die horizontale Position und anschließend die vertikale Position angegeben werden. Es ist auch möglich, einen der Werte wegzulassen. Dadurch wird der andere Wert automatisch auf `center` gesetzt.

```

1  body
2  {
3      background-image: url("/Bilder/Logo/Logo.png");
4      background-size: 50px;
5      background-position: center;
6      background-repeat: repeat-y;
7  }
```



**Übrigens:** Wenn Sie ein Hintergrundbild verwenden, heißt das noch lange nicht, dass die Angabe einer Hintergrundfarbe unzulässig ist. Im Gegenteil: Oft wird beim Hintergrund auch eine **Kombination von Hintergrundfarbe und Hintergrundbild** verwendet. Nützlich ist dies, wenn sich das Hintergrundbild nicht über die ganze Seite erstreckt, das Hintergrundbild transparent ist oder das Hintergrundbild nicht geladen werden kann.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen

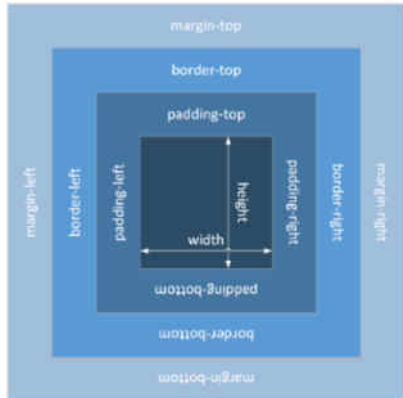


Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Box-Modell](#)

## Box-Modell



In CSS gibt es das sogenannte Box-Modell. Das Box-Modell stellt ein Modell dar, welches die **visuellen Bestandteile eines Elements** widerspiegelt. Eine Box in CSS besteht aus 4 Teilen: Inhalt, Innenabstand, Rahmen und Außenabstand. Die Größe des Inhalts kann bei Block-Elementen über die Eigenschaften `width` und `height` festgelegt werden. Bei Inline-Elementen wird die Breite und Höhe automatisch bestimmt und kann nicht festgelegt werden.

**Inhalt dieser Seite:**

1. Außenabstand
2. Rahmen
3. Innenabstand
4. Box-Modell-Variationen

Auf Grund des Box-Modells ist erkennbar, dass Elemente unter Umständen einen größeren Platzbedarf haben, als in der `width`- und `height`-Eigenschaft festgelegt ist. Die totale Breite setzt sich aus der Breite des linken Außenabstands, der Breite des linken Rahmens, der Breite des linken Innenabstands, der Breite des Inhalts, der Breite des rechten Innenabstands, der Breite des rechten Rahmens und der Breite des rechten Außenabstands zusammen. Die totale Höhe setzt sich aus der Höhe des oberen Außenabstands, der Höhe des oberen Rahmens, der Höhe des oberen Innenabstands, der Höhe des Inhalts, der Höhe des unteren Innenabstands, der Höhe des unteren Rahmens und der Höhe des unteren Außenabstands zusammen.

### Außenabstand

Der Außenabstand stellt den **Abstand zwischen anderen Elementen und dem Rahmen des Elements** dar. Spezifiziert wird der Außenabstand mittels der Eigenschaften `margin`, `margin-left` (linker Abstand), `margin-right` (rechter Abstand) und `margin-bottom` (unterer Abstand). Die Angabe erfolgt in den bekannten Einheiten.

Die Eigenschaft `margin` ist als **universelle Angabe** zu betrachten. Hier können die Abstände auf 4 verschiedene Arten notiert werden. Wird nur ein Wert angegeben, so gilt der Abstand für alle Seiten. Werden zwei Parameter angegeben, so gilt der erste Parameter für oben und unten und der zweite für links und rechts. Bei der Notation von drei Parametern gilt der erste für oben, der zweite für links und rechts und der dritte für unten. Werden alle vier Parameter angegeben, so gilt die Reihenfolge oben, rechts, unten, links.

Wie Sie also sehen, kann mittels der `margin`-Eigenschaft alles festgelegt werden, was mit den einzelnen Eigenschaften `margin-left`, `margin-top`, `margin-right` und `margin-bottom` möglich ist. Die Notation mittels `margin` ist kürzer, hingegen ist die Angabe der einzelnen Eigenschaften einfacher zu lesen, vor allem wenn Sie für alle vier Seiten unterschiedliche Angaben haben. Welche der zwei Varianten Sie nutzen, entscheiden Sie also am besten je nach Situation.

```

1  body
2  {
3      background-color: blue;
4  }
5
6  p
7  {
8      background-color: red;
9      margin: 25px 50px;
10 }
```



### Rahmen

Um unserem Element einen Rahmen zu geben, gibt es die Eigenschaften `border`. Auch hier gibt es wieder die Eigenschaften, welche sich auf eine bestimmte Seite spezifizieren: `border-left`, `border-top`, `border-right` und `border-bottom`. Der Unterschied besteht jedoch darin, dass in der `border`-Eigenschaft immer nur ein Rahmen angegeben werden kann, welcher dann für alle Seiten gilt. Wollen wir also z. B. für links und rechts einen anderen Rahmen haben als für oben und unten, so müssen wir `border-left`, `border-right`, `border-top` und `border-bottom` notieren. Innerhalb der verschiedenen `border`-Eigenschaften können bis zu 3 Parameter angegeben werden: Breite (Zahl mit Einheit), Stil und Farbe. Eine Auflistung der verschiedenen Rahmen-Stile sehen Sie in der untenstehenden Tabelle:

<b>none</b>	kein Rahmen
<b>solid</b>	durchgezogener Rahmen
<b>dashed</b>	gestrichelter Rahmen
<b>dotted</b>	gepunkteter Rahmen
<b>double</b>	doppelter durchgezogener Rahmen
<b>ridge</b>	3D Rahmen (geteilt)
<b>groove</b>	3D Rahmen (geteilt)
<b>inset</b>	3D Rahmen ("nach innen gelegt")
<b>outset</b>	3D Rahmen ("nach außen gelegt")

Des Weiteren gibt es für alle Rahmen-Eigenschaften die Erweiterungen `-width`, `-style` und `-color`. So können wir dann z. B. mit der Eigenschaft `border-right-style` den Stil für den Rahmen auf der rechten Seite festlegen. Diese Erweiterungen können aber nicht nur in Verbindung mit den seitenspezifischen Eigenschaften, sondern auch mit der allgemeinen Eigenschaft `border` verwendet werden: z. B. `border-width`. Für die daraus entstandenen Eigenschaften `border-width`, `border-style` und `border-color` gilt dieselbe Regel, wie auch für `margin`: Es sind 1 bis 4 Angaben möglich. Es handelt sich also wieder um universelle Eigenschaften.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Box-Modell](#)

```

1  body
2  {
3      background-color: blue;
4  }
5
6  p
7  {
8      background-color: red;
9      margin: 25px 50px;
10     padding: 10px;
11     border: 5px dotted lime;
12 }

```

Ab CSS3 ist es möglich, **Ecken abzurunden**. Hierzu dienen die Eigenschaften `border-radius`, `border-top-left-radius`, `border-top-right-radius`, `border-bottom-right-radius` und `border-bottom-left-radius`. `border-radius` ist als universelle Eigenschaft anzusehen und ermöglicht die Angabe von 1 bis 4 Parametern. Wird ein Parameter angegeben so gilt dieser für alle Ecken. Bei zwei Parametern gilt der erste für links oben und der zweite für rechts unten. Bei drei Parametern gilt die Reihenfolge oben links, oben rechts und unten links und zum Schluss unten rechts. Bei vier Parametern erfolgt die Angabe im Uhrzeigersinn, angefangen bei der Ecke oben links.

```

1  body
2  {
3      background-color: blue;
4  }
5
6  p
7  {
8      background-color: red;
9      margin: 25px 50px;
10     padding: 10px;
11     border: 5px dotted lime;
12     border-radius: 10px;
13 }

```

## Innenabstand

Die Angabe des Innenabstands, also des **Abstands zwischen Rahmen und Inhalt**, erfolgt mittels der Eigenschaft `padding`. Auch hier gibt es die seitenspezifischen Angaben `padding-left`, `padding-top`, `padding-right` und `padding-bottom`. `padding` ist, sowie `margin` auch, als universelle Eigenschaft anzusehen. Bei der Notation sind die vier genannten Möglichkeiten (siehe [Außenabstand](#)) erlaubt.

```

1  body
2  {
3      background-color: blue;
4  }
5
6  p
7  {
8      background-color: red;
9      margin: 25px 50px;
10     border: 5px dotted lime;
11     padding: 20px 40px;
12 }

```

## Box-Modell-Variationen

Das bisher starre Box-Modell kann seit CSS3 über die Eigenschaft `box-sizing` verändert werden. Die Voreinstellung ist `box-sizing` und führt dazu, dass sich die angegebene Breite und Höhe lediglich auf den Inhalt beziehen, d. h. durch die zusätzliche Angabe von Außenabstand, Rahmen und Innenabstand vergrößert sich der totale Platzbedarf des Elements. Über den Wert `border-box` lässt sich **das Box-Modell so anpassen**, dass sich die Breiten- und Höhenangabe auf Inhalt, Innenabstand und Rahmen beziehen, d. h. lediglich der Außenabstand vergrößert den tatsächlichen Platzbedarf. Bei Verwendung des Werts `border-box` sollte darauf geachtet werden, dass von der angegebenen Größe der Rahmen und Innenabstand abgezogen werden muss, wenn man wissen möchte, wie viel Platz dem eigentlichen Inhalt zur Verfügung steht.

```

1  h1, h2
2  {
3      background-color: blue;
4      width: 500px;
5      margin: 20px;
6      padding: 20px;
7      border: 10px solid black;
8      text-align: center;
9  }
10
11 h1
12 {

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Box-Modell](#)

```
13     box-sizing: content-box;
14 }
15
16 h2
17 {
18     box-sizing: border-box;
19 }
```



**Übrigens:** Der Hintergrund, welcher mit `background-color` oder `background-image` spezifiziert ist, erstreckt sich bei Elementen immer nur über den Rahmen, den Innenabstand und den Inhalt, nicht aber über den Außenabstand.

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Layout](#)

## Layout

Elemente können in CSS auf verschiedene Arten positioniert werden: statisch, relativ, absolut und fixiert. Das Webseiten-Layout selbst wird zumeist über statische Positionierung gestaltet. Einige Bestandteile einer Seite (wie z. B. Kopf- und Fußzeile) können aber durchaus absolut oder fixiert positioniert werden. Um die Art der Positionierung festzulegen, gibt es die Eigenschaft `position`.

### Inhalt dieser Seite:

1. Statische Positionierung
2. Absolute Positionierung
3. Überlappung und Überlauf

## Statische Positionierung

Die statische Positionierung kann über den Wert `static` festgelegt werden. Die Angabe dieser Positionierungsart ist jedoch zumeist nicht notwendig, da `static` die Voreinstellung ist. Eine statische Positionierung bedeutet, dass die Elemente so positioniert sind wie diese im HTML-Code notiert sind. Die Positionierung erfolgt also untereinander oder nebeneinander (je nach Typ und Größe). Zur statischen Positionierung gibt es neben der `margin`-Eigenschaft, welche Sie bereits kennengelernt haben und einen Abstand zu anderen Elementen herstellt, noch die `float`-Eigenschaft. Mit der `float`-Eigenschaft lassen sich **Elemente fließend platzieren**. Diese Eigenschaft wird unter anderem gerne dazu genutzt, ein Bild in einen Textfluss einzubinden. Als Werte für `float` sind `none` (kein Umfluss), `left` (Positionierung links) und `right` (Positionierung rechts) möglich.

```

1  img
2  {
3      float: left;
4      margin-right: 10px;
5      margin-bottom: 5px;
6  }
```

Neben der Einbindung von Elementen in einen Textfluss kann die `float`-Eigenschaft auch einfach nur zur Positionierung genutzt werden. Dieses Verfahren wird z. B. verwendet, wenn wir ein **mehrspaltiges Layout** erstellen wollen. Werden mehrere Elemente links oder rechts positioniert (so wie auch im Beispiel), dann muss auf die Notationsreihenfolge im HTML-Code geachtet werden. Bei Elementen die links platziert werden sollen, wird das erste Element ganz links platziert. Bei Elementen die rechts platziert werden sollen, wird das zuerst notierte Element ganz rechts platziert.

```

1  div
2  {
3      width: 200px;
4      height: 200px;
5  }
6
7  #links
8  {
9      float: left;
10     background-color: red;
11 }
12
13 #links2
14 {
15     float: left;
16     background-color: lime;
17 }
18
19 #rechts
20 {
21     float: right;
22     background-color: blue;
23 }
24
25 #rechts2
26 {
27     float: right;
28     background-color: yellow;
29 }
```

Immer wieder kann es vorkommen, dass Sie einen mit der `float`-Eigenschaft erstellten **Umfluss beenden** wollen. Ein typisches Beispiel ist folgendes: Sie haben ein Bild auf der linken Seite und einen Text auf der rechten Seite. Darunter sollen nun eine Überschrift und ein weiterer Text folgen. Das Bild ist aber höher als der erste Text. Nun haben wir das Problem, dass die Überschrift und der zweite Text direkt unter dem ersten Text platziert werden. Dies ist aber nicht gewünscht. Um dieses Problem zu lösen, können wir die `clear`-Eigenschaft verwenden. Als Werte für die `clear`-Eigenschaft sind `none` (Voreinstellung), `left`, `right` und `both` möglich. `left` bewirkt, dass das Element unterhalb des linken Elements positioniert wird. `right` bewirkt das gleiche nur eben für die rechte Seite. Mit `both` wird sichergestellt, dass sich das Element unterhalb beider / aller Elemente befindet. Da dies in den meisten Fällen erwünscht ist, wird `both` als Wert zumeist bevorzugt.

```

1  
2  <p>...</p>
3  <div>Beispiel zur statischen Positionierung von Elementen in CSS aus dem Tutorial von Homepage-Webhilfe</div>
4
1  img
2  {
3      float: left;
4      width: 250px;
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Layout](#)

```

5  }
6
7  p
8  {
9      float: left;
10     width: 490px;
11     margin: 0 0 0 10px;
12 }
13
14 div
15 {
16     clear: both;
17     width: 750px;
18     padding: 10px 0 0;
19     text-align: center;
20     font-style: italic;
21 }

```



## Absolute Positionierung

Über den Wert `absolute` in der `position`-Eigenschaft können wir ein Element absolut positionieren. Die Positionierung erfolgt dabei über die Eigenschaften `left`, `top`, `right` und `bottom`. Die gleichzeitige Verwendung der Eigenschaften `top` und `bottom` und der Eigenschaften `left` und `right` ist nicht möglich. Die absolute Positionierung erfolgt standardmäßig **in Bezug auf die ganze Seite** (body-Element) und nicht in Bezug auf das Eltern-Element. Wollen wir die Elemente absolut in **Relation zum Eltern-Element** platzieren, so können wir dem Eltern-Element den Wert `relative` in der `position`-Eigenschaft zuweisen.

```

1  div
2  {
3      position: absolute;
4      width: 200px;
5      height: 200px;
6  }
7
8  #box1
9  {
10     top: 50px;
11     background-color: red;
12 }
13
14 #box2
15 {
16     left: 300px;
17     background-color: lime;
18 }
19
20 #box3
21 {
22     right: 50px;
23     background-color: blue;
24 }
25
26 #box4
27 {
28     bottom: 150px;
29     right: 0px;
30     background-color: yellow;
31 }

```



Bei der fixierten Positionierung (`position="fixed"`) erfolgt die Positionierung wie bei der absoluten Positionierung nur mit dem Unterschied, dass als **Bezugspunkt das Browserfenster** verwendet wird. Deshalb ist diese Positionierung z. B. ideal für „fixierte Fußzeilen“.

## Überlappung und Überlauf

Haben wir mehrere Elemente die sich gegenseitig überlappen, wird im Vordergrund das zuletzt notierte Element (im HTML-Code) angezeigt. Um diese sogenannte **Stapelreihenfolge** zu ändern, gibt es in CSS die Eigenschaft `z-index`. Als Wert der Eigenschaft wird eine Zahl (ohne Einheit) angegeben. Umso größer die Zahl ist, umso weiter im Vordergrund befindet sich das Element. Die `z-index` Eigenschaft kann nicht bei statisch angeordneten Elementen eingesetzt werden.

```

1  div
2  {
3      position: absolute;
4      width: 200px;
5      height: 200px;
6  }
7
8  #box1
9  {

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Layout](#)

```

10     z-index: 1;
11     background-color: red;
12 }
13
14 #box2
15 {
16     top: 50px;
17     left: 50px;
18     z-index: 0;
19     background-color: lime;
20 }
```

Über die Eigenschaft `overflow` lässt sich der sogenannte Überlauf steuern. Ein Überlauf entsteht, wenn der gewünschte Inhalt nicht mehr in das Element hineinpasst. Dies kann natürlich nur passieren, wenn dem Element eine  **feste Höhe**  zugewiesen ist. Als Werte für die `overflow`-Eigenschaft stehen folgende Werte zur Verfügung:

<b>visible</b>	Der Überlauf wird angezeigt, jedoch außerhalb des Elements.
<b>hidden</b>	Der Überlauf wird nicht angezeigt.
<b>auto</b>	Der Überlauf wird im Element angezeigt. Hierfür werden falls benötigt Scrollbalken angezeigt. <i>(Standardeinstellung)</i>
<b>scroll</b>	Der Überlauf wird im Element angezeigt. Das Element verfügt immer über beide Scrollbalken.

```

1  p
2  {
3      width: 250px;
4      height: 150px;
5      margin-bottom: 50px;
6      border: 1px solid black;
7  }
8
9  #text1
10 {
11     overflow: visible;
12 }
13
14 #text2
15 {
16     overflow: scroll;
17 }
18
19 #text3
20 {
21     overflow: hidden;
22 }
23
24 #text4
25 {
26     overflow: auto;
27 }
```

Um den Überlauf nur für die X- oder Y-Achse zu steuern kann, an Stelle der `overflow`-Eigenschaft, die Eigenschaft `overflow-x` und `overflow-y` eingesetzt werden. Die dort verfügbaren Werte entsprechen der `overflow`-Eigenschaft.

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » CSS » Effekte

## Effekte

Mit CSS ist es möglich, einige „Effekte“ zu erzeugen, welche wir Ihnen auf dieser Seite vorstellen möchten.

- Inhalt dieser Seite:**
1. Schatten
  2. Transparenz

## Schatten

Nicht nur für Text ist es möglich, einen **Schatten** zu erstellen (`text-shadow`), sondern auch für Elemente. Diesen sogenannten Box-Schatten können wir mittels der `box-shadow`-Eigenschaft festlegen. Als Wert werden 2 bis 5 Parameter angegeben. Der erste Parameter gibt den horizontalen Versatz an und der zweite den vertikalen Versatz. Mit dem dritten und somit dem ersten optionalen Parameter wird der Radius des Weichzeichens angegeben. Der vierte Parameter dient zur Angabe der Größe des Schattens. Mit Hilfe des 5. und letzten Parameters wird die Farbe des Schattens festgelegt, wovon die Standardeinstellung schwarz ist.

```

1  div
2  {
3      background-color: red;
4      width: 200px;
5      height: 200px;
6      box-shadow: 20px 10px 25px blue;
7  }

```



## Transparenz

Mit Hilfe der Transparenz (Durchsichtigkeit) ist es möglich, die **Deckkraft** eines Elements zu verringern, wodurch das dahinterliegende Element oder der Hintergrund des Eltern-Elements durchschimmert. Die Deckkraft kann mittels der Eigenschaft `opacity` festgelegt werden. Die Angabe erfolgt in einer Fließkommazahl (ohne Einheit) mit Werten zwischen 0 (für keine Deckkraft) und 1 (für volle Deckkraft).

Wie Sie in der Vorschau des untenstehenden Beispiels bemerken werden, wird bei der Verwendung der `opacity`-Eigenschaft nicht nur der Hintergrund des aktuellen Elements durchsichtig, sondern zugleich auch die Schrift. Dieser Effekt ist nicht immer erwünscht, weshalb oft an Stelle der `opacity`-Eigenschaft der Wert der **Hintergrundfarbe** angepasst wird, denn durch die Angabe eines RGBA- oder HSLA-Farbwerts an Stelle eines RGB- oder HSL-Farbwerts kann der Hintergrund eines Elements ebenfalls transparent gemacht werden. Das untenstehende Beispiel enthält deshalb zwei Beispiele, bei welchem die beiden genannten Varianten direkt verglichen werden können.

```

1  body
2  {
3      background-color: lightgray;
4  }
5
6  #text1, #text2
7  {
8      color: yellow;
9      width: 200px;
10     padding: 5px;
11 }
12
13 #text1
14 {
15     background-color: blue;
16     opacity: 0.5;
17 }
18
19 #text2
20 {
21     background-color: rgba(0, 0, 255, 0.5);
22 }

```



<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--





Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Transformationen](#)

## Transformationen

In CSS ist es möglich, mittels der Eigenschaft `transform` Elemente zu transformieren. Hierzu zählt verschieben, skalieren, neigen und rotieren.

**Inhalt dieser Seite:**

1. Verschieben
2. Skalieren
3. Neigen
4. Rotieren

### Verschieben

Um Elemente zu verschieben benötigen wir als Wert der `transform`-Eigenschaft das Schlüsselwort `translate`, welches von einem runden Klammernpaar gefolgt wird. Innerhalb der Klammer werden zwei Parameter angegeben: die X-Verschiebung und die Y-Verschiebung. Beide Parameter müssen als Zahl zusammen mit einer Einheit (außer der Wert ist 0) angegeben werden. Getrennt werden die zwei Parameter durch ein Komma.

```

1  div
2  {
3      background-color: red;
4      width: 200px;
5      height: 200px;
6      transform: translate(100px, 25px);
7  }
```



### Skalieren

Über das Schlüsselwort `scale` lassen sich Elemente skalieren. Auch hier werden ein Klammernpaar und zwei Werte als Parameter benötigt. Die Parameter werden jedoch hier nicht zusammen mit einer Einheit angegeben. Es wird also nur eine Zahl angegeben, welche den **Skalierungsfaktor** angibt (z. B. 0.5 = 50%, 1.5 = 150%, 2.0 = 200% usw.).

Bei der Skalierung muss beachtet werden, dass das Element **aus dem Mittelpunkt heraus vergrößert bzw. verkleinert** wird. Geändert werden kann dies mittels der `transform-origin`-Eigenschaft, welche wir weiter unten beim Unterthema [Rotieren](#) genauer erklären werden.

```

1  div
2  {
3      background-color: red;
4      width: 200px;
5      height: 200px;
6      transform: scale(1.4, 0.8);
7  }
```



### Neigen

Um ein Element zu neigen, notieren wir in der `transform`-Eigenschaft das Schlüsselwort `skew`. Als Parameter werden wieder zwei Werte in der Einheit `deg` (`degree = Grad`) angegeben. Der erste Wert bestimmt die X-Neigung und der zweite die Y-Neigung.

```

1  div
2  {
3      background-color: red;
4      width: 200px;
5      height: 200px;
6      transform: skew(20deg, 10deg);
7  }
```



**Übrigens:** Die Schlüsselwörter `translate`, `scale` und `skew` können um das Zeichen X oder Y erweitert werden (z. B. `translateX` oder `scaleY`). Innerhalb der Klammern wird dann lediglich ein Parameter angegeben, welcher sich auf die X- oder Y-Achse auswirkt. Eine Notation wie z. B. `skew(30deg, 0)` könnte dann also durch `skewX(30deg)` ersetzt werden.

### Rotieren

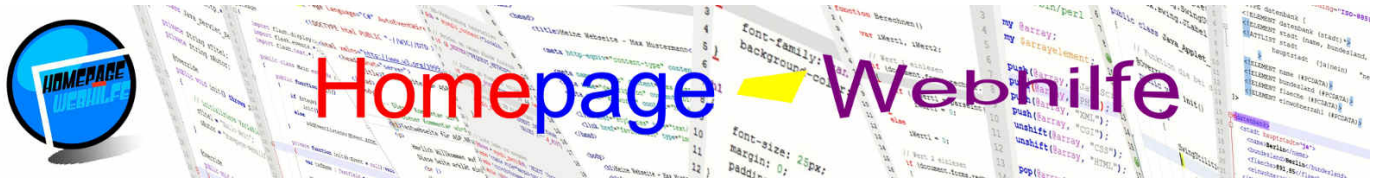
Über das Schlüsselwort `rotate` lassen sich Elemente rotieren. Innerhalb der Klammer wird ein Wert in der `deg`-Einheit (Grad) angegeben. Eine Angabe wie `-10deg` ist erlaubt, könnte aber natürlich auch durch `350deg` ersetzt werden.

Die `transform-origin`-Eigenschaft legt (bei der Rotation) den **Drehpunkt** fest. Als Wert für die `transform-origin`-Eigenschaft werden zwei Parameter angegeben, wovon sich der erste auf die X-Achse und der zweite auf die Y-Achse bezieht. Die Parameter können sowohl den Wert `left`, `center` oder `right` (für die X-Achse) und `top`, `center` oder `bottom` (für die Y-Achse) besitzen, als auch eine Angabe in den CSS bekannten Einheiten (Pixel, Prozent etc.). Die Eigenschaft `transform-origin` kann auch in Zusammenhang mit der Transformation zur Skalierung und Neigung verwendet werden. Dadurch ist es möglich, eine Transformation nicht von der Mitte aus durchzuführen, sondern von einer anderen Position.

```

1  div
2  {
3      background-color: red;
4      width: 200px;
5      height: 200px;
6      transform: rotate(20deg);
7      transform-origin: top left;
8  }
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Transformationen](#)



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

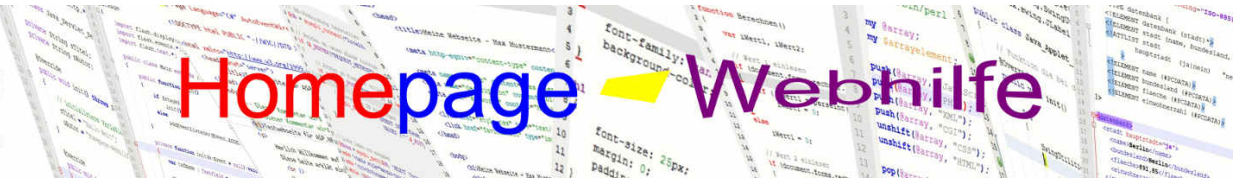
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Pseudoklassen](#)

## Pseudoklassen

Pseudoklassen sind Erweiterungen für Selektoren. Sie werden direkt hinter dem Selektor angegeben. Die Namen von Pseudoklassen sind festgelegt und beginnen immer mit einem Doppelpunkt `:`. Pseudoklassen dienen dazu, einen Regelblock auf ein Element nur dann anzuwenden, wenn eine bestimmte Eigenschaft (dabei ist nicht die CSS-Eigenschaft gemeint) gegeben ist.

**Inhalt dieser Seite:**

1. Hover-Effekt
2. Fokussierung
3. Links
4. Elemente
5. Formulare

### Hover-Effekt

Das bekannteste und beliebteste Beispiel für Pseudoklassen ist der Hover-Effekt. Stellen Sie sich vor, Sie haben ein `div`-Element mit roter Hintergrundfarbe und sobald Sie Ihren Mauszeiger innerhalb des `div`-Elements haben, soll die Hintergrundfarbe geändert werden. Eine solche Funktionalität ist in „reinem“ CSS und somit **ohne aktive Skriptsprachen** wie JavaScript möglich. Der Hover-Effekt kann in CSS mittels der Pseudoklasse `:hover` angesprochen werden. Typische Beispiele für den Hover-Effekt sind ausklappbare Menüs und die Hervorhebung von Tabellenzeilen. Die Pseudoklasse `:hover` sollte jedoch mit Vorsicht verwendet werden, da auf Geräten mit Touchscreen bzw. Geräten ohne Maus kein Hover-Effekt zur Verfügung steht.

Ein weiterer ähnlicher Effekt wie `:hover` können wir mit `:active` erreichen. `:active` tritt ein, wenn die Maus bzw. der Cursor sich auf einem Element befindet und die **Maustaste gedrückt** gehalten wird. Man spricht davon, dass das Element aktiv ist (daher auch der Name der Pseudoklasse). Im unteren Beispiel mit dem `div`-Element wird also die Hintergrundfarbe auf hellgrün geändert, sobald sich Ihr Cursor innerhalb des Elements befindet und Sie die Maustaste gedrückt halten.

```

1  div
2  {
3      background-color: red;
4      width: 200px;
5      height: 200px;
6  }
7
8  div:hover
9  {
10     background-color: blue;
11 }
12
13 div:active
14 {
15     background-color: lime;
16 }
```



### Fokussierung

Ein paar HTML-Elemente (z. B. `input`, `select` und `textarea`) können den sogenannten **Tastaturfokus** erhalten. Bei Elementen, die den Tastaturfokus erhalten können, wird beim „Durchschalten“ mittels Tab angehalten. Natürlich kann der Tastaturfokus auch manuell über das Klicken in ein anderes Element (z. B. Textfeld) mittels Maus oder Touchscreen gesetzt werden. Für die Fokussierung durch den Browser gibt es die Pseudoklasse `:focus`. Dadurch ist es z. B. möglich, das aktuell gewählte Textfeld zu markieren (siehe Beispiel).

```

1  body
2  {
3      line-height: 1.8em;
4  }
5
6  input
7  {
8      border: 1px solid red;
9  }
10
11 input:focus
12 {
13     border: 1px solid blue;
14     border-radius: 5px;
15 }
```



### Links

Neben den Pseudoklassen `:hover`, `:active` und `:focus` gibt es für Links zudem noch die Pseudoklassen `:link` (für nicht besuchte Links) und `:visited` (für besuchte Links). Mit Hilfe dieser Pseudoklassen ist es nun endlich möglich, das Standardverhalten von Browsern in Bezug auf die Darstellung von Links zu ändern. Dabei ist auf die Reihenfolge der Notation zu achten: `:link`, `:visited`, `:focus`, `:hover`, `:active`. Natürlich ist es auch möglich, eine oder mehrere der Pseudoklassen wegzulassen. Um Links zu formatieren, kommen hauptsächlich die CSS-Eigenschaften `color` und `text-decoration` zum Zuge.

```

1  a:link
2  {
3      color: red;
4  }
5
6  a:visited
7  {
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Pseudoklassen](#)

```

8     color: orange;
9 }
10
11 a:focus
12 {
13     color: lime;
14 }
15
16 a:hover
17 {
18     color: blue;
19 }
20
21 a:active
22 {
23     color: black;
24 }
    
```

**Übrigens:** In Navigationsleisten möchte man oft Links immer gleich anzeigen, egal ob diese bereits besucht wurden, noch nicht besucht wurden oder fokussiert sind. Dies kann z. B. mittels folgendem simplen CSS-Regelblock erfolgen:

```

1 a
2 {
3     color: red;
4 }
    
```

## Elemente

Einige Pseudoklassen sind in der Lage, bestimmte Elemente zu selektieren (z. B. das erste oder letzte Element). Eine Liste dieser sogenannten strukturellen Pseudoklassen finden Sie in folgender Tabelle:

<b>p:first-child</b>	Selektiert das erste Element, sofern es sich um ein p-Element handelt.
<b>p:last-child</b>	Selektiert das letzte Element, sofern es sich um ein p-Element handelt.
<b>p:nth-child</b>	Selektiert das n-te Element, sofern es sich um ein p-Element handelt.
<b>p:nth-last-child</b>	Selektiert das n-te Element, sofern es sich um ein p-Element handelt (Zählung beginnt von unten).
<b>p:first-of-type</b>	Selektiert das erste p-Element.
<b>p:last-of-type</b>	Selektiert das letzte p-Element.
<b>p:nth-of-type</b>	Selektiert das n-te p-Element.
<b>p:nth-last-of-type</b>	Selektiert das n-te p-Element (Zählung beginnt von unten).

Die Selektierung eines **n-ten Elements** kann über verschiedene Art und Weisen erfolgen. Hierfür muss immer hinter dem Namen der Pseudoklasse ein rundes Klammernpaar notiert werden. Für den Wert innerhalb der Klammern gibt es nun verschiedene Möglichkeiten: Wird eine einzelne Zahl angegeben, so wird ein **einzelnes Element** angesprochen. Bei der angegebenen Zahl handelt es sich um die sogenannte **Platznummer**. Das erste Element ist die Nummer 1, das zweite die Nummer 2 usw.. Die Angabe einer Zahl, gefolgt von dem Buchstaben **n**, führt dazu, dass jedes n-te Element selektiert wird, d. h. die Angabe **2n** selektiert jedes 2te Element. Diese Angabe kann noch um ein Pluszeichen **+** gefolgt von einer weiteren Zahl bei Bedarf erweitert werden. Diese zweite Zahl ist als **Offset** anzusehen, d. h. eine Angabe wie **2n+3** selektiert jedes 2te Element ab dem dritten Element. Da es oft vorkommt, dass man jedes zweite Element selektieren möchte, wurden in CSS die Schlüsselwörter **odd** und **even** eingeführt. Die Angabe **odd** entspricht **2n+1** und **even** entspricht der Angabe **2n**.

Im Beispiel unten könnten die Pseudoklassen **:first-child** durch **:first-of-type**, **:last-child** durch **:last-of-type** usw. ersetzt werden. Dies ist jedoch nur möglich, da innerhalb des Elternelements (in diesem Fall das **body**-Element) nur **p**-Elemente vorhanden sind. Wäre das erste Element nun z. B. ein Bild, so würde die Regel **p:first-child** nicht mehr in Kraft treten. Würden wir diese jedoch durch **p:first-of-type** ersetzen, dann würde der Regelblock wieder eintreten.

```

1 p:first-child
2 {
3     text-indent: 10px;
4 }
5
6 p:nth-child(2n+5)
7 {
8     color: red;
9 }
10
11 p:nth-child(3)
12 {
13     color: blue;
14 }
15
16 p:last-child
17 {
18     padding-top: 15px;
    
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Pseudoklassen](#)

```
19 | border-top: 1px solid black;
20 | }
```



## Formulare

Für Formulare gibt es noch weitere Pseudoklassen, mit welchen es möglich ist, Formularfelder je nach deren Werten bzw. deren Status zu selektieren. Die verfügbaren Pseudoklassen sind in der untenstehenden Tabelle aufgelistet.

<b>:checked</b>	Selektiert Checkboxes und Auswahlguppen (Radio-Buttons), welche ausgewählt sind.
<b>:disabled</b>	Selektiert deaktivierte Felder.
<b>:enabled</b>	Selektiert aktivierte Felder.
<b>:in-range</b>	Selektiert Felder, bei welchen der Wert innerhalb des Gültigkeitsbereichs liegt.
<b>:out-of-range</b>	Selektiert Felder, bei welchen der Wert außerhalb des Gültigkeitsbereichs liegt.
<b>:invalid</b>	Selektiert Felder, deren Eingabe ungültig ist.
<b>:valid</b>	Selektiert Felder, deren Eingabe gültig ist.
<b>:required</b>	Selektiert benötigte Felder.
<b>:optional</b>	Selektiert optionale Felder.
<b>:read-only</b>	Selektiert Felder, welche nur gelesen werden können.
<b>:read-write</b>	Selektiert Felder, welche gelesen und geschrieben werden können.

```
1 | Vorname: <input /><br />
2 | Nachname: <input required="required" /><br />
3 | Betreff: <input /><br />
4 | E-Mail: <input required="required" /><br />
5 | Zahl: <input type="number" min="0" max="9" />
```

```
1 | body
2 | {
3 |   line-height: 1.8em;
4 | }
5 |
6 | input:optional
7 | {
8 |   background-color: #EEEEEE;
9 | }
10 |
11 | input:invalid
12 | {
13 |   border: 3px dotted red;
14 | }
15 |
16 | input:out-of-range
17 | {
18 |   color: red;
19 | }
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Pseudoelemente](#)

## Pseudoelemente

Pseudoelemente sind, wie Pseudoklassen auch, **Erweiterungen** für Selektoren. Mit Hilfe von Pseudoelementen ist es möglich, auf Elemente zuzugreifen, welche nicht direkt im HTML-Code notiert sind, jedoch in der **visuellen Darstellung** sichtbar sind. Gekennzeichnet werden Pseudoelemente mit zwei Doppelpunkten `::` gefolgt von dem Namen des Pseudoelements.

### Inhalt dieser Seite:

1. Davor und danach
2. Buchstaben und Zeilen

### Davor und danach

Mit Hilfe der Pseudoelemente `::before` und `::after` ist es möglich, auf den Inhalt eines Elements vor und nach dem eigentlichen notierten Inhalt zuzugreifen bzw. diesen zu erstellen. Beide Pseudoelemente werden im Zusammenhang mit der Eigenschaft `content` verwendet, mit welcher es möglich ist, einen Inhalt vor oder nach dem Inhalt einzufügen. Die Einfügung erfolgt dabei immer zwischen dem eigentlich notierten Inhalt und dem Start- bzw. Endtag. Innerhalb der Regelblöcke können neben der `content`-Eigenschaft auch noch weitere Eigenschaften notiert werden, welche sich auf den eingefügten Inhalt auswirken.

```

1  p::before
2  {
3      content: "„";
4      color: red;
5  }
6
7  p::after
8  {
9      content: "“";
10     color: red;
11 }
```



### Buchstaben und Zeilen

Mit Hilfe des Pseudoelements `::first-letter` können wir auf den **ersten Buchstaben** eines Textes zugreifen. Das Pseudoelement `::first-line` wirkt sich auf die **erste Zeile** eines Textes aus. Dabei ist nicht die erste Zeile im HTML-Code gemeint, sondern die erste im Browser dargestellte Zeile. Die notierten Eigenschaften im Pseudoelement `::first-letter` können von dem Pseudoelement `::first-line` nicht überschrieben werden.

```

1  p
2  {
3      font-size: 14px;
4  }
5
6  p::first-letter
7  {
8      font-size: 18px;
9      font-weight: bold;
10 }
11
12 p::first-line
13 {
14     font-size: 16px;
15 }
```



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Funktionen](#)

## Funktionen

Funktionen sind vordefinierte bzw. spezielle Werte für CSS-Eigenschaften. Die Angabe erfolgt mit dem Namen der Funktion gefolgt von einem runden Klammernpaar. Innerhalb der Klammern können dann, je nach Funktion, ein oder mehrere Parameter angegeben werden, welche mit Kommas getrennt werden. Sie werden jetzt denken, dass Sie solche Funktionen bereits kennengelernt haben, so wie z. B. `url`, `rgba`, `scale` oder `rotate`. Doch dabei handelt es sich laut Spezifikation nicht um Funktionen, auch wenn diese vom Aufbau identisch sind.

### Inhalt dieser Seite:

1. Berechnung
2. Attribut
3. Farbverlauf

## Berechnung

Die Funktion `calc()` kann dazu genutzt werden, Berechnungen durchzuführen. Die Funktion ist ideal, um **relative und absolute Einheiten zu kombinieren**. Stellen Sie sich vor, Sie möchten ein Bild auf die volle Browserbreite anzeigen. In diesem Fall würde Ihnen vermutlich die Angabe `width: 100%` einfallen. Nun entschließen Sie sich aber, dem Bild einen Abstand von 10px auf allen Seiten zu geben. Nun müssen wir den Wert der Eigenschaft `width` noch ersetzen, da die Angabe von 100% in diesem Moment nicht mehr stimmt und somit einen Scrollbalken erzeugt. An dieser Stelle kann nun die Funktion `calc()` eingesetzt werden. Als Parameter wird eine Formel angegeben. Das untere Beispiel zeigt, wie sich das oben genannte Problem lösen lässt:

```
1  img
2  {
3      width: calc(100% - (2 * 10px));
4      margin: 10px;
5  }
```



## Attribut

Über die Funktion `attr()` können wir auf ein **Attribut des HTML-Elements** zugreifen. Hierfür wird als Parameter der Name des Attributs (ohne Anführungszeichen) angegeben. Genutzt werden kann die Funktion z. B. im Zusammenhang mit der `content`-Eigenschaft.

```
1  a::before
2  {
3      content: attr(title) " ";
4  }
```



**Übrigens:** Um in CSS zwei Zeichenketten (siehe Beispiel) zu kombinieren, notieren wird nicht (wie in anderen Sprachen) ein Pluszeichen.

## Farbverlauf

Einen Farbverlauf in CSS können Sie mittels der Funktionen `linear-gradient` oder `radial-gradient` erstellen.

Die Funktion `linear-gradient` erzeugt einen **linearen Farbverlauf** und erwartet als Parameter Farbwerte, die für den Farbverlauf genutzt werden sollen. Möchten wir zusätzlich die „Schritte“ (sogenannte *color-stops*) und somit die Endpositionen eines Farbverlaufs einer einzelnen Farbe festlegen, können wir nach dem Farbwert, gefolgt von einem Leerzeichen, die Stopp-Position mit einer relativen oder absoluten Einheit angeben. Zusätzlich ist noch die Angabe eines Winkels möglich, in welchem der Verlauf angezeigt werden soll. Dieser muss, sofern er verwendet wird, als 1. Parameter übergeben werden.

Mit Hilfe der Funktion `radial-gradient` ist es möglich, einen **kreis- bzw. ellipsenförmigen Farbverlauf** zu erzeugen. Als Parameter werden auch hier Farbwerte und ggf. die Stopp-Positionen angegeben. Über einen weiteren (optionalen) Parameter, der als 1. Parameter übergeben werden muss, lässt sich die Größe und Positionierung des Farbverlaufs verändern. Als erstes muss dort die Größe (zuerst Breite, dann Höhe) gefolgt von dem Schlüsselwort `at` angegeben werden. Nun muss als nächstes noch die Positionierungs-Angabe folgen. Hier wird zuerst wieder die X-Position und anschließend die Y-Position angegeben. Bei den Angaben sind neben relativen Einheiten (wie im Beispiel), auch absolute Einheiten erlaubt.

```
1  div
2  {
3      width: 200px;
4      height: 200px;
5      margin: 10px;
6  }
7
8  #box1
9  {
10     background: linear-gradient(blue, purple, red);
11 }
12
13 #box2
14 {
15     background: linear-gradient(blue 20%, purple 50%, red 80%);
16 }
17
18 #box3
19 {
20     background: linear-gradient(60deg, blue, purple, red);
21 }
22
23 #box4
24 {
25     background: radial-gradient(blue, purple, red);
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

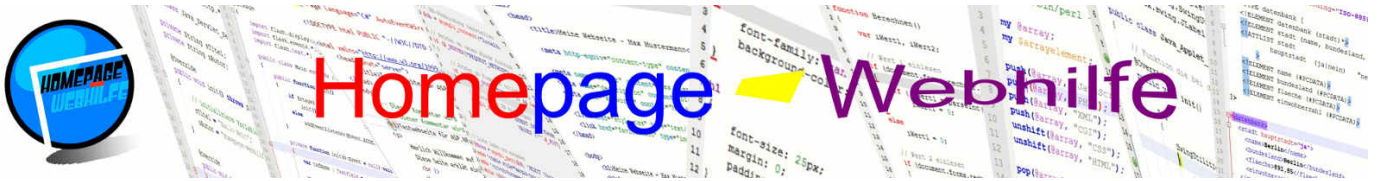
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Funktionen](#)

```
26 | }
27 |
28 | #box5
29 | {
30 |     background: radial-gradient(blue 20%, purple 50%, red 80%);
31 | }
32 |
33 | #box6
34 | {
35 |     background: radial-gradient(80% 60% at 60% 80%, blue, purple, red);
36 | }
```



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung





Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Weitere CSS-Eigenschaften](#)

## Weitere CSS-Eigenschaften

Auf dieser Seite stellen wir ein paar CSS-Eigenschaften vor, welche sich in keine der bisherigen Themen einordnen lässt. Dies kommt zum einen daher, da diese zu keinem der Themen passt oder da es sich um elementspezifische Eigenschaften von HTML-Elementen handelt.

### Inhalt dieser Seite:

1. Cursor
2. Listen
3. Tabellen

### Cursor

Mit Hilfe der Eigenschaft `cursor` lässt sich der Mauszeiger (auch als Cursor bezeichnet) festlegen. Hierfür muss als Wert eines der verfügbaren Schlüsselwörter angegeben werden, welche Sie aus der unten stehenden Tabelle entnehmen können. Für die meisten Elemente entspricht der Wert `default` dem angezeigten Cursor. Bei Hyperlinks ist es der Cursor des Typs `pointer`. Die Tabelle enthält die wichtigsten und meist benutzten Cursor-Typen.

<b>auto</b>	Der Cursor wird automatisch vom Browser gewählt.
<b>none</b>	Der Cursor wird ausgeblendet.
<b>default</b>	Der normale "Standard" Cursor.
<b>pointer</b>	Der Cursor zeigt auf etwas.
<b>grab</b>	Der Cursor greift etwas.
<b>move</b>	Der Cursor zeigt ein Symbol zum Verschieben.
<b>no-drop</b>	Der Cursor zeigt ein Symbol, dass das Element hier nicht abgelegt werden kann.
<b>text</b>	Der Cursor für Texteingaben.
<b>wait</b>	Der Cursor zeigt ein Wartesymbol.
<b>progress</b>	Der Cursor zeigt ein Wartesymbol in Verbindung mit dem Standard-Cursor.

```

1  div
2  {
3      background-color: red;
4      width: 200px;
5      height: 200px;
6      cursor: pointer;
7  }
    
```



### Listen

Die Eigenschaft `list-style-type` definiert den **Typ** einer Liste. Hiermit ist es möglich, das Aufzählungszeichen von Aufzählungslisten oder die Aufzählungsart einer nummerierten Liste zu ändern. Für Aufzählungslisten sind hier die folgenden Werte erlaubt: `none` (kein Zeichen), `disc` (Kreis gefüllt), `circle` (Kreis ungefüllt) und `square` (Rechteck). Für nummerierte Listen stehen hingegen folgende Werte zur Verfügung: `none` (keine Aufzählung), `decimal` (Dezimalzahlen), `lower-greek` (kleine griechische Buchstaben), `lower-latin` (kleine lateinische Buchstaben), `lower-roman` (kleine römische Zahlen), `upper-latin` (große lateinische Buchstaben) und `upper-roman` (große römische Zahlen).

Die Eigenschaft `list-style-image` gibt die Möglichkeit, mit Hilfe des Schlüsselworts `url` eine **Grafik als Aufzählungszeichen** anzuzeigen. Wird zusätzlich die Eigenschaft `list-style-type` angegeben, so erhält die Grafik den Vorzug. Sollte die Grafik jedoch nicht geladen werden können oder unterstützt der Browser die Grafik nicht, so kann der Browser trotzdem das angegebene Aufzählungszeichen anzeigen. Es handelt sich hierbei um einen sogenannten „Fallback“.

Standardmäßig werden die Aufzählungszeichen (egal ob Aufzählungsliste oder nummerierte Liste) außerhalb des Elements angezeigt. Diese Einstellung kann mittels der Eigenschaft `list-style-position` geändert werden. Über den Wert `inside` ist es damit möglich, die Zeichen innerhalb der Liste zu **platzieren**. Der Wert `outside` ist der Standardwert und muss nicht explizit angegeben werden.

Die Eigenschaft `list-style` dient als **universelle Eigenschaft**. Bei der Notation der Werte in der Eigenschaft `list-style` muss dabei auf die Reihenfolge Typ (`list-style-type`), Position (`list-style-position`) und Bild (`list-style-image`) geachtet werden. Es ist möglich, ein oder mehrere Werte davon wegzulassen.

```

1  ul
2  {
3      padding: 0;
4      list-style-type: square;
5      list-style-image: url("/Bilder/Icons/right.svg");
6      list-style-position: inside;
7  }
    
```



### Tabellen

Mit Hilfe der Eigenschaft `caption-side` ist es möglich, die im `caption`-Element notierte **Überschrift** zu platzieren. Als Werte sind `top` und `bottom` möglich, wovon der Wert `top` dem Standardwert entspricht. Über den Wert `bottom` lässt sich die Überschrift unterhalb der Tabelle, an Stelle von oberhalb der Tabelle, platzieren.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

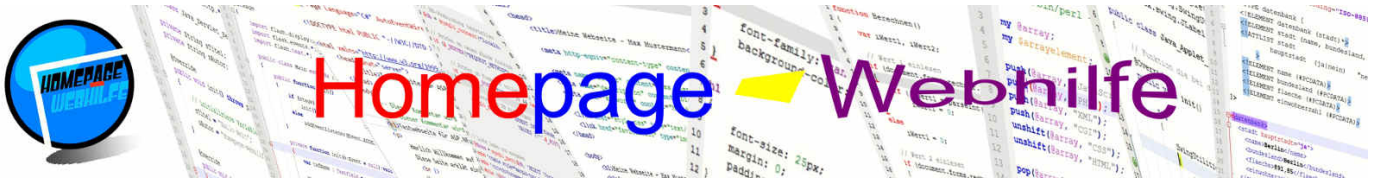
#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » Weitere CSS-Eigenschaften

Die Eigenschaft `table-layout` ermöglicht das **Verändern des Tabellenlayouts**. Als Werte sind `auto` (Standard-Wert) und `fixed` möglich. Die Breite von Tabellenspalten werden normalerweise automatisch ermittelt. Haben Sie jedoch eine Breite für eine Zelle festgelegt und diese enthält einen nicht umbrechbaren Inhalt, so wird die Zelle automatisch vergrößert, sodass der Inhalt hineinpasst. Dies kann dazu führen, dass eine Tabelle breiter als „erlaubt“ ist. Über den Wert `fixed` lässt sich dieses Problem „lösen“. Damit wird die angegebene Breite immer eingehalten. Dies hat aber u. U. Überlappungen zur Folge (siehe Beispiel).

```

1  table
2  {
3      table-layout: fixed;
4      caption-side: bottom;
5      width: 100px;
6  }
7
8  td
9  {
10     height: 30px;
11     vertical-align: top;
12 }
13
14 td:first-child
15 {
16     width: 60%;
17 }
18
19 td:last-child
20 {
21     width: 40%;
22 }
    
```



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Media Queries](#)

## Media Queries

### Inhalt dieser Seite:

1. Druck-Optimierung
2. Responsives Webdesign

Media Queries erlauben es, Regelblöcke nur auf **bestimmte Ausgabemedien** anzuwenden. Hierbei kann sowohl der Typ (Bildschirm oder Drucker) als auch die **Auflösung** abgefragt werden. Mit Hilfe dieser Technologie werden zum einen Webseiten druckoptimiert und zum anderen responsiv gemacht. Das Erstellen eines solchen „mehrgleisigen“ Layouts bedeutet natürlich mehr Aufwand, doch dieser Aufwand lohnt sich. Spätestens jetzt werden Sie merken, wie wichtig ein gutes HTML-Layout ist.

Für die Definition von Media Queries gibt es mehrere Möglichkeiten. Zum einen ist es möglich, innerhalb einer Datei (oder innerhalb der `style`-Tags) die sogenannte `@media`-Rule einzufügen. Dies sieht nun bspw. wie folgt aus:

```

1  @media screen
2  {
3      /* Hier kommt der Code für die Browseranzeige! */
4  }
5
6  @media print
7  {
8      /* Hier kommt der Code für die Druckausgabe! */
9  }
```

Alternativ können wir den CSS-Code in mehrere Dateien aufteilen und eine Unterscheidung des Medientyps an Hand des `media`-Attributs im `link`-Element durchführen:

```

1  <link rel="stylesheet" href="/web.css" type="text/css" media="screen" />
2  <link rel="stylesheet" href="/papier.css" type="text/css" media="print" />
```

Bei der Angabe eines Medientyps werden die Schlüsselwörter `all` (alle Medien), `screen` (Browseranzeige) oder `print` (Druckausgabe) verwendet. Diese Schlüsselwörter können mit dem Schlüsselwort `and` und weiteren Angaben kombiniert werden. Mehrere Media Queries können mit Hilfe eines Kommas kombiniert werden. Um die Größe des Ausgabemediums zu überprüfen, stehen uns die Eigenschaften `min-width` und `max-width` zur Verfügung. Um solche Eigenschaften in Media Queries einzubauen, wird der Eigenschaftsname gefolgt von einem Doppelpunkt und dem Wert in Klammern notiert. Dies kann dann z. B. so aussehen:

```

1  @media screen and (max-width: 800px)
2  {
3      /* Hier kommt der Code! */
4  }
```

Eine Angabe wie `(min-width: 400px)` bewirkt, dass die CSS-Regeln auf alle Geräte angewendet werden, bei welchen die Breite des Ausgabemediums größer oder gleich 400px sind. Eine Angabe wie `(max-width: 800px)` wendet die CSS-Regeln auf allen Geräte an, bei welchen die Breite des Ausgabemediums kleiner oder gleich 800px sind.

**Wichtig:** Ältere Browser unterstützen nur die einfache Form der Media Queries (und nicht die Verwendung der Eigenschaften `min-width` oder `max-width`), mit welchen lediglich zwischen `screen` und `print` unterschieden wird. Das Problem das hierdurch entsteht, ist, dass der Browser den zuletzt erkannten Medientyp `screen` fälschlicherweise für alle Browseranzeigen anwendet. Um diesem Problem entgegenzuwirken, notieren wir vor dem Medientyp der „problematischen“ Media Queries das Schlüsselwort `only`. Moderne Browser ignorieren dieses Schlüsselwort, beachten jedoch das Media Query selbst. Ältere Browser kennen das `only` Schlüsselwort nicht und verwerfen den kompletten Media Query.

```

1  @media only screen and (max-width: 800px)
2  {
3      /* Hier kommt der Code! */
4  }
```

## Druck-Optimierung

Um unsere Webseite für den Druck zu optimieren, können wir ein Media Query mit dem Schlüsselwort `print` nutzen. Bei der Druck-Optimierung sollte darauf geachtet werden, dass **unwichtige Bestandteile der Website ausgeblendet** werden. Hierzu zählen z. B. Navigationsleisten oder auch die Fußzeile. Des Weiteren sollte auch beachtet werden, dass Hintergründe (Bild und Farbe) von Browsern standardmäßig nicht gedruckt werden. Deshalb sollten diese in Druck-Layouts weitestgehend vermieden werden.

In CSS gibt es die Eigenschaften `page-break-after` (Seitenumbruch nach dem Element), `page-break-before` (Seitenumbruch vor dem Element) und `page-break-inside` (Seitenumbruch innerhalb des Elements), mit welchen die Umbrüche gesteuert werden können. Für alle Eigenschaften stehen die Werte `auto` (Browser entscheidet über den Seitenumbruch) und `avoid` (Seitenumbruch vermeiden, wenn möglich) zur Verfügung. Für die Eigenschaften `page-break-after` und `page-break-before` stehen zusätzlich die Werte `always` (Seitenumbruch immer einfügen), `left` (fügt einen Seitenumbruch ein, sodass die nächste Seite eine linke Seite ist) und `right` (fügt einen Seitenumbruch ein, sodass die nächste Seite eine rechte Seite ist) zur Verfügung.

Das folgende Beispiel zeigt ein Webseiten-Layout mit einfachen Druck-Optimierungen:

```

1  <div id="seite">
2      <header>
3          <h1>Meine Webseite</h1>
4      </header>
5
6      <nav>
7          <ul>
8              <li><a href="">Seite 1</a></li>
9              <li><a href="">Seite 2</a></li>
10             <li><a href="">Seite 3</a></li>
11             <li><a href="">Seite 4</a></li>
12             <li><a href="">Seite 5</a></li>
13         </ul>
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Media Queries](#)

```

14     </nav>
15
16     <main>
17         <p>...</p>
18         <article>
19             <h2>Artikel 1</h2>
20             <p>...</p>
21         </article>
22         <article>
23             <h2>Artikel 2</h2>
24             <p>...</p>
25         </article>
26         <article>
27             <h2>Artikel 3</h2>
28             <p>...</p>
29         </article>
30         <article>
31             <h2>Artikel 4</h2>
32             <p>...</p>
33         </article>
34         <article>
35             <h2>Artikel 5</h2>
36             <p>...</p>
37         </article>
38     </main>
39
40     <footer>
41         
42         <p>
43             Copyright 2016 by Homepage-Webhilfe<br />
44             All rights reserved!
45         </p>
46         <br />
47     </footer>
48 </div>
49
50 @media all
51 {
52     /* Seiten-Titel */
53     header>h1
54     {
55         margin: 0 0 10px;
56         padding: 0;
57         text-align: center;
58     }
59
60     /* Inhaltsbereich */
61     main h2
62     {
63         margin: 25px 0 0;
64         padding: 0;
65     }
66     main p
67     {
68         margin: 5px 5px 0;
69         text-align: justify;
70     }
71 }
72
73 @media screen
74 {
75     body
76     {
77         /* Hintergrundfarbe um Seiten-Container abzusetzen */
78         background-color: #EEEEEE;
79     }
80
81     /* Seiten-Container */
82     #seite
83     {
84         background-color: white;
85         margin: 20px;
86         padding: 10px;
87     }
88
89     /* Navigationsleiste (links mit Aufzählungspunkten) */
90     nav
91     {
92         float: left;

```

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Media Queries](#)

```

44     width: 20%;
45   }
46
47   /* Inhaltsbereich */
48   main
49   {
50     float: right;
51     width: 80%;
52   }
53   main p:last-child
54   {
55     /* Abstand zur Fußzeile halten */
56     margin-bottom: 40px;
57   }
58
59   /* Fußzeile
60   * links -> Bild
61   * rechts -> Copyright-Hinweis
62   */
63   footer
64   {
65     margin: 0 10px;
66     padding-top: 10px;
67     /* Elementfluss der Navigationsleiste und des Inhaltsbereichs beenden */
68     clear: both;
69     border-top: 2px solid black;
70   }
71   footer>img
72   {
73     float: left;
74     width: 50px;
75   }
76   footer>p
77   {
78     float: right;
79     width: calc(100% - 50px);
80     text-align: center;
81     margin: 0;
82   }
83   footer>br
84   {
85     /* Elementfluss für Logo und Copyright-Hinweis beenden */
86     clear: both;
87   }
88 }
89
90 @media print
91 {
92   /* Navigationsleiste -> für Druck ausblenden */
93   nav
94   {
95     display: none;
96   }
97
98   /* Seitenumbruch steuern */
99   main>article
100  {
101    page-break-inside: avoid;
102  }
103
104   /* Fußzeile -> für Druck ausblenden */
105   footer
106   {
107     display: none;
108   }
109 }

```



## Responsives Webdesign

Um ein responsives Webdesign zu erstellen, können bei den Media Queries die Eigenschaften `min-width` und `max-width` abgefragt werden. Wie bereits oben erklärt, führt eine Angabe wie `(max-width: 800px)` dazu, dass die dortigen Regeln für alle Geräte mit einer Auflösung von 800px oder kleiner angewendet werden. Bei `min-width` ist dies genau anders herum. Bei der Notation im CSS-Code oder bei der Einbindung im HTML-Code muss auf die Reihenfolge geachtet werden. Wird `max-width` verwendet, so müssen die Werte in absteigender Reihenfolge (siehe Beispiel) notiert werden. Bei der Verwendung von `min-width` hingegen in



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Media Queries](#)

aufsteigender Reihenfolge.

Bildquelle: [Vektor-Grafik von Freepik](#)



Für mobile Geräte muss der sogenannte „Viewport“ eingestellt werden. Dies kann mit Hilfe des `meta`-Elements und dem Attribut `name` mit dem Wert `viewport` eingestellt werden. Im `content`-Attribut können wir die **Breite der Website** und den **Skalierungsfaktor** einstellen. Hierfür notieren wir im `content`-Attribut den Wert `width=device-width`. Diese Anweisung bewirkt, dass die angezeigte Größe der Website der verfügbaren Breite des Geräts entspricht. Des Weiteren kann im `content`-Attribut auch der Wert `initial-scale` angegeben werden. Hiermit ist es möglich, den Skalierungsfaktor einzustellen. Dieser sollte auf 1.0 gesetzt werden. Hierzu folgendes Beispiel, welches auf vielen Webseiten und auch im unteren Beispiel eingesetzt wird:

```
1 | <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

Das folgende Beispiel zeigt ein responsives Webdesign, welches eine Bildschirmansicht mit zwei zusätzlichen Abstufungen für kleinere Auflösungen enthält.

```
1 | <div id="seite">
2 |   <header>
3 |     
4 |   </header>
5 |
6 |   <nav>
7 |     <ul>
8 |       <li><a href="#">Seite 1</a></li>
9 |       <li><a href="#">Seite 2</a></li>
10 |      <li><a href="#">Seite 3</a></li>
11 |      <li><a href="#">Seite 4</a></li>
12 |      <li><a href="#">Seite 5</a></li>
13 |    </ul>
14 |  </nav>
15 |
16 |  <main>
17 |    <p>...</p>
18 |    <article>
19 |      <h2>Artikel 1</h2>
20 |      <p>...</p>
21 |    </article>
22 |    <article>
23 |      <h2>Artikel 2</h2>
24 |      <p>...</p>
25 |    </article>
26 |    <article>
27 |      <h2>Artikel 3</h2>
28 |      <p>...</p>
29 |    </article>
30 |    <article>
31 |      <h2>Artikel 4</h2>
32 |      <p>...</p>
33 |    </article>
34 |    <article>
35 |      <h2>Artikel 5</h2>
36 |      <p>...</p>
37 |    </article>
38 |  </main>
39 |
40 |  <aside>
41 |    <h2>Über uns</h2>
42 |    <p>...</p>
43 |    <h2>Wussten Sie schon?</h2>
44 |    <p>...</p>
45 |  </aside>
46 |
47 |  <footer>
48 |    Copyright 2016 by Homepage-Webhilfe - All rights reserved!
49 |  </footer>
50 | </div>
51 |
52 | @media all
53 | {
54 |   body
55 |   {
56 |     background-color: #EEEEEE;
57 |   }
58 |
59 |   #seite
60 |   {
61 |     padding: 10px;
62 |     margin: 10px;
63 |     background-color: white;
64 |   }
65 | }
```

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Media Queries](#)

```

15     header>img
16     {
17         width: 100%;
18     }
19
20     /* Navigation als 1. Spalte anzeigen */
21     nav
22     {
23         float: left;
24         width: calc(20% - 20px);
25         margin: 10px;
26     }
27
28     /* Inhaltsbereich als 2. Spalte anzeigen */
29     main
30     {
31         float: left;
32         width: calc(60% - 20px);
33         margin: 10px;
34     }
35
36     /* Nebenbereich als 3. Spalte anzeigen */
37     aside
38     {
39         float: right;
40         width: calc(20% - 20px);
41         margin: 10px;
42     }
43
44     footer
45     {
46         clear: both;
47         text-align: center;
48     }
49 }
50
51 /* Gilt für alle Bildschirme mit einer Auflösung <= 820px */
52 @media only screen and (max-width: 820px)
53 {
54     /* Ränder entfernen, sodass Seite direkt am Bildschirmrand beginnt */
55     body
56     {
57         margin: 0;
58         padding: 0;
59     }
60     #seite
61     {
62         margin: 0;
63         padding: 0;
64     }
65
66     /* Navigation etwas vergrößern und Inhaltsbereich etwas verkleinern */
67     nav
68     {
69         width: calc(30% - 20px);
70     }
71     main
72     {
73         width: calc(70% - 20px);
74     }
75
76     /* Nebenbereich unterhalb des Inhaltsbereichs anzeigen */
77     aside
78     {
79         float: right;
80         width: calc(70% - 20px);
81         clear: none;
82         border-top: 1px solid black;
83     }
84 }
85
86 /* Gilt für alle Bildschirme mit einer Auflösung <= 480px */
87 @media only screen and (max-width: 480px)
88 {
89     /* Navigation jetzt oberhalb des eigentlichen Inhaltes anzeigen */
90     nav
91     {
92         border-bottom: 1px solid black;

```

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Media Queries](#)

```
93     }
94
95     /* Navigation, Inhaltsbereich und Neben-Bereich bekommen die volle Breite */
96     nav, main, aside
97     {
98         float: left;
99         width: calc(100% - 20px);
100        clear: none;
101    }
102 }
```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislöcher



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung





Sie befinden sich hier: [Homepage-Webhilfe](#) » [CSS](#) » [Abschluss](#)

## Abschluss

Nun haben wir auch schon das Ende dieses CSS Tutorials erreicht. Mit dem Wissen, welches Sie im HTML- und CSS-Kurs gelernt haben, ist es Ihnen nun möglich, bereits **professionelle statische Webseiten** zu erstellen.

CSS besitzt neben den hier vorgestellten Eigenschaften natürlich noch viele weitere Eigenschaften. Diese werden jedoch in der Praxis kaum eingesetzt oder werden von Browsern nicht richtig unterstützt. Als Referenz kann neben diesem Tutorial und unseren Karteikarten die [CSS Referenz von W3Schools](#) verwendet werden.

Doch wie geht es nun eigentlich weiter: Je nachdem was Sie benötigen, ist es nun an der Reihe, client- oder serverseitige Skript- oder Programmiersprachen zu lernen. Mit clientseitigen Skript- und Programmiersprachen wie [JavaScript](#) oder [ActionScript](#) ist es möglich, eine Webseite **aktiv** zu gestalten. Um die Webseite hingegen dynamisch zu erstellen, gibt es noch Sprachen wie z. B. [PHP](#), [Perl](#) und [ASP.NET](#). Sofern Sie an diesen Sprachen Interesse haben, würden wir uns freuen, wenn Sie die auf dieser Website verfügbaren Tutorials nutzen.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislungen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

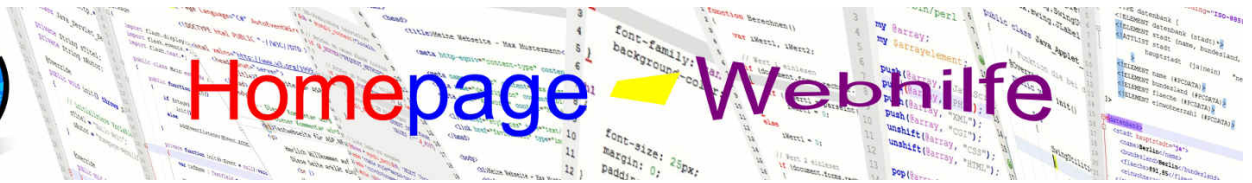
Java EE

XML

# E-Book

## JavaScript





Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Einführung](#)

## Einführung



JavaScript (oder kurz JS) ist eine Skriptsprache, welche eingesetzt wird, um **aktive Webseiten** zu erstellen. Die Sprache kann dazu genutzt werden, HTML-Inhalte zu verändern, zu entfernen oder auch zu hinzufügen. Deshalb wird auch von **dynamischem HTML** gesprochen.

Der Programmcode von JavaScript wird unverändert an den Browser geschickt und vom Browser ausgeführt (fachlich als interpretiert bezeichnet). Abfragen wie Passwörter oder andere sicherheitsrelevante Informationen sollten daher nicht mittels JavaScript durchgeführt werden.

Die Programmierung in JavaScript kann falls gewünscht objektorientiert erfolgen. Die Sprachbestandteile von JavaScript sind im sogenannten **ECMAScript** definiert. Auch wenn JS ursprünglich für die Webentwicklung gedacht war, gibt es in der Zwischenzeit auch die Möglichkeit, JavaScript als Sprache auf Servern und Mikrocontrollern zu verwenden.

JavaScript wird in einer sogenannten **Sandbox** ausgeführt, d. h. das JavaScript-Programm kann nur auf die Inhalte der Browserseite zugreifen. Ein Zugriff auf das Browserfenster (z. B. das Schließen des aktuellen Fensters) oder der Zugriff auf Dateien auf dem Zielsystem sind verboten. Dieses Verfahren ist eines der wichtigsten Sicherheits-Features von JavaScript.

### Inhalt dieser Seite:

1. Geschichte
2. Syntax
3. Platzierung
4. Variablen und Datentypen
5. Ausgabe

## Geschichte

Die Geschichte von JavaScript beginnt bereits im Jahr 1995. Im September 1995 veröffentlichte Netscape die Vorabversion des Netscape Navigators 2.0 mit der integrierten Skriptsprache, die damals noch **LiveScript** hieß. Bereits Ende des Jahres wurde bekannt gegeben, dass Netscape eine Kooperation mit Sun Microsystems eingeleitet hat. Ziel war es mittels der Skriptsprache LiveScript innerhalb von Java-Anwendungen (Applets) interagieren zu können. Auf Grund dieser Kooperation und Änderung wurde die Sprache LiveScript in JavaScript umbenannt.

Die erste Version von JavaScript (1.0.0) wurde im März 1996 veröffentlicht. JavaScript wurde zu diesem Zeitpunkt bereits vom Microsoft Internet Explorer unterstützt. Seit 2006 (JavaScript Version 1.7.0) wird JavaScript von allen gängigen Browsern unterstützt: Mozilla Firefox, Microsoft Internet Explorer, Opera, Apple Safari, Google Chrome.

Die Schnittstelle zur Interaktion in Java-Anwendungen mittels JavaScript (namens LiveConnect) wurde abgesehen von den Browsern Netscape Navigator und Mozilla Firefox in anderen Browsern nie implementiert. LiveConnect existiert heute nicht mehr und wurde mit der Firefox-Version 3.5 entfernt.

## Syntax

JavaScript ist vom Syntax an die Programmiersprache C angelehnt. Bevor wir also nun mehr über die Skriptsprache lernen, wollen wir hier erst ein paar grundlegende Regeln zur Notation bzw. zum sogenannten Syntax besprechen.

In JavaScript gibt es sogenannte **Statements** (zu Deutsch: Anweisungen). Statements können z. B. Variablen-Deklarationen und -Zuweisungen sowie Funktionsaufrufe sein. Anweisungen müssen dabei immer mit einem Semikolon `;` abgeschlossen werden.

```
1 | x = 2 * 7;
```

Für verschiedene Zwecke kommen die runden Klammern, eckige Klammern und geschweifte Klammern zum Einsatz. Die **geschweiften Klammernpaare** werden dafür genutzt, Anweisungen in Blöcke zusammen zu fassen. Solche Blöcke werden bei Objekten, Funktionen, Schleifen und Bedingungen eingesetzt. Der Inhalt von Blöcken wird dabei auf Grund der besseren Lesbarkeit um einige Leerzeichen (z. B. 4 Leerzeichen) oder einen Tab eingerückt. **Eckige Klammern** werden bei Arrays eingesetzt. **Runde Klammern** hingegen können bei Funktionsaufrufen oder zur Gruppierung bei Berechnungen genutzt werden. Auf die konkrete Nutzung gehen wir in den jeweiligen Themen noch genauer ein.

JavaScript ist **case-sensitive**, d. h. dass JavaScript bei Schlüsselwörtern, Variablenamen, Funktionsnamen etc. zwischen Groß- und Kleinschreibung unterscheidet. So ist eine Variable (dazu weiter unten mehr) mit dem Namen `nummer` also eine andere, wie die Variable mit dem Namen `nummer`. Zudem hat sich die sogenannte **Camel Case Schreibweise** durchgesetzt. Dabei werden die einzelnen Wörter von einem Variablenamen groß geschrieben: z. B. `Kundennummer`. Zudem hat es sich in JavaScript eingebürgert, den ersten Buchstaben klein zu schreiben: z. B. `kundennummer`.

Kommentare dienen hauptsächlich zu „Dokumentationszwecken“ und können in JavaScript auf 2 verschiedene Arten notiert werden. Mit den 2 Schrägstrichen `//` wird die restliche Zeile ab der aktuellen Position auskommentiert. Über die Zeichenfolge `/*` können wir auch einen mehrzeiligen Kommentar erstellen. Um den Kommentar „zu beenden“, benötigen wir die Zeichenfolge `*/`. Natürlich ist es mit dieser Kommentierungsart auch möglich, nur einen Teil innerhalb einer Zeile auszukommentieren.

```
1 | // Einzeiliger Kommentar
2 |
3 | /*
4 |  * Mehrzeiliger
5 |  * Kommentar
6 |  */
7 | /* Einzeiliger Kommentar */
```

## Platzierung

JavaScript kann wie CSS-Code auch an verschiedenen Stellen platziert werden. Die gängigste Variante ist die Platzierung in einer **externen Datei**. Hier wird der Code komplett in der JavaScript-Datei (Endung `.js`) abgelegt. Die Einbindung der Datei erfolgt mittels des zweiteiligen `script`-Elements. Als Attribut sollte das `type`-Attribut (MIME-Typ von JavaScript) und das `src`-Attribut (URL zur Datei) angegeben werden. Dies sieht wie folgt aus:

```
1 | <script type="text/javascript" src="Diashow.js"></script>
```

Haben wir nur einen kleineren Code oder betrifft dieser Code nur die aktuelle Seite, so können wir den JavaScript-Code natürlich auch direkt **in der HTML-Datei** notieren. Hier nutzen wir ebenfalls das `script`-Element. Das `src`-Attribut fällt hier jedoch weg. Der Code selbst wird innerhalb der `script`-Tags notiert:

```
1 | <script type="text/javascript">
2 | alert('Hallo!');
3 | </script>
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Einführung](#)

**Wichtig:** JavaScript-Code wird während dem Laden des HTML-Codes ausgeführt, d. h. wenn Sie JavaScript-Anweisungen innerhalb eines HTML-Dokuments notieren, werden diese **vom Browser sofort ausgeführt**. Dies kann dazu führen, dass auf Elemente von HTML zugegriffen werden, welche zum aktuellen Zeitpunkt noch gar nicht existieren. Des Weiteren hat dies negative Auswirkung auf die **Ladezeit**, da das Laden der HTML-Seite angehalten wird, solange der JS-Code ausgeführt und / oder die JavaScript-Datei geladen wird. `script`-Elemente werden deshalb üblicherweise vor dem schließenden `body`-Tag notiert. Es gilt zu beachten, dass `script`-Elemente nur innerhalb von `head`- und `body`-Elementen vorkommen dürfen. Des Weiteren sollte JS-Code, welcher beim Laden ausgeführt werden soll, nicht direkt im `script`-Element, sondern innerhalb des [Seitenlade-Events](#) notiert werden. Dazu jedoch später mehr.

JavaScript-Code kann jedoch auch über einen **Link** (`a`-Element) ausgeführt werden. Hierfür notieren wir als Wert im `href`-Attribut `javascript:` gefolgt von dem auszuführenden JavaScript-Code (normalerweise ein Funktionsaufruf).

```
1 | <a href="javascript:alert('Hallo!')">Bitte klicken!</a>
```

Der JS-Code, welcher bei einem Event ausgeführt werden soll, kann zudem direkt im HTML-Code angegeben werden. Dies könnte wie folgt aussehen (näheres dazu [später](#)):

```
1 | <p onclick="alert('Hallo!')">Bitte klicken!</p>
```

**Übrigens:** Innerhalb der Event-Attribute von HTML kann das Semikolon weggelassen werden, sofern nur eine Anweisung notiert wird.

## Variablen und Datentypen

Variablen können in JavaScript mittels dem Schlüsselwort `var` deklariert werden. Hinter dem Schlüsselwort muss ein Leerzeichen und anschließend der Variablenname angegeben werden. Der Variablenname muss innerhalb des **Gültigkeitsbereichs** (Scope) eindeutig sein. Der Gültigkeitsbereich einer Variablen erstreckt sich über den aktuellen Block und alle untergeordneten Blöcke. Dieses System werden Sie nachher in der Praxis noch besser kennenlernen.

```
1 | var variablenName;
```

Um einer Variablen einen Wert zuzuweisen, nutzen wir das Gleichheitszeichen `=`. Diesen Vorgang nennt man in der Fachsprache **Wertzuweisung** oder Variablenzuweisung. Die erste Zuweisung einer Variablen wird als **Initialisierung** bezeichnet. Eine Variable kann natürlich mehrmals zugewiesen werden.

Anders wie in vielen anderen Programmiersprachen, muss bei der Deklaration einer Variablen kein Typ angegeben werden. Trotzdem gibt es in JavaScript Datentypen: `Number`, `Boolean`, `String`, `Object` und `Array`.

Der Datentyp `Number` enthält, wie sich schon vom Englischen ableiten lässt, **numerische Werte**. Dabei kann es sich sowohl um Dezimalwerte als auch um Gleitkommawerte (Werte mit Nachkommastellen) handeln. JavaScript speichert alle Werte intern als 64bit Gleitkommawerte (*IEEE 754*) ab. Dezimalzahlen können neben der geläufigen Dezimaldarstellung auch hexadezimal (mit führendem `0x`) oder oktal (mit führender `0`) angegeben werden. Bei Gleitkommazahlen wird als Trennzeichen ein Punkt angegeben. Das folgende Beispiel zeigt die Verwendung numerischer Werte und der mehrmaligen Wertzuweisung einer Variablen:

```
1 | var nummer = 12;
2 | var nummer2 = 37.59;
3 |
4 | // Hier eine weitere Zuweisung
5 | nummer = 47;
6 |
7 | // Und nochmal eine
8 | nummer = 79;
```

Der Datentyp `Boolean` stellt einen Datentyp mit zwei möglichen Zuständen dar: `true` (wahr) und `false` (unwahr).

```
1 | var status = true;
```

`String` ist der Datentyp für sogenannte **Zeichenketten**. Zeichenketten stellen eine Aneinanderreihung von Zeichen (Buchstaben, Zahlen und Sonderzeichen) dar und müssen in doppelten oder einfachen Anführungszeichen angegeben werden. Eine Zeichenkette kann dabei kein, ein oder mehrere Zeichen enthalten.

```
1 | var name = "Peter";
```

Zu den Datentypen `Object` und `Array` gehen wir später genauer ein.

Wollen wir mehrere Variablen auf einmal deklarieren, so können wir diese durch Kommas trennen. Zu beachten gilt jedoch, dass jede Variable, welche initialisiert werden soll, separat initialisiert werden muss.

```
1 | var zahlA = 17, zahlB = 23, zahlC;
```

Zusätzlich gibt es in JavaScript noch die zwei Spezial-Datentypen bzw. Werte `undefined` und `null`. Eine Variable, welche deklariert ist, jedoch noch **keinen Wert zugewiesen** bekommen hat, besitzt den Wert `undefined` (undefiniert). Der Wert `null` kann jeder Variablen zugewiesen werden und ist ein allgemeiner Wert für „ungültige Variablen“.

## Ausgabe

Nun wird es erst spannender, denn jetzt wollen wir damit anfangen, das **erste JavaScript-Skript** zu schreiben. Doch hierfür müssen wir erst noch wissen, wie wir in JavaScript eine einfache Ausgabe erzeugen können.

Die einfachste Möglichkeit ist die Verwendung der in JavaScript integrierten Funktion `alert()`. Hierfür notieren wir den Funktionsnamen `alert`, gefolgt von einer öffnenden runden Klammer `(`. Anschließend notieren wir die Zeichenkette. Diese kann direkt angegeben werden, so wie im Beispiel, oder mittels einer Variablen übergeben werden. Nun folgt als nächstes die schließende runde Klammer `)`. Wie bereits oben schon angesprochen, ist ein solcher Funktionsaufruf ein Statement und muss deshalb mit einem Semikolon abgeschlossen werden. Die `alert()`-Funktion erzeugt ein **Meldungsfenster**, was eine Art PopUp-Fenster darstellt. Eine mit dieser Funktion erzeugte Meldung besitzt immer einen OK-Button.

```
1 | alert("Hallo!");
```

Zusätzlich gibt es noch die Funktionen `confirm()` und `prompt()`. `confirm()` zeigt ein Fenster mit OK- und Abbrechen-Button an. Als Rückgabewert gibt die Funktion `true` (falls OK-Button gedrückt wurde) oder `false` (falls Abbrechen-Button gedrückt wurde) zurück. Der `confirm()`-Funktion wird ebenfalls eine

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Einführung](#)

Zeichenkette übergeben. Der Aufruf der Funktion `prompt()` erzeugt ein Fenster mit Eingabefeld. Als Parameter (Werte, welche übergeben werden und innerhalb des Klammerspaars notiert sind) werden zwei Zeichenketten übergeben. Die Erste stellt die „Beschriftung“ dar, wohingegen die Zweite der anzuzeigende Wert im Textfeld ist. Die beiden Parameter werden dabei (wie es bei anderen Funktionen auch der Fall ist) durch ein Komma getrennt. Der zweite Parameter ist optional und kann somit weggelassen werden. Ein Fenster, welches mit der `prompt()`-Funktion erzeugt wurde, besitzt ebenfalls zwei Buttons: „OK“ und „Abbrechen“. Wird auf „Abbrechen“ geklickt so gibt die Funktion den Wert `null` zurück. Wird auf „OK“ geklickt, so gibt die Funktion den Text des Textfeldes zurück. Im unteren Beispiel werden die drei Funktionen demonstriert. Um zu sehen, was die Funktionen zurückgeben, werden beide Rückgabewerte mit Hilfe von `alert()` ausgegeben.

```
1 | alert(confirm("Sind Sie damit einverstanden?"));
2 | alert(prompt("Geben Sie Ihren Namen ein:", "Peter"));
```

Mit der Funktion `write()` des `document`-Objektes (dazu später mehr) ist es möglich, **Daten in das HTML-Dokument zu schreiben**. In der Praxis wird diese Funktion eher selten genutzt. Für die ersten Tests ist diese Funktion jedoch sehr hilfreich, da diese im Gegensatz zu den Meldungsfenster den Text einfach nur auf die Seite schreibt und keine PopUps erzeugt. Um die Funktion aufzurufen, notieren wir `document.write()`. Dabei muss am Ende ebenfalls wieder ein Semikolon angestellt werden. Als Parameter wird der Funktion eine Zeichenkette übergeben.

```
1 | document.write("Hallo!");
```

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Bedingungen](#)

## Bedingungen

Bedingungen in Programmiersprachen erlauben es, bestimmte **Fälle abzuprüfen**, z. B. ob eine Variable einen bestimmten Wert hat. JavaScript bietet, wie in vielen anderen Sprachen auch, zwei Möglichkeiten: `if` (einfache Verzweigung) und `switch-case` (mehrfache Verzweigung).

### Inhalt dieser Seite:

1. Einfache Verzweigung
2. Mehrfache Verzweigung

## Einfache Verzweigung

Eine einfache Verzweigung dient in erster Linie dafür, nur **einen bestimmten Fall abzuprüfen**. Hierfür notieren wir als erstes das Schlüsselwort `if` gefolgt von einem runden Klammernpaar. Zwischen dem Schlüsselwort und dem Klammernpaar wird meistens noch ein Leerzeichen angegeben. Dieses dient jedoch nur zur besseren Lesbarkeit. Ein Fall stellt dabei immer einen Block dar, weshalb nach der schließenden runden Klammer ein geschweiftes Klammernpaar notiert wird. Innerhalb des Blocks werden nun Anweisungen notiert. Dies können normale Statements aber auch weitere Verzweigungen sein. Eine einfache Verzweigung kann wie folgt aussehen:

```
1  if (BEDINGUNG)
2  {
3      // Anweisungen
4  }
```

An welcher Stelle die geschweiften Klammern gesetzt werden, wirkt sich nicht auf dessen Funktion aus. Folgender Aufbau (im Vergleich zu oben) wäre ebenfalls möglich:

```
1  if (BEDINGUNG) {
2      // Anweisungen
3  }
```

**Wichtig:** Es empfiehlt sich, für einen der zwei „Stile“ zu entscheiden und diesen dann im kompletten Skript bzw. auf der kompletten Website auch durchgängig zu verwenden.

Gibt es nur ein Statement, welches innerhalb des Blocks notiert werden soll, so können die geschweiften Klammern auch weggelassen werden.

```
1  if (BEDINGUNG)
2      // Anweisung
```

Zusätzlich ist es mit solchen Verzweigungen (öfters auch einfach nur als Abfrage bezeichnet) auch möglich, noch zusätzlich einen Code-Block zu notieren, für den Fall, dass die **Bedingung nicht zutrifft**. Hierfür wird nach dem `if`-Block das Schlüsselwort `else` und ein weiterer Block notiert. Auch hier können gegebenenfalls die Klammern weggelassen werden.

```
1  if (BEDINGUNG)
2  {
3      // Anweisungen falls Bedingung zutrifft
4  }
5  else
6  {
7      // Anweisungen falls Bedingung zutrifft
8  }
```

Die angegebene Bedingung muss immer einen Wert vom Typ `Boolean` sein. Wollen wir überprüfen, ob eine Variable den Wert `true` enthält, so müssen wir also lediglich den Variablennamen notieren. Möchten wir prüfen, ob die Variable den Wert `false` enthält, so notieren wir den Variablennamen mit einem vorangestellten Ausrufezeichen (Negierung).

```
1  if (computerAngeSchaltet)
2  {
3      // Anweisungen
4  }
```

JavaScript bietet noch einige sogenannte **Operatoren**, die für den Vergleich genutzt werden können. Mit Hilfe dieser Vergleichs-Operatoren ist es möglich, einen Wert vom Typ `Boolean` „zu erzeugen“. Die untenstehende Tabelle zeigt die in JavaScript verfügbaren Vergleichs-Operatoren mit den Beispielwerten a und b:

<code>a == b</code>	Wert a ist gleich b
<code>a === b</code>	Wert und Typ a ist gleich b
<code>a != b</code>	Wert a ist ungleich b
<code>a !== b</code>	Wert und Typ a ist ungleich b
<code>a &gt; b</code>	Wert a ist größer als b
<code>a &gt;= b</code>	Wert a ist größer als oder gleich b
<code>a &lt; b</code>	Wert a ist kleiner als b
<code>a &lt;= b</code>	Wert a ist kleiner als oder gleich b

**Wichtig:** JavaScript kann beim Vergleich zwischen Wert und Wert und Wert und Typ unterscheiden. So ergibt die Bedingung `"123" == 123` den Wert `true`, wohingegen die Bedingung `"123" === 123` `false` ist. Dieses Schema gilt auch für `!=` und `!==`. Dies kommt daher, dass bei der Bedingung `"123" == 123` eine automatische Typkonvertierung durchgeführt wird. Dies passiert z. B. auch dann, wenn Sie der Funktion `alert()` eine Zahl übergeben: es erfolgt eine automatische Konvertierung in eine Zeichenkette.

Des Weiteren sollte beachtet werden, dass ein Vergleich von Objekten nur dann funktioniert, wenn es sich um das selbe Objekt handelt, nicht aber, wenn es sich um

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Bedingungen](#)

ein Objekt mit gleichem Inhalt ([dazu später mehr](#)) handelt.

Hier nun das erste Beispiel mit einem Meldungsfenster:

```
1 | if (confirm("Bitte klicken Sie auf einen der Buttons!"))
2 |     document.write("Sie haben OK geklickt!");
3 | else
4 |     document.write("Sie haben Abbrechen geklickt!");
```

**Übrigens:** Natürlich können wir auch mehrere Bedingungen verknüpfen. Hierfür gibt es das logische Und `&&` und das logische Oder `||`. Bei der Kombination von Und und Oder empfiehlt es sich bzw. ist es notwendig, mit weiteren runden Klammernpaaren zu arbeiten.

```
1 | if (BEDINGUNGA && BEDINGUNGB)
2 |     // Anweisung
```

Wie bereits oben erwähnt ist es möglich, Bedingungen zu **verschachteln**. Eine Verzweigung die im `else`-Zweig verschachtelt wurde, könnte dann z. B. so aussehen:

```
1 | if (confirm("Bitte klicken Sie auf einen der Buttons!"))
2 |     document.write("Sie haben auf OK geklickt!");
3 | else
4 | {
5 |     if (confirm("Bitte klicken Sie nochmals auf einen der Buttons!"))
6 |         document.write("Sie haben beim 1. Mal auf Abbrechen und beim 2. Mal OK geklickt!");
7 |     else
8 |         document.write("Sie haben beides Mal auf Abbrechen geklickt!");
9 | }
```

Das obige Beispiel kann jedoch in JavaScript verkürzt werden. Wenn sich die weitere Verschachtelung im `else`-Zweig befindet, können wir nämlich an Stelle des Schlüsselworts `else` das Schlüsselwort `else if` verwenden, hinter welchem eine weitere Bedingung notiert wird. Diese Bedingung wird natürlich nur geprüft, wenn die erste Bedingung nicht zutrifft. Das folgende Beispiel zeigt, wie der obige Code mittels `else if` ersetzt und verkürzt werden kann:

```
1 | if (confirm("Bitte klicken Sie auf einen der Buttons!"))
2 |     document.write("Sie haben auf OK geklickt!");
3 | else if (confirm("Bitte klicken Sie nochmals auf einen der Buttons!"))
4 |     document.write("Sie haben beim 1. Mal auf Abbrechen und beim 2. Mal OK geklickt!");
5 | else
6 |     document.write("Sie haben beides Mal auf Abbrechen geklickt!");
```

**Wichtig:** Natürlich ist auch die Verwendung mehrerer `else if`-Bedingungen möglich. Es gilt lediglich zu beachten, dass sofern ein einfacher `else`-Zweig vorhanden ist, dieser immer als letztes notiert werden muss.

**Übrigens:** In JavaScript gibt es die Möglichkeit, einfache Verzweigungen einzeilig darzustellen. Diese Methode kann nur dann verwendet werden, wenn sowohl im `if`- als auch im `else`-Zweig ein Statement notiert wird, welches jeweils einen Wert zurückgibt. Das Resultat kann dann z. B. einer Variablen zugewiesen werden. Um diese **Kurzdarstellung** zu verwenden, notieren wir die Bedingung (vorzugsweise in runden Klammern) gefolgt von einem Fragezeichen `?`. Nun folgt das Statement bzw. der Wert falls die Bedingung zutrifft. Als nächstes notieren wir einen Doppelpunkt `:` gefolgt von dem Statement oder dem Wert für den Fall, dass die Bedingung nicht zutrifft. Dies sieht dann z. B. so aus:

```
1 | var wert = (a > b) ? "a ist größer als b" : "a ist nicht größer als b";
```

## Mehrfache Verzweigung

Mehrfache Verzweigungen werden in JavaScript mit `switch-case` realisiert. Um eine solche Verzweigung aufzustellen, benötigen wir das Schlüsselwort `switch` und einen Wert der geprüft werden soll. Dieser muss dafür in einem runden Klammernpaar notiert werden. Als nächstes notieren wir einen Block mit geschweiften Klammern. Innerhalb des Blocks werden nun die **verschiedenen Fälle** angegeben. Hierfür notieren wir das Schlüsselwort `case` und anschließend den Wert, welcher mit dem oben angegebenen Wert verglichen werden soll, gefolgt von einem Doppelpunkt `:`. Nun notieren wir einen Zeilenumbruch und darunter unsere auszuführenden Statements (hier wird kein Block mit geschweiften Klammern benötigt). Als letzte Anweisung muss das Schlüsselwort `break` (gefolgt von einem Semikolon) notiert werden. Ähnlich wie bei `if-else` gibt es auch hier die Möglichkeit, den Fall abzufangen, wenn **keine der aufgelisteten Bedingungen zutrifft**. Hier nutzen wir an Stelle des `case`-Schlüsselworts `default`. Das folgende Beispiel soll diesen Text verdeutlichen:

```
1 | var monat = 3;
2 |
3 | switch (monat)
4 | {
5 |     case 1:
6 |         document.write("Januar");
7 |         break;
8 |     case 2:
9 |         document.write("Februar");
10 |        break;
11 |     case 3:
12 |         document.write("März");
13 |         break;
14 |     case 4:
15 |         document.write("April");
16 |         break;
17 |     case 5:
18 |         document.write("Mai");
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Bedingungen](#)

```

19     break;
20 case 6:
21     document.write("Juni");
22     break;
23 case 7:
24     document.write("Juli");
25     break;
26 case 8:
27     document.write("August");
28     break;
29 case 9:
30     document.write("September");
31     break;
32 case 10:
33     document.write("Oktober");
34     break;
35 case 11:
36     document.write("November");
37     break;
38 case 12:
39     document.write("Dezember");
40     break;
41 default:
42     document.write("Ungültiger Monat!");
43     break;
44 }
```

`switch-case` ist zum Überprüfen eines Wertes mit verschiedenen anderen einzelnen Werten gedacht. Die Verwendung von **Wertebereichen** ist zwar in JavaScript realisierbar, jedoch sollte darauf komplett verzichtet werden, da dies als schlechter Programmierstil gilt. Wir gehen darauf hier nicht weiter ein. Verwenden Sie für Wertebereiche daher `if` und `else`.

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Schleifen](#)

## Schleifen

JavaScript bietet insgesamt 4 Schleifen: `for`, `while`, `do-while` und `for-in`. Da die `for-in`-Schleife nur in Verbindung mit Arrays verwendet werden kann, werden wir auf diese erst im Thema [Arrays](#) genauer eingehen. Schleifen werden dazu verwendet, einen bestimmten Code so lange auszuführen, bis die Bedingung nicht mehr zutrifft.

### Inhalt dieser Seite:

1. Zählschleife
2. Kopfgesteuerte Schleife
3. Fußgesteuerte Schleife

## Zählschleife

Die `for`-Schleife oder auch gerne als Zählschleife bezeichnet wird, wie der Name schon vermuten lässt, gerne dazu verwendet, bestimmte **Vorgänge n mal auszuführen**. Hierfür bietet die `for`-Schleife schon ein einfaches Grundgerüst, um diese „Zählung“ durchzuführen. Eine `for`-Schleife wird durch das Schlüsselwort `for` gefolgt von einem runden Klammernpaar und einem Block ausgezeichnet. Die geschweiften Klammern des Blocks können weggelassen werden, wenn nur eine Anweisung innerhalb des Schleifenblocks notiert wird. Innerhalb der runden Klammern (zusammen mit dem Schlüsselwort `for` als **Schleifenkopf** bezeichnet) werden 3 Bestandteile notiert, welche durch Semikolon getrennt werden. Der erste Teil dient zur Variablendeklaration. Diese wird vor dem Aufruf der Schleife ausgeführt. Der zweite Teil stellt die Bedingung dar. Die Bedingung wird vor jedem Durchlauf der Schleife und somit auch vor dem ersten Durchlauf geprüft. Der dritte Teil dient bei der üblichen Zählschleife zum Hochzählen der im ersten Teil definierten Variablen. Hierfür kann der Operator `++` verwendet werden. Diese Operation wird in der Fachsprache **Inkrementierung** genannt. Die Anweisung `i++` entspricht der Anweisung `i += 1`, welche wiederum der Anweisung `i = i + 1` entspricht. Eine solche Verkürzung ist auch bei der Subtraktion mittels `i -- 1` und `i--` (**Dekrementierung**) möglich. Für Multiplikation und Division ist lediglich die etwas kürzere Schreibform `i *= 2` und `i /= 2` möglich.

```

1  for (var i = 0; i < 10; i++)
2  {
3      document.write(i);
4      document.write(" ");
5  }

```

Um aus einer Schleife herauszuspringen können wir das Schlüsselwort `break` (wie beim `switch-case`) nutzen. Mit Hilfe des Statements `continue` ist es möglich, die Ausführung des Codes im Schleifenblock (auch als **Schleifenrumpf** bezeichnet) zu beenden und wieder zum Schleifenkopf zu springen.

## Kopfgesteuerte Schleife

Die `while`-Schleife ist eine kopfgesteuerte Schleife, d. h. die angegebene Bedingung wird geprüft bevor der Code der Schleife (Schleifenrumpf) ausgeführt wird. Eine `while`-Schleife wird mit dem Schlüsselwort `while` und der Bedingung (angegeben in runden Klammern) gebildet. Anders wie bei der `for`-Schleife gibt es hier so etwas wie eine Variablendeklaration oder einen Code zur Inkrementierung einer Variablen nicht. Das folgende Beispiel zeigt, wie das obige Beispiel der `for`-Schleife mittels einer `while`-Schleife ersetzt werden kann:

```

1  var i = 0;
2
3  while (i < 10)
4  {
5      document.write(i);
6      document.write(" ");
7
8      i++;
9  }

```

**Wichtig:** Für einen Zählvorgang (wie im Beispiel oben) sollte in der Praxis keine `while`-Schleife, sondern eine `for`-Schleife verwendet werden. Das Beispiel soll lediglich die Verwendung der Schleife erklären.

**Übrigens:** Die `for`-Schleife ist ebenfalls eine kopfgesteuerte Schleife, da die Bedingung geprüft wird, bevor der Code im Schleifenrumpf ausgeführt wird.

## Fußgesteuerte Schleife

Die `do-while`-Schleife ist ähnlich wie die `while`-Schleife, jedoch mit dem gravierenden Unterschied, dass es sich bei der `do-while`-Schleife um eine fußgesteuerte Schleife handelt. Dies hat zur Folge, dass der Code des Schleifenrumpfs **mindestens einmal ausgeführt** wird. Die Prüfung der Bedingung der Schleife wird immer am Ende eines Schleifendurchlaufs ausgeführt. Das folgende Beispiel erzeugt die gleiche Ausgabe wie das Beispiel zur `for`- und `while`-Schleife, ist jedoch nicht als äquivalenter Ersatz zu sehen, denn wäre die Variable `i` nicht mit 0 initialisiert sondern z. B. mit 10, so würde das folgende Beispiel 10 ausgeben, obwohl die Bedingung `i < 10` ist. Vom Aufbau wird bei der `do-while`-Schleife zuerst das Schlüsselwort `do` gefolgt von einem Block notiert. Anschließend wird das Schlüsselwort `while` gefolgt von der Bedingung angegeben. Nach der runden Klammer, in welcher die Bedingung angegeben ist, muss noch ein Semikolon notiert werden. Bei der `do-while`-Schleife können die geschweiften Klammern nicht weggelassen werden.

```

1  var i = 0;
2
3  do
4  {
5      document.write(i);
6      document.write(" ");
7
8      i++;
9  }
10 while (i < 10);

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

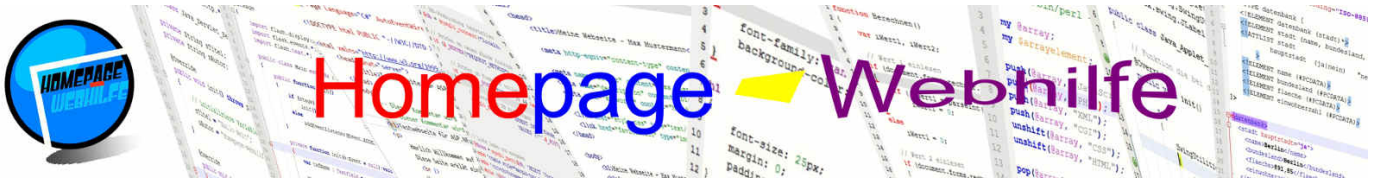
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Schleifen](#)

**Übrigens:** Der Variablenname `i` für Zählschleifen kommt vermutlich vom englischen und stellt den Anfangsbuchstaben des Worts *index* dar. Abgeleitet wurde das Ganze angeblich aus der mathematischen Summe. Werden Schleifen innerhalb von Schleifen verwendet, wird üblicherweise `j` für die zweite Schleife verwendet, `k` für die dritte Schleife usw.. Für Schleifen die nacheinander folgen, kann wieder die gleiche Variable verwendet werden.

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Arrays](#)

## Arrays

Ein Array stellt eine **Aneinanderreihung von Variablen** dar (Liste). Arrays können in JavaScript zur Laufzeit verändert werden, d. h. es können **Elemente** (dies sind die „einzelnen Variablen“ des Arrays) hinzugefügt, entfernt und sortiert werden. Ebenfalls ist es möglich, das Array zu durchsuchen, zu sortieren und zu filtern.

Ein Array wird in einer einfachen Variablen hinterlegt. Um eine Array-Variablen zu initialisieren notieren wir das Schlüsselwort `new` gefolgt vom Datentypenamen `Array`. Dahinter folgt ein rundes Klammersymbol. Bei dieser Notation handelt es sich um eine **Objekt-Instanziierung** (so nennt sich der Vorgang der Objekt-Erzeugung). Diese werden wir später beim Thema [Objektorientierung](#) nochmals aufgreifen. Als Parameter für den sogenannten Konstruktor (dies ist die Funktion für die Instanziierung des Objektes) können wir die Werte für das Array übergeben. Werden keine Werte übergeben, so wird ein leeres Array erzeugt. Ein Array kann Werte vom gleichen, aber auch von unterschiedlichen Datentypen enthalten. Zweites ist jedoch auf Grund von Such- und Sortierfunktionen nicht unbedingt zu empfehlen. Die Initialisierung eines Arrays könnte wie folgt aussehen:

```
1 | var sprachen = new Array("HTML", "CSS", "JavaScript", "ActionScript", "PHP", "Perl", "ASP.NET");
```

Zur Deklaration und Initialisierung eines Arrays gibt es auch noch eine zweite Schreibweise, welche als **Kurzschreibweise** anzusehen ist. Hierfür notieren wir bei der Wertzuweisung die eckigen Klammern `[` und `]`. Innerhalb der eckigen Klammern notieren wir nun die Werte, die wir dem Array zuweisen wollen. Dabei werden mehrere Werte durch Komma getrennt. Ein äquivalentes Beispiel zur Array-Initialisierung wie oben in der Kurzschreibweise sieht wie folgt aus:

```
1 | var sprachen = ["HTML", "CSS", "JavaScript", "ActionScript", "PHP", "Perl", "ASP.NET"];
```

Um auf die Elemente eines Arrays zuzugreifen, müssen wir den Namen des Arrays gefolgt von einem eckigen Klammersymbol notieren. Innerhalb der eckigen Klammern wird der sogenannte **Index** notiert. Die Elemente eines Arrays sind durchgängig nummeriert. Das erste Element besitzt den Index 0 (nicht 1!), das zweite Element den Index 1, das dritte Element den Index 2 usw.. Das folgende Beispiel gibt „CSS“ aus:

```
1 | var sprachen = new Array("HTML", "CSS", "JavaScript", "ActionScript", "PHP", "Perl", "ASP.NET");
2 |
3 | document.write(sprachen[1]);
```

## Array-Schleife

Wie bereits schon im Thema [Schleifen](#) angesprochen, gibt es in JavaScript die `for-in`-Schleife (umgangssprachlich auch gerne als „Array-Schleife“ bezeichnet). Diese wird dazu verwendet, die Elemente eines Arrays „durchzugehen“. Um eine `for-in`-Schleife zu notieren, geben wir das Schlüsselwort `for` gefolgt von einem runden Klammersymbol an (so wie bei der `for`-Schleife auch). Innerhalb der Klammern wird das Schlüsselwort `var`, gefolgt von einem Variablennamen, angegeben. Hinter dem Variablennamen muss nun noch das Schlüsselwort `in` sowie der Name des Arrays angegeben werden. Die in der Schleife definierte Variable enthält den Index, welcher genutzt werden kann, um auf die Elemente des Arrays zuzugreifen.

```
1 | var sprachen = new Array("HTML", "CSS", "JavaScript", "ActionScript", "PHP", "Perl", "ASP.NET");
2 |
3 | for (var sprache in sprachen)
4 | {
5 |     document.write(sprachen[sprache]);
6 |     document.write("<br />");
7 | }
```

**Übrigens:** Die `for-in`-Schleife kann auch dazu verwendet werden, die Eigenschaften eines Objekts durchzugehen. Dabei enthält die Variable, welche in der Schleife deklariert wurde, nicht den Index, sondern den Namen der Eigenschaft.

## Array verändern

Um ein Array zu verändern, gibt es die vier Funktionen: `push()`, `pop()`, `shift()` und `unshift()`. Um diese Funktionen aufrufen zu können, müssen wir den Namen der Variable gefolgt von einem Punkt und dem Funktionsnamen (inkl. Klammern mit Parameter) notieren. Über diese Weise können auch Eigenschaften gesetzt oder abgerufen werden. Der Funktion `push()` können ein oder mehrere Werte / Parameter übergeben werden, welche dem Array **am Ende hinzugefügt werden** sollen. Als Rückgabewert gibt die Funktion die **neue Länge** des Arrays zurück. Die Funktion `unshift()` ist im Vergleich zur `push()`-Funktion ähnlich, jedoch mit dem Unterschied, dass die Elemente nicht ans Ende, sondern **an den Anfang hinzugefügt** werden. Die Funktion `pop()` **entfernt das letzte Element** des Arrays und gibt den entfernten Wert zurück. Die Funktion `shift()` ist wiederum mit der Funktion `pop()` zu vergleichen. Auch hier liegt der Unterschied darin, dass sich die Funktion nicht auf das Ende, sondern auf den Anfang bezieht, d. h. die Funktion `shift()` **entfernt das erste Element** des Arrays.

```
1 | var sprachen = new Array("XML", "HTML", "CSS", "JavaScript", "ActionScript");
2 |
3 | document.write("Folgendes Element wurde entfernt: ");
4 | document.write(sprachen.shift());
5 |
6 | document.write("<br />");
7 |
8 | document.write("Das Array hat nun folgende Länge: ");
9 | document.write(sprachen.push("Java SE"));
```

**Übrigens:** Mit der Eigenschaft `length` ist es möglich, die **Länge des Arrays** abzurufen. Wie bereits erwähnt, gilt auch hier der Syntax mit dem Punkt (z. B. `sprachen.length`).

### Inhalt dieser Seite:

1. Array-Schleife
2. Array verändern
3. Arrays zusammenführen
4. Array in String umwandeln
5. Array durchsuchen
6. Array sortieren
7. Array filtern

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Arrays](#)

## Arrays zusammenführen

Um einem Array ein ganzes Array oder mehrere Arrays hinzuzufügen, gibt es die Funktion `concat()`. Als Parameter werden ein oder mehrere Arrays übergeben, welche mit dem ursprünglichen Array zusammengeführt werden sollen. Die Funktion `concat()` gibt ein neues Array zurück, d. h. das ursprüngliche Array sowie die übergebenen Arrays werden nicht verwendet.

```

1 var sprachenClient = new Array("HTML", "CSS", "JavaScript", "ActionScript");
2 var sprachenServer = new Array("PHP", "Perl", "ASP.NET");
3
4 var sprachen = sprachenClient.concat(sprachenServer);
5
6 document.write("1. Element: ");
7 document.write(sprachen[0]);
8 document.write("<br />");
9
10 document.write("letztes Element: ");
11 document.write(sprachen[sprachen.length - 1]);
12 document.write("<br />");
13
14 document.write("Länge: ");
15 document.write(sprachen.length);

```



## Array in String umwandeln

JavaScript bietet eine einfache Funktion, um die Werte eines Arrays in eine Zeichenkette umzuwandeln. Bei dieser Funktion handelt es sich um `join()`. Dieser Funktion kann als Parameter eine Zeichenkette übergeben werden, welche als **Trennzeichen** zwischen den einzelnen Elementen verwendet werden soll. Dieser Parameter ist optional. Wird dieser nicht angegeben, so wird ein Komma (ohne anschließendes Leerzeichen) zur Trennung verwendet.

```

1 var sprachen = new Array("HTML", "CSS", "JavaScript", "ActionScript", "PHP", "Perl", "ASP.NET");
2
3 document.write(sprachen.join(", "));

```



## Array durchsuchen

Die Funktion `indexOf()` durchsucht ein Array nach einem bestimmten Wert, welcher als Parameter übergeben werden muss. Wird der gesuchte Wert gefunden, so gibt die Funktion den Index des Elements zurück. Andernfalls gibt die Funktion `-1` zurück. Optional kann der Funktion ein **Startindex** als zweiter Parameter übergeben werden. Dadurch beginnt die Suche nicht beim ersten Element, sondern bei dem Element mit dem angegebenen Index. Eine Suche vom Ende zum Anfang bzw. die Suche nach dem letzten Element ist über die Funktion `lastIndexOf()` möglich. Die Möglichkeit mit dem zweiten optionalen Parameter gilt hier ebenfalls.

```

1 var sprachen = new Array("HTML", "CSS", "JavaScript", "ActionScript", "PHP", "Perl", "ASP.NET");
2
3 document.write("Position von JavaScript: ");
4 document.write(sprachen.indexOf("JavaScript"));

```



**Wichtig:** Die Suche nach Objekten in einem Array ist nicht immer ganz problemlos. Auf diese Problematik gehen wir im Thema [Objektorientierung](#) genauer ein.

## Array sortieren

Um die Werte eines Arrays zu sortieren, können wir die Funktion `sort()` verwenden. Die Sortierung erfolgt dabei **aufsteigend** (bei Zahlen nach Größe, bei Zeichenketten alphabetisch). Dazu folgendes Beispiel:

```

1 var sprachen = new Array("HTML", "CSS", "JavaScript", "ActionScript", "PHP", "Perl", "ASP.NET");
2
3 sprachen.sort();
4
5 for (var sprache in sprachen)
6 {
7     document.write(sprachen[sprache]);
8     document.write("<br />");
9 }

```



## Array filtern

Um aus einem Array nur **bestimmte Elemente herauszufiltern**, können wir die Funktion `filter()` nutzen. Als Parameter wird der `filter()`-Funktion eine Funktion als Referenz (dazu später [mehr](#)) übergeben, d. h. wir notieren als Parameter den Namen der Funktion, wie beim Funktionsaufruf auch, jedoch ohne die runden Klammern. Der angegebenen Funktion wird dabei ein einzelnes Element des Arrays als Parameter übergeben und muss `true` (Element soll beibehalten werden) oder `false` (Element soll „entfernt“ werden) zurückgeben. Auch hier gilt zu beachten, dass die Funktion `filter()` ebenfalls ein neues Array zurückgibt und das Ursprungs-Array nicht verändert.

Gerade bei solchen Funktionen, wie der Filterungsfunktion, wird gerne ein **Lambda-Ausdruck** (auch Lambda-Funktion oder anonyme Funktion genannt) verwendet.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

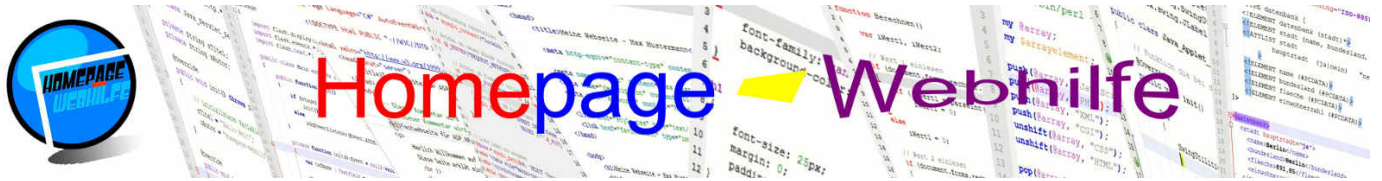
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Arrays](#)

Dieser ersetzt dabei die angegebene Filterungsfunktion. Bei dem Lambda-Ausdruck notieren wir eine Variablenamen (ohne das `var`-Schlüsselwort) gefolgt von einem Pfeiloperator `=>`. Anschließend folgt (bezogen auf das Beispiel der Filterung) ein Ausdruck bzw. eine Bedingung, um zu prüfen, ob der Wert in das neue Array mit einbezogen werden soll. Diese Bedingung muss, wie bei der `if`-Bedingung auch, einen Wert vom Typ `Boolean` zurückliefern. Alle dort vorgestellten Operatoren, sowie die Verknüpfungoperatoren `&&` und `||`, können hier verwendet werden.

```

1 | var zahlen = new Array(12, 32, 8, 50, 83, 29, 30, 100, 46, 23, 7, 132, 57, 92, 20);
2 | var zahlenNeu = zahlen.filter(x => x >= 10 && x < 100);
3 |
4 | document.write(zahlen.join(", "));
5 |
6 | document.write("<br />");
7 |
8 | document.write(zahlenNeu.join(", "));
  
```



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

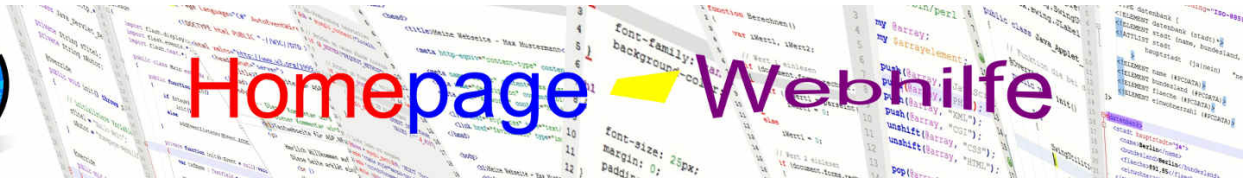
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislödingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Zahlen und Mathematik](#)

## Zahlen und Mathematik

Zahlen sind in JavaScript vom Typ `Number`. Um mit Zahlen zu arbeiten<sup>1</sup>, gibt es verschiedene **Operatoren**, welche in der untenstehenden Tabelle aufgelistet sind. Die Tabelle enthält ebenfalls die verfügbaren **Kurzschreibweisen** und deren Bedeutung.

Name	Formel	Formel (kurz)	Formel (schrittweise)
Addition	$x = y + z$	$x += y$ $\Delta$ $x = x + y$	$x++$ $\Delta$ $x += 1$ $\Delta$ $x = x + 1$
Subtraktion	$x = y - z$	$x -= y$ $\Delta$ $x = x - y$	$x--$ $\Delta$ $x -= 1$ $\Delta$ $x = x - 1$
Multiplikation	$x = y * z$	$x *= y$ $\Delta$ $x = x * y$	-
Division	$x = y / z$	$x /= y$ $\Delta$ $x = x / y$	-
Modulo	$x = y \% z$	$x \% = y$ $\Delta$ $x = x \% y$	-

**Wichtig:** An Stelle von `x++` und `x--` können wir auch `++x` und `--x` notieren. Der Unterschied besteht darin, dass bei `x++` und `x--` der „alte Wert“ zurückgegeben wird. Bei `++x` und `--x` wird die sogenannte Inkrementierung bzw. Dekrementierung ausgeführt, bevor die Variable den Wert zurückgibt. Hierzu folgendes Beispiel:

```

1 | var x = 10;
2 | alert(x++); // Gibt 10 aus
3 | alert(x);   // Gibt 11 aus

1 | var x = 10;
2 | alert(++x); // Gibt 11 aus
3 | alert(x);   // Gibt 11 aus
    
```

## Konvertierung

Um **Zeichenketten in Zahlen** umzuwandeln, gibt es die Funktionen `parseInt()` und `parseFloat()`. Die Funktion `parseInt()` wandelt dabei die Zeichenkette in eine Dezimalzahl um, `parseFloat()` hingegen in eine Gleitkommazahl (mit Nachkommastellen). Werden zwei Zeichenketten mit dem `+`-Operator verknüpft, so wird keine Addition ausgeführt, sondern die Zeichenketten werden aneinandergehängt (verbunden / verknüpft). Diese Regelung gilt auch bei Verwendung von einer Zeichenkette und einer Zahl: So ergibt `"12" + 3` nicht etwa `15`, sondern `"123"`.

Kann die Zeichenkette nicht umgewandelt werden, so geben die Konvertierungsfunktionen den Wert `NaN` zurück. Um zu überprüfen, ob eine Zahl den Wert `NaN` besitzt, gibt es die Funktion `isNaN()`, welche als Parameter den Wert erwartet und als Rückgabe einen Wert vom Typ `Boolean` zurückgibt.

```

1 | var zahlTextA = "12";
2 | var zahlTextB = "3";
3 | var zahlA = parseInt(zahlTextA);
4 | var zahlB = parseInt(zahlTextB);
5 |
6 | document.write(zahlTextA + zahlTextB);
7 | document.write("<br />");
8 | document.write(zahlA + zahlB);
    
```

**Übrigens:** Optional kann der `parseInt()`-Funktion ein zweiter Parameter übergeben werden, in welchem der **Exponent** angegeben wird, d. h. mit der Anweisung `parseInt("00100110", 2)` wird die Zeichenkette als Binärzahl interpretiert und wir erhalten den Dezimalwert `38` zurück. Standardmäßig wird für den Exponent `10` verwendet.

**Wichtig:** Mit der Funktion `Number()` ist es ebenfalls möglich, eine Konvertierung durchzuführen, jedoch muss hierbei beachtet werden, dass die Funktion `Number()` immer Gleitkommazahlen erzeugt, falls in der Zeichenkette Nachkommastellen vorhanden sind. Des Weiteren ist die Funktion `parseInt()` „weniger empfindlich“ als `Number()`, was die Konvertierung angeht: So gibt `parseInt("123a")` die Zahl `123` zurück, wohingegen `Number("123a")` als Wert `NaN` zurückgibt. Die Funktion `Number()` wird jedoch auch gerne dazu verwendet, eine Zahl in ein `Number`-Objekt umzuwandeln, um z. B. eine Funktion wie `toFixed()` aufzurufen, ohne davor die Zahl einer Variablen zuzuweisen.

## Minimum und Maximum

Um den minimalen Wert oder maximalen Wert von zwei oder mehreren Werten festzustellen, gibt es die Funktion `min()` und `max()` des statischen `Math`-Objekts. Um die Funktion aufzurufen, benötigen wir also wieder den Punktoperator: z. B. `Math.min()`. Als Parameter können zwei oder mehrere Zahlen übergeben. Kann einer der Zahlen nicht konvertiert werden, so wird `NaN` zurückgegeben.

```

1 | document.write("Minimum: ");
2 | document.write(Math.min(12, 32, 8, 50, 83, 29, 30, 100, 46, 23, 7, 132, 57, 92, 20));
3 |
4 | document.write("<br />");
5 |
6 | document.write("Maximum: ");
7 | document.write(Math.max(12, 32, 8, 50, 83, 29, 30, 100, 46, 23, 7, 132, 57, 92, 20));
    
```

**Übrigens:** Um den Minimal- oder Maximalwert aus einem Array zu bestimmen, rufen wir die Funktion `apply()` mit Referenz auf die Funktion `Math.min()` auf. Dies sieht dann z. B. so aus: `Math.min.apply(meinArray)`. Die `apply()`-Funktion kann auch für andere Funktionen angewendet werden. Die `apply()`-Funktion übergibt dabei die Werte des Arrays der referenzierten Funktion wie wenn es einzelne Werte wären.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Zahlen und Mathematik](#)

## Runden

Um Gleitkommazahlen auf Dezimalzahlen zu runden, gibt es in JavaScript drei Funktionen, die dem `Math`-Objekt angehören: `floor()`, `ceil()` und `round()`. `floor()` rundet auf die **nächstkleinere Zahl** ab, `ceil()` dagegen rundet auf die **nächstgrößere Zahl** auf. Die Rundung von `round()` erfolgt nach dem kaufmännischen Prinzip: 2,3 wird zu 2, 2,7 zu 3. Hier gilt zu beachten, dass z. B. 2,49 auf 2 abgerundet wird. Die Rundung erfolgt also nicht von der letzten Nachkommastelle zur ersten, sondern als ganze Zahl.

```

1  var zahl = 12.57;
2
3  document.write("Abrunden: ")
4  document.write(Math.floor(zahl));
5
6  document.write("<br />");
7
8  document.write("Aufrunden: ")
9  document.write(Math.ceil(zahl));
10
11 document.write("<br />");
12
13 document.write("Kaufmännisches Runden: ")
14 document.write(Math.round(zahl));

```



## Zufallszahlen

In JavaScript lassen sich mit der Funktion `random()` des `Math`-Objekts Zufallszahlen generieren. Bei den generierten Zahlen handelt es sich um **Gleitkommazahlen** zwischen 0 und 1. Dabei kann 1 als generierte Zahl nie auftreten.

```

1  for (var i = 0; i < 10; i++)
2  {
3      document.write(Math.random().toFixed(4));
4      document.write("<br />");
5  }

```



**Übrigens:** Die Funktion `toFixed()` kann dazu genutzt werden, eine Gleitkommazahl nur mit einer **bestimmten Anzahl an Nachkommastellen** darzustellen. `toFixed()` kann bei allen numerischen Variablen angewendet werden und erwartet als Parameter die Anzahl an Nachkommastellen.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislinsen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Zeichenketten](#)

## Zeichenketten

Zeichenketten sind in JavaScript vom Typ `String` (engl. für Zeichenkette). Wie auch für Zahlen gibt es für Zeichenketten einige nützliche Hilfsfunktionen, die uns das Arbeiten mit Zeichenketten erleichtern. Die Länge einer Zeichenkette kann mittels der Eigenschaft `length` ermittelt werden.

### Inhalt dieser Seite:

1. Strings verketteten
2. String zerlegen
3. Teilzeichenkette extrahieren
4. Suche in einem String

## Strings verketteten

Um zwei oder mehrere Zeichenketten miteinander zu verbinden (fachsprachlich als Verkettung bezeichnet), können wir entweder die `concat()`-Funktion oder das Pluszeichen (wie für die Addition) verwenden. Üblicherweise wird auf Grund des kürzeren Codes und der besseren Lesbarkeit die Verkettung mit dem Pluszeichen vorgezogen.

```

1  var name = "Peter";
2  var textA = "Hallo ";
3  var textB = ", wie geht es dir?";
4
5  document.write(textA.concat(name.concat(textB)));
6
7  document.write("<br />");
8
9  document.write(textA + name + textB);

```



## String zerlegen

Die Funktion `split()` trennt die Zeichenkette am angegebenen Zeichen bzw. an der angegebenen Zeichenkette und gibt ein Array zurück.

```

1  var text = "Hallo Peter, wie geht es dir?";
2  var liste = text.split(" ");
3
4  document.write(liste.join("<br />"));

```



## Teilzeichenkette

Um aus einer Zeichenkette eine Teilzeichenkette zu extrahieren, gibt es die zwei Funktionen `substr()` und `substring()`. Beiden Funktionen wird als erster Parameter der sogenannte Startindex übergeben. Der **Startindex** ist die Positionsnummer innerhalb der Zeichenkette. Dabei wird, wie beim Array-Index ebenfalls, bei 0 zu zählen begonnen, d. h. das erste Zeichen besitzt den Index 0, das zweite Zeichen den Index 1 usw.. Beide Funktionen besitzen einen zweiten Parameter mit unterschiedlicher Bedeutung. Der zweite Parameter ist jedoch optional. Bei der Funktion `substr()` legt der zweite Parameter die **Länge** der zu extrahierenden Teilzeichenkette fest. Bei der Funktion `substring()` hingegen gibt der zweite Parameter den **Endindex** an. Mit dieser Funktion ist es also möglich, eine Teilzeichenkette von Position a bis Position b zu extrahieren. Bei `substr()` wird hingegen eine Teilzeichenkette von Position a mit der Länge von b extrahiert.

```

1  var text = "Hallo Peter, wie geht es dir?";
2
3  document.write(text.substr(6, 5));
4
5  document.write("<br />");
6
7  document.write(text.substring(13, 28));

```



## Suche in einem String

Wollen wir innerhalb eines Strings nach einer Teilzeichenkette oder einem Zeichen suchen, so können uns die Funktionen `indexOf()` und `lastIndexOf()` behilflich sein. Der zu suchende String wird dabei den Funktionen als Parameter übergeben. Den Funktionen kann zusätzlich noch ein zweiter Parameter übergeben werden, dieser legt dabei den Startindex für die Suche fest.

```

1  var text = "Hallo Peter, wie geht es dir?";
2
3  document.write("Position Name: " + text.indexOf("Peter"));
4
5  document.write("<br />");
6
7  document.write("Position letztes Leerzeichen: " + text.lastIndexOf(" "));

```



**Übrigens:** Mit Hilfe der Funktion `toString()` lassen sich Werte von einem anderen Datentyp (z. B. ein Wert vom Typ `Number`) in eine Zeichenkette umwandeln. Der Aufruf dieser Funktion ist meistens jedoch nicht notwendig, da dort eine automatische Typkonvertierung stattfindet. Eine solche Konvertierung findet z. B. statt, wenn Sie `"12" + 3` notieren.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » JavaScript » Datumsangaben

## Datumsangaben

### Inhalt dieser Seite:

1. Datumsteile auslesen
2. Datumsteile ändern
3. Formatierte Ausgabe

Ein Datum kann in JavaScript mittels des Objekts `Date` dargestellt werden. Um auf die Bestandteile eines Datums zuzugreifen, gibt es einige Funktionen, welche wir weiter unten vorstellen. Des Weiteren gibt es auch Funktionen die es ermöglichen, das Datum im Nachhinein zu verändern. Um ein Datum zu initialisieren (also zu erstellen), stellen wir Ihnen 4 verschiedene Varianten vor. Für alle Varianten ist es notwendig, ein `Date`-Objekt mittels des Konstruktors zu erstellen. Zur Erinnerung: Hierfür notieren wir das Schlüsselwort `new` gefolgt von dem Objektnamen (also `Date`) und einem runden Klammerspaar. Innerhalb der Klammern werden die Funktions-Parameter (Konstruktor-Parameter) übergeben.

Bei der ersten Variante geben wir der Funktion kein Parameter mit. Dadurch erhält das erstellte Datum das **aktuelle Datum mit aktueller Uhrzeit**.

```
1 | var datum = new Date();
```

Die zweite Möglichkeit ist, der Funktion einen numerischen Wert zu übergeben. Dieser numerische Wert muss den sogenannten **UNIX-Zeitstempel als Millisekundenwert** widerspiegeln. Der UNIX-Zeitstempel ist ein numerischer Wert, welcher die Sekunden seit dem 01.01.1970 um Mitternacht bis zum gewünschten Zeitpunkt darstellt.

```
1 | var datum = new Date(1451602800000); // Entspricht 01.01.2016 00:00:00
```

Eine weitere Variante ist, dem Konstruktor die **Einzelteile des Datums und Uhrzeit** zu übergeben. Dabei muss die Reihenfolge Jahr, Monat, Tag, Stunde, Minute, Sekunde, Millisekunde eingehalten werden. Bitte beachten Sie, dass die Monats-Angabe nullbasierend ist, d. h. Monat 0 entspricht Januar (1), Monat 1 entspricht Februar (2), ... und Monat 11 entspricht Dezember (12).

```
1 | var datum = new Date(2016, 1, 1, 3, 4, 5, 678); // Entspricht 01.02.2016 03:04:05,678
```

Die letzte Variante ist die Angabe einer **Zeichenkette**. Hier muss das Datum (evtl. mit Uhrzeit) in einem bestimmten Format übergeben werden. Leider kann es hier zu Problemen bei einigen Browsern kommen, da nicht alle Formate von jedem Browser unterstützt werden. Die Verwendung des **ISO-8601 Formats** gilt als sicher und sollte von allen Browsern umgesetzt werden können.

```
1 | var datum = new Date("2016-02-01"); // Entspricht 01.02.2016 00:00:00
```

## Datumsteile auslesen

Wie bereits oben erwähnt, gibt es einige Funktionen, um die Bestandteile eines Datums und einer Uhrzeit auszulesen. Dabei gibt es Funktionen, mit welchen das UTC-Datum bzw. die **UTC-Zeit** und das Lokaldatum bzw. die **Lokalzeit** ausgegeben werden kann. Wird im Konstruktor ein Millisekundenwert übergeben, so wird zu diesem zusätzlich die Zeitdifferenz für die Lokalzeit mit eingerechnet. Bei der Angabe der einzelnen Datums- und Uhrzeit-Bestandteile wird exakt die notierte Angabe als Lokalzeit interpretiert. Bei Angabe der Zeichenkette wird diese als UTC-Zeit interpretiert, sofern nicht im String etwas anderes explizit angegeben ist.

Lokalzeit	<code>getDay()</code>	Gibt den Wochentag zurück (0 = Sonntag, 1 = Montag, ..., 6 = Samstag).
	<code>getDate()</code>	Gibt den Tag des Monats zurück (1 - 31).
	<code>getMonth()</code>	Gibt den Monat zurück (0 = Januar, 1 = Februar, ..., 11 = Dezember).
	<code>getFullYear()</code>	Gibt das Jahr zurück.
	<code>getHours()</code>	Gibt die Stunden zurück (0 - 23).
	<code>getMinutes()</code>	Gibt die Minuten zurück (0 - 59).
	<code>getSeconds()</code>	Gibt die Sekunden zurück (0 - 59).
UTC-Zeit	<code>getMilliseconds()</code>	Gibt die Millisekunden zurück (0 - 999).
	<code>getUTCDay()</code>	Gibt den Wochentag zurück (0 = Sonntag, 1 = Montag, ..., 6 = Samstag).
	<code>getUTCDate()</code>	Gibt den Tag des Monats zurück (1 - 31).
	<code>getUTCMonth()</code>	Gibt den Monat zurück (0 = Januar, 1 = Februar, ..., 11 = Dezember).
	<code>getUTCFullYear()</code>	Gibt das Jahr zurück.
	<code>getUTCHours()</code>	Gibt die Stunden zurück (0 - 23).
	<code>getUTCMinutes()</code>	Gibt die Minuten zurück (0 - 59).
	<code>getUTCSeconds()</code>	Gibt die Sekunden zurück (0 - 59).
<code>getUTCMilliseconds()</code>	Gibt die Millisekunden zurück (0 - 999).	

Hier folgendes Beispiel zur Ausgabe einiger Datumsbestandteile:

```
1 | var datum = new Date();
2 |
3 | document.write("Datum: " + datum.getDate() + "<br />");
4 | document.write("Monat: " + (datum.getMonth() + 1) + "<br />");
5 | document.write("Jahr: " + datum.getFullYear() + "<br />");
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Datumsangaben](#)

## Datumsteile ändern

Passend zu den Funktionen zum Auslesen der Datumsbestandteile gibt es auch Funktionen zum Setzen der Datumsbestandteile. Auch hier gibt es wieder Funktionen für die UTC-Zeit und Lokalzeit.

Lokalzeit	<code>setDate()</code>	Setzt den Tag des Monats (1 - 31).
	<code>setMonth()</code>	Setzt den Monat (0 = Januar, 1 = Februar, ..., 11 = Dezember).
	<code>setFullYear()</code>	Setzt das Jahr.
	<code>setHours()</code>	Setzt die Stunden (0 - 23).
	<code>setMinutes()</code>	Setzt die Minuten (0 - 59).
	<code>setSeconds()</code>	Setzt die Sekunden (0 - 59).
	<code>setMilliseconds()</code>	Setzt die Millisekunden (0 - 999).
UTC-Zeit	<code>setUTCDate()</code>	Setzt den Tag des Monats (1 - 31).
	<code>setUTCMonth()</code>	Setzt den Monat (0 = Januar, 1 = Februar, ..., 11 = Dezember).
	<code>setUTCFullYear()</code>	Setzt das Jahr.
	<code>setUTCHours()</code>	Setzt die Stunden (0 - 23).
	<code>setUTCMinutes()</code>	Setzt die Minuten (0 - 59).
	<code>setUTCSeconds()</code>	Setzt die Sekunden (0 - 59).
	<code>setUTCMilliseconds()</code>	Setzt die Millisekunden (0 - 999).

```

1 var datum = new Date();
2
3 datum.setDate(29);
4 datum.setMonth(0);
5
6 document.write("Datum: " + datum.getDate() + "<br />");
7 document.write("Monat: " + (datum.getMonth() + 1) + "<br />");
8 document.write("Jahr: " + datum.getFullYear() + "<br />");

```

Übrigens: Mit Hilfe der Funktion `getTimezoneOffset()` ist es möglich, die **Zeitdifferenz** zwischen UTC-Zeit und Lokalzeit zu ermitteln. Die Funktion gibt einen Minutenwert zurück. In Deutschland gibt die Funktion `-60` (bei Winterzeit) oder `-120` (bei Sommerzeit) zurück.

## Formatierte Ausgabe

Es gibt einige **vorgefertigte Funktionen**, um das Datum und die Uhrzeit auszugeben. Dabei geben alle Funktionen bis auf `toUTCString()` und `toISOString()` die Lokalzeit zurück. Die folgende Tabelle zeigt alle verfügbaren Funktionen zur formatierten Ausgabe:

<code>toLocaleDateString()</code>	Gibt das Datum zurück (lokale Darstellung).
<code>toDateString()</code>	Gibt das Datum zurück.
<code>toLocaleTimeString()</code>	Gibt die Uhrzeit zurück (lokale Darstellung).
<code>toTimeString()</code>	Gibt die Uhrzeit zurück.
<code>toLocaleString()</code>	Gibt das Datum und die Uhrzeit zurück (lokale Darstellung).
<code>toString()</code>	Gibt das Datum und die Uhrzeit zurück.
<code>toUTCString()</code>	Gibt das Datum und die Uhrzeit mit UTC-Zeit zurück.
<code>toISOString()</code>	Gibt das Datum und die Uhrzeit im ISO-Format zurück.

Immer wieder kann es passieren, dass Sie das **Datum und die Uhrzeit anders formatieren** wollen. Eine solche Formatierungsfunktion müssen wir uns jedoch selbst schreiben. So ist es, im Gegensatz zu Sprachen wie C (Funktion `strftime()`), nicht möglich, eine Funktion mit sogenannten Kürzeln aufzurufen. Mit Hilfe der Funktionen `getX()` ist es jedoch möglich, unsere Ausgabe zusammenzubauen. Hierzu folgendes Beispiel:

```

1 var datum = new Date();
2
3 document.write("Datum: " + datum.getDate() + "." + (datum.getMonth() + 1) + "." + datum.getFullYear());
4 document.write("<br />");
5 document.write("Uhrzeit: " + datum.getHours() + ":" + datum.getMinutes() + ":" + datum.getSeconds());

```

Wie Ihnen sicherlich auffallen wird, sind beim obigen Beispiel keine **führenden Nullen** sichtbar. Eine Funktion um Zahlen mit führenden Nullen aufzufüllen, gibt es in JavaScript ebenfalls nicht, d. h. wir müssen uns eine solche Funktion ebenfalls selber schreiben. Da wir bisher noch nicht besprochen haben, wie man eigene

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Datumsangaben](#)

Funktionen definiert, haben wir das „Problem“ mit den führenden Nullen mit einzeiligen `if`-Bedingungen direkt bei der Ausgabe gelöst.

```

1  var datum = new Date();
2
3  document.write("Datum: " +
4    (datum.getDate() >= 10 ? datum.getDate() : ("0" + datum.getDate())) + "." +
5    ((datum.getMonth() + 1) >= 10 ? (datum.getMonth() + 1) : ("0" + (datum.getMonth() + 1))) + "." +
6    datum.getFullYear());
7  document.write("<br />");
8  document.write("Uhrzeit: " +
9    (datum.getHours() >= 10 ? datum.getHours() : ("0" + datum.getHours())) + ":" +
10   (datum.getMinutes() >= 10 ? datum.getMinutes() : ("0" + datum.getMinutes())) + ":" +
11   (datum.getSeconds() >= 10 ? datum.getSeconds() : ("0" + datum.getSeconds())));

```



**Wichtig:** Um die Funktionsweise des `Date`-Objekts besser kennenzulernen, empfiehlt es sich mit Hilfe von kleineren „Test-Programmen“ die Funktionen sowie Ein- und Ausgaben des `Date`-Objekts auszuprobieren.

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Funktionen](#)

## Funktionen

### Inhalt dieser Seite:

1. Funktion mit Rückgabe
2. Funktionsausdruck
3. Selbst-aufrufende Funktion

Funktionen sind dazu gedacht, einen **Code mehrmals (und von verschiedenen Stellen) zu nutzen**. Die Funktionen und Objekte, welche zum JavaScript-Sprachkern gehören, sind bereits im Browser bzw. JavaScript-Interpreter integriert. Deren Code wird als „**native code**“ bezeichnet und kann nicht ohne weiteres eingesehen werden.

Um eigene Funktionen zu definieren, notieren wir das Schlüsselwort `function` gefolgt von einem Leerzeichen und dem Namen der Funktion. Der **Funktionsname** ist, wie der Variablenname auch, frei definierbar, muss jedoch innerhalb des Gültigkeitsbereichs eindeutig sein. Auch für Funktionen sollte die „Camel Case“-Schreibweise angewendet werden. Oft werden in JavaScript deutsche Funktionsnamen am Anfang groß geschrieben. Dies ist jedoch nur eine Frage des Programmierstils. Zurück zur Funktionsnotation: Hinter dem Funktionsnamen wird ein rundes Klammersymbol notiert. Innerhalb der Klammern können nun die **Parameter** für die Funktion kommagetrennt angegeben werden. Die Namen dieser Parameter können ebenfalls wieder frei definiert werden. Variablennamen innerhalb der Funktion, sowie die Namen der Parameter selbst, dürfen dabei jedoch nicht mehrmals vorkommen und müssen somit eindeutig sein. Nach der schließenden runden Klammer notieren wir ein Paar von geschweiften Klammern (Block). Innerhalb des **Blocks** wird nun der Code der Funktion notiert.

Um **Funktionen aufzurufen**, notieren wir den Namen der Funktion, gefolgt von einem runden Klammersymbol, in welchem die Werte für die Parameter kommagetrennt angegeben werden. Da es sich bei dem Funktionsaufruf um ein Statement handelt, muss am Ende ein Semikolon notiert werden.

```

1  function Ausgabe()
2  {
3      alert("Herzlich Willkommen!");
4  }
5
6  function AusgabeMitParameter(name)
7  {
8      document.write("Hallo " + name + "!");
9  }
10
11 Ausgabe();
12 AusgabeMitParameter("Peter");

```

Um Parameter von Funktionen als **optionale Parameter** zu kennzeichnen, können wir bei der Deklaration der Funktion nach dem Parameternamen ein Gleichheitszeichen gefolgt von dem zum Parameter gehörenden **Standardwert** (engl. *default value*) angeben. Dabei gilt zu beachten, dass optionale Parameter immer „ganz rechts“ stehen müssen. Die Reihenfolge zur Angabe der Parameter muss dabei immer eingehalten werden. Haben wir z. B. eine Funktion mit den Parametern `a`, `b`, `c` und `d`, bei welcher der Parameter `c` und `d` optional sind, so ergeben sich folgende 3 Möglichkeiten zum Aufruf der Funktion: `(a, b)`, `(a, b, c)` und `(a, b, c, d)`. Hierzu ein Beispiel:

```

1  function Addieren(a, b, c = 0, d = 0)
2  {
3      document.write(a + b + c + d);
4      document.write("<br />");
5  }
6
7  Addieren(1, 2);
8  Addieren(1, 2, 3);
9  Addieren(1, 2, 3, 4);

```

## Funktion mit Rückgabe

Funktionen können einen sogenannten Rückgabewert haben. Der Rückgabewert erlaubt es, Informationen an den Aufrufer zurückzugeben. Diesen Rückgabewert haben wir schon bei vielen Funktionen wie z. B. bei `parseInt()` genutzt. Um von der Funktion aus einen Wert zurückzugeben notieren wir das Schlüsselwort `return`, gefolgt von dem Wert, welcher zurückgegeben werden soll. Da es sich um ein Statement handelt, wird auch hier am Ende ein Semikolon verwendet.

```

1  function Addition(zahlA, zahlB)
2  {
3      return zahlA + zahlB;
4  }
5
6  document.write("Das Ergebnis aus 3 + 8 ist: " + Addition(3, 8));

```

**Wichtig:** Durch das Schlüsselwort `return` springen wir aus der Funktion. Anweisungen die „unterhalb“ der `return`-Anweisung notiert sind, werden nicht mehr ausgeführt.

**Übrigens:** Bei einer Funktion, welche keinen Rückgabewert hat, können wir das `return`-Statement auch einfach dazu nutzen, um aus der Funktion „herauszuspringen“.

## Funktionsausdruck

Die Notierung einer Funktion als Funktionsausdruck (bisher hat es sich immer um Funktionsdeklaration gehandelt) erlaubt es, die Funktion **einer Variablen oder einem Objekt** zuzuweisen. Der bisherige Syntax von der Funktionsdeklaration bleibt bei dem Funktionsausdruck gleich, jedoch mit dem Unterschied, dass **kein Funktionsname** mehr angegeben wird, denn ab sofort wird die Funktion mittels der Variable oder des Objekts aufgerufen. Wird eine Funktion einer Variablen zugewiesen, muss natürlich darauf geachtet werden, dass nach der schließenden geschweiften Klammer `}` ein Semikolon `;` notiert werden muss.

```

1  var add = function (zahlA, zahlB)
2  {

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Funktionen](#)

```

3     return zahlA + zahlB;
4 };
5
6 document.write("Das Ergebnis aus 3 + 8 ist: " + add(3, 8));

```



## Selbst-aufrufende Funktion

Selbst-aufrufende Funktionen (engl. *self invoking functions*) rufen, wie der Name schon sagt, sich selbst völlig automatisch auf. Genutzt werden solche Funktionen, um Quellcode zu **gliedern und strukturieren**. Denn es ist auch möglich, neben „normalem Quellcode“, weitere Variablen und sogar **Funktionen innerhalb dieser Funktion** zu notieren. Diese sind dann aber nur innerhalb der selbst-aufrufenden Funktion verfügbar (**Gültigkeitsbereich**) und können somit „von außen“ nicht aufgerufen, abgerufen oder geändert werden.

Die Angabe einer selbst-aufrufenden Funktion erfolgt ähnlich wie die eines Funktionsausdrucks. Zusätzlich muss der ganze Ausdruck noch in ein rundes Klammerspaar gepackt werden. Hinter der letzten schließenden runden Klammer `)` muss nun noch ein weiteres rundes Klammerspaar angegeben werden, welches dem Funktionsaufruf entspricht. Hier ist es natürlich auch wieder möglich, **Parameter** mitzugeben (sofern diese bei der Deklaration des Funktionsausdrucks notiert wurden).

Das folgende Beispiel enthält 3 selbst-aufrufende Funktionen: Die erste Funktion stellt dabei eine ganz einfache Funktion ohne Parameter dar, die zweite verfügt über einen Funktionsparameter und die dritte gibt zusätzlich noch einen Wert zurück, wessen Rückgabe direkt mittels `document.write()` auf die Seite geschrieben wird.

```

1 (function ()
2 {
3     alert("Herzlich Willkommen!");
4 })();
5
6 (function (name)
7 {
8     document.write("Hallo " + name + "!");
9 })("Peter");
10
11 document.write("<br />");
12
13 document.write("Das Ergebnis aus 3 + 8 ist: " +
14 (function (zahlA, zahlB)
15 {
16     return zahlA + zahlB;
17 })(3, 8)
18 );

```



**Wichtig:** In JavaScript gibt es einige Funktionen, wo Funktionen als **Referenz** übergeben werden müssen. Hier kann dann entweder ein Funktionsausdruck notiert werden oder der Name der Funktion ohne Klammern. Würden wir an dieser Stelle den Funktionsnamen mit Klammern angeben, so würde die Funktion direkt aufgerufen werden. Dies ist jedoch nicht gewünscht, da die Funktion erst beim Eintreten eines Ereignisses aufgerufen werden soll.

```

1 | window.onload = MeineFunktion;

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Objektorientierung](#)

## Objektorientierung

### Inhalt dieser Seite:

1. Objekteigenschaften
2. Objektfunktionen
3. Konstruktorfunktion
4. Objekt-Arrays
5. Objekterweiterung

Ein Objekt ist ein **allgemein gehaltener Datentyp**. Objekte können Sie mittels JavaScript selbst erstellen. Zu den bereits existierenden Objekt-Datentypen (auch als **globale Objekte** bezeichnet) von JS zählen `Boolean`, `Number`, `String` und `Array`. Weitere globale Objekte, bei welchen es sich jedoch nicht um Objekt-Datentypen handelt, sind z. B. `Date` und `Math`. Objekte können Variablen (sogenannte **Eigenschaften** bzw. **Objekteigenschaften**) und Funktionen (sogenannte **Objektfunktionen**) enthalten. Vereinfacht gesagt ist ein Objekt also dazu gedacht, **mehrere Werte innerhalb eines Datentyps** und somit innerhalb einer Variablen zu speichern. Objekte können dabei auf zwei verschiedene Arten erstellt werden.

Die erste Variante, die wir vorstellen werden, sind die **Einzelobjekte**. Einzelobjekte können ohne weiteres nicht mehrmals genutzt werden. Verwendet werden solche Objekte z. B. innerhalb einer Funktion, um bestimmte Daten besser „zusammen zu halten“. Um ein Einzelobjekt zu erzeugen, notieren wir das Schlüsselwort `new`, gefolgt von dem Objekttyp `Object`.

```
1 | var computer = new Object();
```

Nun haben wir ein formloses Objekt, auf welches wir nun mittels des **Punktoperators** `.` auf dessen Eigenschaften und Funktionen zugreifen können. Bevor wir auf dessen Eigenschaften und Funktionen lesend oder aufrufend zugreifen, müssen wir diese natürlich erst zuweisen. Dies sieht dann wie folgt aus:

```
1 | var computer = new Object();
2 | computer.betriebssystem = "Windows 7";
3 | computer.starteComputer = function ()
4 | {
5 |     document.write("Computer wird gestartet ...");
6 | };
7 |
8 | // Betriebssystem ausgeben
9 | document.write(computer.betriebssystem + "<br />");
10 | // Funktion aufrufen
11 | computer.starteComputer();
```

Um ein Objekt zu erzeugen, gibt es ebenfalls noch eine **Kurzschreibweise**. An Stelle ein Objekt mit dem `new`-Schlüsselwort zu **instancieren** (so nennt sich der Vorgang der Objekterzeugung), können wir auch einfach nur ein Paar von geschweiften Klammern notieren. Dies sieht wie folgt aus:

```
1 | var computer = { };
```

Der Vorteil dieser Variante ist, dass hier ganz einfach Eigenschaften und Funktionen direkt bei der Instanziierung zugewiesen werden können. Hierfür wird bei der Zuweisung nicht das bekannte Gleichheitszeichen `=`, sondern ein Doppelpunkt `:` verwendet. Des Weiteren wird die Zuweisung nicht mit einem Semikolon `;` abgeschlossen, sondern mit einem Komma `,`. Das Komma dient lediglich zur Trennung von mehreren Zuweisungen und kann bei der letzten Zuweisung somit weggelassen werden. Das Beispiel von weiter oben, lässt sich dadurch nun wie folgt ersetzen:

```
1 | var computer =
2 | {
3 |     betriebssystem : "Windows 7",
4 |     starteComputer : function ()
5 |     {
6 |         document.write("Computer wird gestartet ...");
7 |     }
8 | };
9 |
10 | // Betriebssystem ausgeben
11 | document.write(computer.betriebssystem + "<br />");
12 | // Funktion aufrufen
13 | computer.starteComputer();
```

Alle Objekte werden mittels einer **Referenz** an andere Variablen, Funktionen oder ähnliches weitergegeben. Haben wir also nun eine zweite Variable namens `computer2` und weisen dieser den Wert der Variable `computer` zu, so enthält `computer2` nicht etwa eine Kopie von `computer`, sondern enthält **exakt das gleiche Objekt**. Dies kommt daher, da in `computer` und `computer2` lediglich eine Referenz zu einem gewissen Speicherbereich gespeichert ist. Diese Referenz ist als Wert anzusehen und wurde bei der Zuweisung von der einen auf die andere Variable übertragen. Nehmen wir nun an, wir ändern jetzt am Objekt `computer2` die Eigenschaft `betriebssystem`, so ist dadurch auch automatisch der Wert der Eigenschaft `betriebssystem` des Objekts `computer` geändert. Das folgende Beispiel erläutert dieses Szenario nochmals:

```
1 | var computer =
2 | {
3 |     betriebssystem : "Windows 7"
4 | };
5 | var computer2 = computer;
6 |
7 | computer2.betriebssystem = "Windows 10";
8 | alert(computer.betriebssystem); // Gibt "Windows 10" aus
```

## Objekteigenschaften

Objekteigenschaften sind die „Variablen von Objekten“. Zugewiesen werden können diese, wie bereits oben gezeigt, mit dem Punktoperator `.` oder direkt bei der Instanziierung (siehe Beispiel). Dabei ist darauf zu achten, dass hier kein Semikolon am Ende notiert wird, da es sich hier nicht um ein Statement, sondern lediglich um eine Zuweisung von Eigenschaften und Funktionen (auch Methoden genannt) zu einem Objekt handelt. Eigenschaften können „von außen“ gelesen, aber auch gesetzt werden.

```
1 | var computer =
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Objektorientierung](#)

```

2  {
3      marke       : "Acer",
4      modell      : "Aspire E15",
5      betriebssystem : "Windows 7"
6  };
7
8  document.write(computer.marke + " " + computer.modell + " mit " + computer.betriebssystem);

```



## Objektfunktionen

Objektfunktionen (auch Objektmethoden genannt) werden in Form eines **Funktionsausdrucks** angegeben. Wie bei den Eigenschaften auch, können Funktionen direkt bei der Instanziierung sowie im Nachhinein hinzugefügt werden. Das folgende Beispiel zeigt die Zuweisung von Funktionen direkt beim Instanzieren des Objekts:

```

1  var computer =
2  {
3      betriebssystem : "Windows 7",
4      benutzer      : "admin",
5      passwort      : "123",
6
7      stateComputer : function ()
8      {
9          document.write(this.betriebssystem + " wird gestartet ...<br />");
10     },
11     stoppeComputer : function ()
12     {
13         document.write(this.betriebssystem + " wird heruntergefahren ...<br />");
14     },
15     benutzerLogin  : function (name, passwort)
16     {
17         if (name == this.benutzer && passwort == this.passwort)
18             document.write("Benutzer " + name + " erfolgreich angemeldet!<br />");
19         else
20             document.write("Benutzername oder Passwort falsch!<br />");
21     }
22 };
23
24 computer.stateComputer();
25 computer.benutzerLogin("admin", "");
26 computer.benutzerLogin("admin", "123");
27 computer.stoppeComputer();

```



Das im obigen Beispiel verwendete Schlüsselwort `this` erlaubt den expliziten Zugriff auf das aktuelle Objekt. Die Notation des `this`-Schlüsselworts ist notwendig, sodass der Interpreter weiß, um was für eine Variable o. Ä. es sich handelt. Wird `this` weggelassen, so geht der Interpreter davon aus, dass es sich um eine lokale Variable, eine globale Variable oder einen Übergabeparameter (bei einer Funktion) handelt. `this` verweist immer auf das **aktuelle Objekt**. Wird es also innerhalb eines Objekts (z. B. in einer Objektfunktion) verwendet, so verweist `this` auf dessen Objekt. Wird `this` außerhalb eines Objekts verwendet, so verweist `this` auf das globale Objekt `window`, welches zum sogenannten BOM gehört ([dazu später mehr](#)). Im obigen Beispiel könnte jedoch z. B. `this.betriebssystem` durch `computer.betriebssystem` ersetzt werden.

## Konstruktorfunktion

Eine Konstruktorfunktion (engl. *constructor function*) erlaubt es, ein Objekt so zu gestalten, dass wir es **mehrmals** und auch an **unterschiedlichen Stellen** wiederverwenden können. Hierfür deklarieren wir eine Funktion, dessen Funktionsnamen dem Objektamen entspricht. Die Parameter der Funktion sind als Konstruktorparameter anzusehen und können so wie Funktionsparameter gestaltet werden. Innerhalb der Konstruktorfunktion werden üblicherweise neben dem dazugehörigen Initialisierungs-Programmcode, auch Objekteigenschaften und Objektfunktionen angegeben. Nachdem wir unser Objekt nun erstellt haben, müssen wir dieses nur noch instanzieren. Dies geht über die bekannte Variante mit dem `new`-Schlüsselwort, gefolgt von dem definierten Objektamen. Dies sieht dann z. B. so aus:

```

1  function Computer(betriebssystem, benutzerName = "admin", benutzerPasswort = "123")
2  {
3      // Eigenschaften
4      this.betriebssystem = betriebssystem;
5      this.benutzer = benutzerName;
6      this.passwort = benutzerPasswort;
7
8      // Funktionen
9      this.stateComputer = function ()
10     {
11         document.write(this.betriebssystem + " wird gestartet ...<br />");
12     };
13     this.stoppeComputer = function ()
14     {
15         document.write(this.betriebssystem + " wird heruntergefahren ...<br />");
16     };

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Objektorientierung](#)

```

17     this.benutzerLogin = function (name, passwort)
18     {
19         if (name == this.benutzer && passwort == this.passwort)
20             document.write("Benutzer " + name + " erfolgreich angemeldet!<br />");
21         else
22             document.write("Benutzername oder Passwort falsch!<br />");
23     };
24
25     // Code
26     document.write("Computer erstellt!<br />");
27 }
28
29 // Beispiele zur Verwendung des Objekts
30 document.write("Computer 1:<br />");
31 var computer1 = new Computer("Windows 7", "admin", "admin");
32 computer1.stateComputer();
33 computer1.benutzerLogin("admin", "123");
34 computer1.stoppeComputer();
35
36 document.write("<br />");
37
38 document.write("Computer 2:<br />");
39 var computer2 = new Computer("Windows 10");
40 computer2.stateComputer();
41 computer2.benutzerLogin("admin", "123");

```



**Wichtig:** Beim obigen Beispiel und generell bei allen Konstrukturfunktionen ist das `this`-Schlüsselwort unersetzlich.

**Übrigens:** Bei der Objektdeklaration mittels einer Konstrukturfunktion ist es möglich, eine Art von **privaten Variablen** zu erstellen. Auf private Variablen kann man innerhalb eines Objekts zugreifen. Ein Zugriff „von außen“ auf diese Variablen ist jedoch nicht möglich. Private Variablen werden innerhalb der Konstrukturfunktion wie „einfache Variablen“ (mittels `var`-Schlüsselwort) angegeben.

```

1 function Computer()
2 {
3     var betriebssystem = "Windows 7"; // Dies ist eine private Eigenschaft
4 }
5
6 var meinComputer = new Computer();
7 alert(meinComputer.betriebssystem); // Gibt undefined aus

```

## Objekt-Arrays

Arrays mit Objekten als Werte sind von Arrays mit z. B. Zahlen als Werte in erster Linie nicht zu unterscheiden. Jedoch gibt es bei Arrays mit Objekten einige Probleme. Das Problem bei der Verwendung von Objekt-Arrays ist, dass **Suchfunktionen** wie `indexOf()` und `lastIndexOf()` in den meisten Fällen nicht mehr funktionieren. Dies kommt daher, da wir zumeist die Referenz auf das Objekt nicht mehr haben. Eine Suche mit einem Objekt, welches die gleichen Werte hat, bleibt erfolglos, da die Referenz eine andere ist. Die Suche sollte also lediglich mit Hilfe der Eigenschaften durchgeführt werden. Eine solche Funktion gibt es jedoch nicht, d. h. wir müssen die **Suche manuell ausführen**. Dies kann z. B. mittels einer Schleife durchgeführt werden. Das folgende Beispiel zeigt das Ergebnis der erfolglosen Suche mit `indexOf()` und anschließend die „manuelle Suche“ mittels einer Schleife:

```

1 var computerListe = new Array(
2     {
3         marke       : "Acer",
4         modell      : "Aspire E15",
5         betriebssystem : "Windows 7"
6     },
7     {
8         marke       : "Asus",
9         modell      : "R752SA",
10        betriebssystem : "Windows 10"
11    },
12    {
13        marke       : "Toshiba",
14        modell      : "Satellite L70",
15        betriebssystem : "Windows 10"
16    }
17 );
18
19 document.write("Suche mit indexOf: " + computerListe.indexOf(
20     {
21         marke       : "Asus",
22         modell      : "R752SA",
23         betriebssystem : "Windows 10"
24     }
25 ) + "<br />");
26
27 for (var i = 0; i < computerListe.length; i++)
28 {

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Objektorientierung](#)

```

29     if (computerListe[i].marke == "Asus" && computerListe[i].modell == "R752SA" && computerListe[i].betriebssystem ==
"Windows 10")
30     {
31         document.write("Suche mit Schleife: " + i);
32         break;
33     }
34 }

```

Benötigen wir lediglich das Objekt selbst (z. B. zum Abrufen weiterer Eigenschaften oder aufrufen von Objektfunktionen) und nicht dessen Index im Array, so können wir das Array mittels der `filter()`-Funktion filtern. Daraus resultiert dann ein wesentlich kürzerer Code im Vergleich zu der oben programmierten Schleife.

```
1 | computerListe.filter(x => x.marke == "Asus" && x.modell == "R752SA" && x.betriebssystem == "Windows 10");
```

**Übrigens:** Die Sortierung von Objekt-Arrays mit Hilfe der `sort()`-Funktion funktioniert ebenfalls nicht mehr wie gewohnt, da der Interpreter nicht weiß, wie er die Objekte sortieren soll. Um Objekte weiterhin sortieren zu können, müssen wir der `sort()`-Funktion als Parameter eine Funktion übergeben. Diese **Sortierungsfunktion** erhält immer zwei Übergabewerte (Wert a und Wert b) und kann dann einen negativen Wert, 0 oder einen positiven Wert zurückgeben. Ein negativer Wert bedeutet, dass a vor b einsortiert werden muss. 0 bedeutet, dass die Objekte a und b gleich sind. Ein positiver Wert führt dazu, dass a nach b einsortiert wird.

## Objekterweiterung

Der Aufbau von Objekten, welcher mit einer Konstruktorfunktion deklariert wurden oder sogar zum Sprachvorrat von JavaScript gehören, kann im Nachhinein mittels der einfachen Zuweisung, wie wir diese bei Einzelobjekten angewendet haben, nicht mehr erweitert werden. Doch natürlich können auch solche Objekte erweitert werden. Hierfür müssen wir den sogenannten **Prototypen** verändern. Hierzu notieren wir den Objektnamen, gefolgt von einem Punkt, dem Schlüsselwort `prototype` und einem weiteren Punkt. Anschließend muss lediglich noch der Name der neuen Funktion bzw. Eigenschaft angegeben werden und dieser ein Funktionsausdruck oder ein Wert zugewiesen werden.

Das folgende Beispiel erweitert das Objekt `Number` und somit den Datentyp `Number` um die Funktion `padNumber()`, mit welcher es nun möglich ist, Zahlen mit führenden Nullen darzustellen. Diese Variante der Objekterweiterung ist sehr nützlich und wird gerade zum Erweitern der Funktionen von vorhandenen Datentypen verwendet. Wie Sie im Beispiel sehen können, gibt `this` innerhalb des Funktionsausdrucks die Zahl des Objekts zurück.

```

1 | Number.prototype.padNumber = function (pad)
2 | {
3 |     var str = this.toString();
4 |     while (str.length < pad)
5 |         str = "0" + str;
6 |     return str;
7 | };
8 |
9 | var zahl = 12;
10 | for (var i = 0; i < 10; i++)
11 |     document.write(i + ": " + zahl.padNumber(i) + "<br />");

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

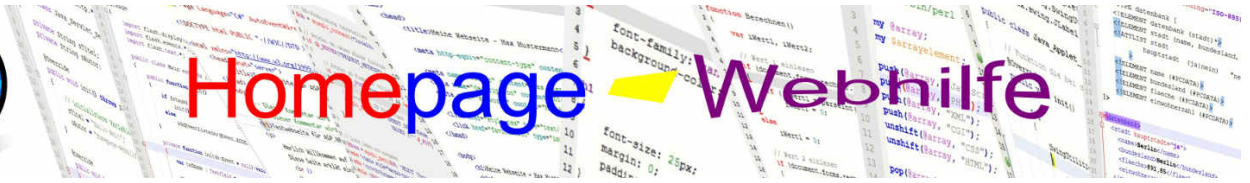
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Exceptions und Debugging](#)

## Exceptions und Debugging

Nachdem wir nun schon die wichtigsten Grundlagen von JavaScript kennengelernt haben, wollen wir uns vorerst mit dem Thema Exceptions und Debugging beschäftigen, bevor wir uns den fortgeschrittenen Themen widmen.

Als **Exceptions** (engl. für *Ausnahmen*) werden Fehler bezeichnet, die während dem Ausführen des JavaScript-Codes auftreten. Zu diesen Fehlern zählen z. B. nicht existierende Variablen oder Funktionen. Tritt eine Exception auf, welche nicht abgefangen wurde, so wird die **Ausführung des Skripts beendet** und eine Fehlermeldung in der Konsole der Webentwickler-Tools ausgegeben.

Unter Debugging versteht man im Allgemeinen die **Behebung von Fehlern** in einem Skript oder Programm. Um diese Fehler zu finden, wird der sogenannte **Debugger** verwendet. Der Debugger ist ein Tool, welches zumeist in einer Entwicklungsumgebung integriert ist. Er erlaubt es, das Skript zeilenweise durchzugehen, um festzustellen, an welcher Stelle der Fehler auftritt. Zusätzlich können währenddessen die aktuellen Werte von Variablen angezeigt werden. Da der Programmcode von JavaScript über einen Interpreter im Browser ausgeführt wird und somit dem Browser als Quelltext vorliegt, wird JavaScript-Code üblicherweise, über den in den meisten Browsern integrierten Debugger, debuggt.

### Inhalt dieser Seite:

1. Konsole
2. Debugger
3. Exceptions abfangen
4. Exceptions auslösen

### Konsole

Die Konsole ist ein Ausgabe-Fenster, welches sich innerhalb der sogenannten **Webentwickler-Tools** im Browser wiederfindet. Die Webentwickler-Tools können entweder über das Menü oder meistens auch mittels der Taste *F12* aufgerufen werden. In den meisten Browsern (u. a. Microsoft Internet Explorer, Mozilla Firefox und Google Chrome) werden die Webentwickler-Tools als eigenständiges oder als angedocktes Fenster mit Registerkarten (Tabs) dargestellt. Unter diesen Registerkarten sollte sich auch eine mit der Aufschrift „Konsole“ oder „Console“ befinden. Dort werden dann z. B. JavaScript-Fehler (sowohl Laufzeitfehler, als auch Syntaxfehler) angezeigt. Für Entwicklungs-Zwecke ist es manchmal ganz praktisch, eine Ausgabe durchzuführen, ohne dabei den Seiteninhalt der Webseite zu verändern. Genau hierfür ist es auch möglich, dass wir über unser JavaScript-Programm eine Ausgabe in diesen Konsolen-Fenster durchführen. Hierfür kann die Funktion `console.log()` verwendet werden. Des Weiteren kann innerhalb des Konsole-Fensters JS-Programmcode manuell ausgeführt werden. Dadurch können wir z. B. auch eine Funktion unseres Programms aktiv aufrufen.

```
1 | console.log("Glückwunsch! Sie haben die Konsole gefunden.");
```



### Debugger

Der Debugger ist, wie bereits oben schon erwähnt, ein Tool, mit welchem wir unseren **Code analysieren** können, um somit im Programm enthaltene **Fehler zu beheben**. Gefunden werden kann der Debugger ebenfalls in den Webentwickler-Tools.



Im Debugger können wir mit Hilfe eines Klicks auf die Zeilennummer einen sogenannten **Breakpoint** setzen. Bei einem Breakpoint (engl. für *Haltepunkt*) hält der Interpreter die Ausführung des Codes an. Dadurch ist es nun möglich, ab dem gewählten Punkt den Code zu debuggen. Wird der Code den Sie debuggen wollen beim Laden automatisch ausgeführt, müssen Sie nach dem Setzen des Breakpoints die Seite neu laden. Die Breakpoints bleiben dabei erhalten. Innerhalb des Debugger-Fensters wird zudem der **Wert von aktuell verwendeten Variablen** angezeigt. Weitere Variablen können jedoch zur „Überwachung“ ohne weiteres manuell hinzugefügt werden. Für die einfache Ausgabe eines Variablenwerts (ohne das Skript dabei debuggen zu wollen), können Sie einfach den Variablennamen in der Eingabezeile im Konsolen-Fenster eingeben. Sie erhalten dann als Rückgabe den Wert der Variable.

### Exceptions abfangen

In einigen Fällen lassen sich bestimmte Laufzeitfehler nicht vermeiden, da diese z. B. durch eine „externe Funktion“ einer fremden Funktion (z. B. einer Library) ausgelöst wird. Um solche Fehler abzufangen, ohne dabei den kompletten Programmablauf zu stoppen, gibt es in JavaScript die sogenannte **try-catch**-Anweisung.

Der Syntax ist wie folgt: Wir notieren das Schlüsselwort `try` gefolgt von einem Block mit Code. Nun folgt das Schlüsselwort `catch`, welches wieder einen Block mit sich bringt. Nach dem Schlüsselwort `catch` muss noch ein rundes Klammernpaar notiert werden, in welchem ein frei verfügbarer Variablenname angegeben werden muss. Über diese Variable können wir innerhalb des `catch`-Blocks die **Meldung des Fehlers** abrufen, welche bisher sonst immer in der Konsole angezeigt wurde.

Der Ablauf ist wie folgt: Wird innerhalb des Codes im `try`-Block eine Exception ausgelöst, so wird nicht etwa das Skript gestoppt, sondern lediglich die Ausführung des Codes im `try`-Block. Anschließend erfolgt ein Sprung in den `catch`-Block. Der Code im `catch`-Block kann nun zur **Fehlerbehandlung** genutzt werden. Dies kann z. B. so aussehen, dass wir auf unserer Seite eine Fehlermeldung ausgeben. Vorteil dieser Variante ist, dass weiterer Code (außerhalb dieses `try`-Blocks) ausgeführt werden kann. Zudem gilt die Verwendung von `try-catch` als „saubere Lösung“ zum Abfangen und Behandeln von Fehlerfällen.

```
1 | try
2 | {
3 |     // Funktion existiert nicht!
4 |     document.writeText("Hallo!");
5 | }
6 | catch (err)
7 | {
8 |     document.write("Ein Fehler ist aufgetreten: " + err);
9 | }
```



Die `try-catch`-Anweisung kann am Ende um einen `finally`-Block erweitert werden. Der Code im `finally`-Block wird nach vollständiger Ausführung des `try`-Blocks oder nach Ausführung des `catch`-Blocks ausgeführt. Der `finally`-Block ist optional und kann somit ohne weiteres weggelassen werden, wenn dieser nicht benötigt wird.

```
1 | try
2 | {
```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Exceptions und Debugging](#)

```

3 // Funktion existiert nicht
4 document.writeText("Hallo!<br />");
5 }
6 catch (err)
7 {
8     document.write("Ein Fehler ist aufgetreten: " + err + "<br />");
9 }
10 finally
11 {
12     document.write("Code 1 ausgeführt!<br />");
13 }
14 document.write("<br />");
15
16 try
17 {
18     // Funktion existiert
19     document.write("Hallo!<br />");
20 }
21 catch (err)
22 {
23     document.write("Ein Fehler ist aufgetreten: " + err + "<br />");
24 }
25 finally
26 {
27     document.write("Code 2 ausgeführt!<br />");
28 }
29 }
    
```



## Exceptions auslösen

Exceptions können nicht nur vom Interpreter ausgelöst werden, sondern auch direkt vom JavaScript-Code und somit von unserem eigenen Code. Hierfür notieren wir das Schlüsselwort `throw` gefolgt von einem Wert, welcher dem `catch`-Block übergeben werden soll. Dies ist oftmals eine Zeichenkette (siehe Beispiel).

Natürlich ist es nur sinnvoll, Exceptions selbst auszulösen, wenn diese dann auch wieder abgefangen werden. Dies klingt im ersten Moment vielleicht etwas verwirrend, doch gerade bei größeren Projekten ist dieses Verfahren sinnvoll.

```

1 var a = 7;
2 var b = 0;
3
4 try
5 {
6     if (b == 0)
7         throw "Variable b darf nicht 0 sein!";
8
9     document.write("Ergebnis: " + (a / b));
10 }
11 catch (err)
12 {
13     document.write("Ein Fehler ist aufgetreten: " + err + "<br />");
14 }
    
```



**Übrigens:** In fast allen Webentwickler-Tools lässt sich auch noch ein sogenannter **Inspektor** finden, mit welchem es möglich ist, die Baumstruktur eines HTML-Dokuments durchzugehen und auch das **HTML-Dokument** zu verändern. Auch das Ändern von **CSS-Eigenschaften** für einzelne Elemente, alle Elemente oder für Klassen und IDs ist möglich.

**Wichtig:** Die Erklärung zu den Webentwickler-Tools, zum Debugger und zur Konsole ist allgemein gehalten und kann sich von Browser zu Browser im konkreten Ablauf und der Beschriftung unterscheiden. Wir empfehlen Ihnen die Verwendung von Mozilla Firefox oder Google Chrome zum Debuggen von JavaScript-Code. Bei Bedarf lesen Sie sich ggf. die Hilfe der Browser zum Thema Webentwickler-Tools durch.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [DOM und BOM](#)

## DOM und BOM

Das DOM und BOM sind Modelle, welche als Schnittstelle genutzt werden können, um auf das Dokument und den Browser zuzugreifen.

Das **DOM** (*Document Object Model*) erlaubt mittels einer Skript- oder Programmiersprache (z. B. JavaScript) den Zugriff auf das **Dokument** (z. B. HTML-Dokument). Das DOM kann als Schnittstelle angesehen werden, welches auf die HTML-Elemente (welche in JS als Objekte vorliegen) zugreifen kann und somit **Elemente hinzufügen, verändern und sogar löschen** kann. Diese Änderung gilt dabei immer nur für die aktuelle Sitzung, wird nicht gespeichert und geht somit beim Wechseln auf eine andere Seite verloren. Um auf das DOM zuzugreifen, benötigen wir das Objekt `document`. Bisher haben wir aus diesem Objekt immer nur die Funktion `write()` verwendet. Diese werden wir jedoch in Zukunft (mal abgesehen von Testskripten) eigentlich kaum mehr verwenden (evtl. nur noch in Verbindung mit einem [PopUp-Fenster](#)).

Das **BOM** (*Browser Object Model*) ist ein Modell zum Zugriff auf den **Browser**. Das BOM wird durch das Objekt `window` gekennzeichnet. Dieses Objekt besitzt unter anderem das Unterobjekt `location` (dazu später [mehr](#)). Auch `document` ist ein Unterobjekt von `window`, somit kann das DOM als Bestandteil des BOMs angesehen werden. Auf einige Eigenschaften und Funktionen des BOMs gehen wir in den Themen [Event-Handling](#), [Fenstersteuerung](#) und [Zeitsteuerung](#) ein.

**Übrigens:** Die Funktionen `alert()`, `confirm()` und `prompt()` sind Funktionen des `window`-Objekts und können somit auch über `window.alert()`, `window.confirm()` und `window.prompt()` aufgerufen werden. Da `window` das Wurzelement ist, kann es jedoch bei der Angabe weggelassen werden. Die Funktionen `parseInt()`, `parseFloat()` und `isNaN()` sind hingegen objektlose Funktionen. Ein Aufruf wie `window.parseInt()` wäre falsch, auch wenn dieser vermutlich in den meisten Browsern fehlerfrei ausgeführt werden würde.

## Elemente auswählen

Um Elemente von einem HTML-Dokument auszuwählen, gibt es 4 Funktionen: `getElementById()`, `getElementsByClassName()`, `getElementsByName()` und `getElementsByTagName()`. `getElementById()` gibt dabei das Element mit der angegebenen **ID** (`id`-Attribut im HTML-Code) zurück. Alle 3 anderen Funktionen geben ein Objekt vom Typ `NodeList` zurück. Dabei handelt es sich um eine Art von Array. Der Zugriff kann ebenfalls über den Index und die eckigen Klammern erfolgen. Die Länge kann auch hier ebenfalls mit der `length`-Eigenschaft abgerufen werden. Mittels der Funktion `getElementsByClassName()` lassen sich alle Elemente selektieren, die den angegebenen **Klassennamen** haben. `getElementsByName()` selektiert alle Elemente mit dem angegebenen Namen als Wert des `name`-Attributs. `getElementsByTagName()` selektiert Elemente an Hand des **Element- bzw. Tag-Namens**. So selektiert `getElementsByTagName("p")` alle Paragraphen-Elemente.

Die mit Hilfe der 4 genannten Methoden selektierten Elemente liegen in JavaScript als `Node`-Objekt vor. Dieses `Node`-Objekt kann nun mittels Funktionen und Eigenschaften verändert werden. Den (HTML-)Inhalt eines Elements können wir mittels der Eigenschaft `innerHTML` abrufen oder verändern. Weitere Attribute können mittels dem Syntax `element.attributName = attributWert` oder über die Funktion `setAttribute()` (z. B. `element.setAttribute("attributName", attributWert)`) geändert werden.

```
1 | <h1 id="titel"></h1>
2 | 
1 | document.getElementById("titel").innerHTML = "Homepage-Webhilfe";
2 | document.getElementById("bild").title = "Das Logo von Homepage-Webhilfe!";
```

Wir können mittels des `Node`-Objekts auch auf dessen Unterelemente zugreifen. Hierfür dient die Eigenschaft `childNodes`, welche ein `NodeList`-Objekt zurückgibt. Über die Funktionen `appendChild()` und `removeChild()` können wir unserem Element weitere **Unterelemente hinzufügen** oder ein **Unterelement entfernen**.

## Style-Eigenschaften

Auf Style-Eigenschaften eines einzelnen Elements können wir ebenfalls mittels des `Node`-Objekts zugreifen. Hierfür müssen wir uns in das Objekt `CSSStyleDeclaration` begeben, welches sich hinter der Eigenschaft `style` des `Node`-Objekts befindet. Die Eigenschaften des `CSSStyleDeclaration`-Objekts, um CSS-Eigenschaften abzurufen und zu ändern, haben den gleichen Namen wie in CSS. Die Schreibweise erfolgt dabei jedoch immer ohne Bindestriche, sondern mit einer Abwandlung der „Camel Case“-Schreibweise. Ein Zugriff auf die CSS-Eigenschaft `margin-top` kann in JavaScript mittels der Eigenschaft `marginTop` erfolgen. Die Eigenschaft `border-right-style` wird hiermit zu `borderRightStyle`. Die Eigenschaft `color` verändert sich hingegen vom Namen nicht.

```
1 | <div id="rechteck" style="width: 200px; height: 200px; background-color: red;"></div>
1 | document.getElementById("rechteck").style.backgroundColor = "lime";
```

## Formulare

Formulare bzw. deren Elemente können an Stelle mit der Funktion `getElementsByName()` auch über eine einfachere Variante angesprochen werden. Hierzu nutzen wir das Unterobjekt `forms` des `document`-Objekts. Nun können wir unser Formular entweder mittels eines Index (`document.forms[0]`, `document.forms[1]`, ...) oder über den Namen (`document.forms.meinFormular1`, `document.forms.meinFormular2`) ansprechen. Auf Unterelemente des Formulars können wir nun ebenfalls mit Hilfe des Namens (`name`-Attribut) zugreifen. In den Elementen selbst stehen uns nun weitere Eigenschaften je nach Typ zur Verfügung: z. B. `value` (eingegabener oder aktueller Wert bzw. Wert des `value`-Attributs), `checked` (für Kontrollkästchen: `true` bedeutet Häkchen ist gesetzt), `disabled` (entspricht dem `disabled`-Attribut, `true` bedeutet Feld ist deaktiviert), `selectedIndex` (für Auswahllisten: Index der ausgewählten Eigenschaft von den Optionen oder -1) und `options` (für Auswahllisten: Liste mit allen Optionen der Auswahlliste).

```
1 | <form name="formular">
2 |   Jahr: <input type="number" min="2000" name="jahr" />
3 | </form>
1 | document.forms.formular.jahr.value = (new Date()).getFullYear();
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [DOM und BOM](#)



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

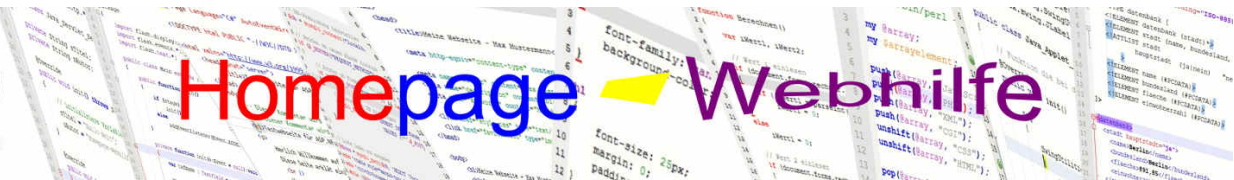
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Event-Handling](#)

## Event-Handling

Unter Event-Handling versteht man den Umgang mit bzw. die Verwendung von sogenannten Events. Events dienen zum Steuern des Programmflusses. Events werden ausgelöst, wenn ein bestimmtes **Ereignis auftritt**. Daraufhin wird dann die angegebene Eventfunktion (auch als **Event-Handler** oder *listener* bezeichnet) aufgerufen. Mit Hilfe von Events können Sie also direkt auf bestimmte Aktionen, die der Benutzer (z. B. Klicken) auslöst, reagieren.

Der Programmcode bzw. die Funktion, welche bei einem Event ausgelöst werden soll, kann dabei auf 3 verschiedene Arten zugewiesen werden, wovon je nach Situation nicht alle Varianten verwendet werden können.

Die einfachste Variante ist die direkte Notation des Programmcodes (dies kann z. B. ein Funktionsaufruf sein) innerhalb des **Attributs des jeweiligen HTML-Elements**. Die Attributnamen setzen sich dabei aus dem Wort `on` und dem Namen des Events zusammen. Daraus entsteht dann z. B. `onclick`, `onmouseover` und `onkeydown`.

```
1 | <p onclick="MeineEventFunktion();">Hier steht Text ...</p>
```

Innerhalb des JavaScript-Codes können wir einem Event ebenfalls eine Eventfunktion zuweisen. Hierfür benötigen wir innerhalb des Codes das `Node`-Objekt des Elements und die Eigenschaft für das jeweilige Event. Der Name der **Event-Eigenschaften** entspricht dem Attributnamen im HTML-Code. Als Wert wird die Funktion (ohne runde Klammern) oder ein Funktionsausdruck angegeben. Der angegebenen Funktion wird beim Aufruf ein Parameter übergeben, welcher Informationen zum Event enthält (dazu später mehr).

```
1 | function MeineEventFunktion(eventObjekt)
2 | {
3 |     // Mein Event-Code
4 | }
5 |
6 | document.getElementById("meinElement").onclick = MeineEventFunktion;
```

An Stelle der direkten Zuweisung der Eigenschaft können wir auch die Funktion `addEventListener()` verwenden. Als Parameter erwartet die Funktion zum einen den Namen des Events (hierbei ohne `on`) und zum anderen die Funktion, welche ausgeführt werden soll (auch hier erfolgt die Angabe als Referenz und somit ohne runde Klammern oder direkt mittels eines Funktionsausdrucks). Der angegebenen Funktion wird auch hier ebenfalls ein Event-Objekt als Parameter übergeben. Der Vorteil von `addEventListener()` im Vergleich zur direkten Zuweisung ist, dass hierdurch **mehrere Event-Handler** an das gleiche Ereignis gebunden werden können. Des Weiteren kann mittels der Funktion `removeEventListener()` ein registrierter Event-Handler ohne weiteres wieder entfernt werden.

```
1 | function MeineEventFunktion(eventObjekt)
2 | {
3 |     // Mein Event-Code
4 | }
5 |
6 | document.getElementById("meinElement").addEventListener("click", MeineEventFunktion);
```

## Maus-Events

Maus-Events dienen dazu, auf verschiedene Ereignisse, welche mittels der Maus (Taste oder Bewegung) ausgelöst wurden, zu reagieren. Das einfachste Event ist das `click`-Event. Dies tritt auf, wenn auf eine der **Maustasten** gedrückt wird. In einigen Fällen würde sich das `click`-Event auch mittels eines Links ersetzen lassen. Hierfür notieren wir im `href`-Attribut des `a`-Elements `javascript:` gefolgt von dem auszuführenden JS-Code (z. B. ein Funktionsaufruf). Natürlich sollte Ihnen bewusst sein, dass Sie mit Hilfe des `click`-Events viel mehr Möglichkeiten haben: So ist es z. B. möglich, abzufragen, welche Taste denn überhaupt gedrückt wurde. Um die Informationen des Events zu erhalten, müssen Sie innerhalb des Attributs auf die Variable `event` zugreifen. Die bessere Lösung wäre jedoch die Zuweisung der Eventfunktion mittels JavaScript, da wir hierdurch automatisch das **Event-Objekt** übergeben bekommen. Auf die Eigenschaften des Objekts gehen wir nachher noch genauer ein.

```
1 | <a onclick="alert('Hallo!')">Bitte klicken!</a>
2 | <br />
3 | <a href="javascript:alert('Hallo!')">Bitte klicken!</a>
```

Weitere JS-Events, welche die Maus betreffen, sind `mousemove`, `mouseout`, `mouseup` und `mousedown`. `mousemove` tritt ein, wenn die Maus innerhalb des Elements, welchem das Event zugewiesen ist, bewegt wird. `mouseout` tritt hingegen auf, wenn der Mauszeiger das Element verlassen hat. Die Events `mouseup` und `mousedown` betreffen die Maustasten und treten ein, wenn eine Maustaste gedrückt wird (`mousedown`) bzw. die Maustaste wieder losgelassen wird (`mouseup`).

Alle Events, welche die Maus (Tasten und Bewegung) betreffen, geben der Ereignisfunktion ein `MouseEvent`-Objekt zurück. Dieses Objekt verfügt nun über einige Eigenschaften, mit welchen wir Informationen über das Ereignis abrufen können. Die Eigenschaft `buttons` gibt eine Bitmaske zurück, welche den **Status der Maustasten** enthält. Bit 0 gibt dabei den Status der linken Maustaste zurück. Bit 1 gibt den Status der rechten Maustaste zurück. Bit 2 bezieht sich auf die mittlere Taste bzw. die Mausrad-Taste. Der Wert 5 z. B. gibt also an, dass die linke Maustaste und die mittlere Maustaste gedrückt wurde (Binärdarstellung 0000 0101). Über die Eigenschaften `pageX` und `pageY` lässt sich die **Position der Maus** relativ zur Seite ermitteln. Mit Hilfe der Eigenschaft `target` können wir auf das Element (in Form des `Node`-Objekts) zugreifen, welches das Ereignis ausgelöst hat. Über die Eigenschaften `offsetLeft` und `offsetTop` lässt sich nun in Kombination mit den oben genannten Eigenschaften `pageX` und `pageY` die **Position der Maus innerhalb des Elements** kalkulieren (siehe Beispiel).

```
1 | <div id="rechteck" style="width: 200px; height: 200px; background-color: red;" onmouseout="MausVerlassen()"></div>
2 | <p>Mausposition: <span id="mausPosition"></span></p>
3 | <p>Maustasten: <span id="mausTasten">0</span></p>
4 |
5 | function MausBewegung(evt)
6 | {
7 |     document.getElementById("mausPosition").innerHTML = (evt.pageX - evt.target.offsetLeft) + " " + (evt.pageY -
8 |     evt.target.offsetTop);
9 | }
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Event-Handling](#)

```

5
6 function MausVerlassen()
7 {
8     document.getElementById("mausPosition").innerHTML = "-";
9 }
10
11 function MausTaste(evt)
12 {
13     document.getElementById("mausTasten").innerHTML = evt.buttons;
14 }
15
16 document.getElementById("rechteck").onmousemove = MausBewegung;
17 document.getElementById("rechteck").addEventListener("mouseup", MausTaste);
18 document.getElementById("rechteck").addEventListener("mousedown", MausTaste);

```



## Tastatur-Events

Um auf Tastatur-Events zu reagieren, gibt es die drei Events: `keydown`, `keyup` und `keypress`. `keydown` und `keyup` sind von der Funktionsweise mit `mousedown` und `mouseup` zu vergleichen. So löst `keydown` aus, wenn eine Taste gedrückt wurde und `keyup`, wenn diese wieder losgelassen wird. Das `keypress`-Event löst hingegen aus, wenn eine Taste bzw. vielmehr ein Zeichen eingegeben wurde. Somit löst `keypress` also kein Event aus, wenn z. B. auf die Taste `Ctrl` oder `Alt` gedrückt wird. Die Verwendung von `keypress` sollte jedoch mit Vorsicht verwendet werden, da es keine klare Regelung gibt und die Browser das Ereignis teilweise unterschiedlich interpretieren. Alle Tastaturereignisse geben der Ereignisfunktion ein `KeyboardEvent`-Objekt mit. Über die Eigenschaft `key` lässt sich eine Zeichenkette abrufen, welche die **Bezeichnung der aktuellen Taste** enthält. Die Eigenschaft `target` ist hier, so wie bei allen anderen Event-Objekten auch, ebenfalls verfügbar. Das untere Beispiel zeigt eine einfache Seite, mit welcher alle drei Events getestet werden können. Das Event wird dabei auf die ganze Seite (`body`-Element) registriert.

```

1 function TasteDown(evt)
2 {
3     document.getElementById("keyDown").innerHTML = evt.key;
4 }
5
6 function TasteUp(evt)
7 {
8     document.getElementById("keyUp").innerHTML = evt.key;
9 }
10
11 function TastePress(evt)
12 {
13     document.getElementById("keyPress").innerHTML = evt.key;
14 }
15
16 document.body.onkeydown = TasteDown;
17 document.body.onkeyup = TasteUp;
18 document.body.onkeypress = TastePress;

```



## Formular-Events

Für Formulare gibt es noch ein weiteres nützliches Event, welches wir hier vorstellen wollen. Das `change`-Event tritt ein, wenn ein Wert, eine Auswahl oder ein Status (bei Radio-Buttons oder Checkboxes) geändert wurde (z. B. durch eine Eingabe, einen Klick oder eine Auswahl). Verwendet werden kann das Event für `input`-Elemente, `textarea`-Elemente, aber auch für `select`-Elemente (Auswahllisten). Für das Formular selbst (`form`-Element) gibt es noch die Events `submit` (beim Versand des Formulars) und `reset` (beim Zurücksetzen des Formulars). Beide Events gelten als „**Cancelable**“, wodurch es möglich ist, den Versand oder das Zurücksetzen des Formulars abzubrechen. Hierfür muss unsere Ereignisfunktion `true` (Aktion erlaubt bzw. weiterführen) oder `false` (Aktion abbrechen) zurückgeben. Wenn wir die Ereignisfunktion innerhalb der entsprechenden Attribute des `form`-Tags aufrufen, müssen wir dort zusätzlich noch `return` vor dem Funktionsnamen notieren (siehe Beispiel).

```

1 <form name="formular" onsubmit="return PruefeVersand();" onreset="return PruefeZuruecksetzen();" >
2   <input type="checkbox" name="aktivierung" onchange="AendereTextfeld()" />
3   <input type="text" name="textfeld" disabled="disabled" />
4   <br />
5   <select name="auswahl" onchange="ZeigeAuswahl()" >
6     <option value="H">HTML</option>
7     <option value="C">CSS</option>
8     <option value="J">JavaScript</option>
9     <option value="A">ActionScript</option>
10  </select>
11  <br />
12  <input type="reset" value="Zuruecksetzen" /><input type="submit" value="Senden" />
13 </form>
14 <br />
15 <p>Ihre Auswahl: <span id="auswahl"></span></p>
1
2 function PruefeVersand()
3 {
4     return confirm("Sind Ihre Angaben korrekt?");

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Event-Handling](#)

```

4  }
5
6  function PruefeZuruecksetzen()
7  {
8      return confirm("Wollen Sie die Eingaben wirklich löschen?");
9  }
10
11 function ZeigeAuswahl()
12 {
13     document.getElementById("auswahl").innerHTML =
14     document.forms.formular.auswahl.options[document.forms.formular.auswahl.selectedIndex].innerHTML + " (" +
15     document.forms.formular.auswahl.value + ")";
16 }
17
18 function AendereTextfeld()
19 {
20     document.forms.formular.textfeld.disabled = !document.forms.formular.aktivierung.checked;
21 }

```

**Übrigens:** Natürlich könnten wir im obigen Beispiel die Event-Handler auch mittels der JavaScript-Zuweisung bzw. der Funktion `addEventListener()` registrieren, jedoch haben wir keinen Bedarf an dem Ereignis-Objekt (dies ist nämlich nur vom Typ `Event` und besitzt abgesehen von der `target`-Eigenschaft keine für uns wichtigen Eigenschaften), wodurch der obige Code etwas kürzer und einfacher erscheint.

## Seitenlade-Events

Das `load`-Event, welches zu den globalen Events gehört und somit dem `window`-Objekt zugewiesen werden muss, wird ausgelöst, wenn die aktuelle Seite mit allen Bestandteilen (wie z. B. Grafiken) geladen wurde. Der JS-Code, welcher nach dem Laden der Seite sofort aufgerufen werden soll, sollte innerhalb dieses Eventhandlers ausgeführt werden. JavaScript-Code außerhalb von Ereignisfunktionen bzw. Funktionen sollte vollständig vermieden werden, da dadurch nicht sichergestellt ist, dass die Seite und das DOM der Seite bereits vollständig geladen ist.

```

1  function SeitenladeEvent()
2  {
3      document.write("Seite erfolgreich geladen!");
4  }
5
6  window.addEventListener("load", SeitenladeEvent);

```

**Übrigens:** Grundsätzlich geben alle Ereignisse von JavaScript an die Ereignisfunktionen (sofern diese nicht über eines der HTML-Attribute aufgerufen wird) immer ein `Event`-Objekt mit. Jedoch können wir bei der Deklaration der Funktion den Parameter für das `Event`-Objekt weglassen, falls wir diesen nicht benötigen.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Fenstersteuerung](#)

## Fenstersteuerung

In diesem Thema beschäftigen wir uns hauptsächlich mit dem BOM (*Browser Object Model*). Das wichtigste Objekt des BOMs ist `window`. Die darin enthaltenen Eigenschaften erlauben es, **Informationen über die Seite** abzufragen, eine **Weiterleitung** durchzuführen oder weitere Fenster in Form eines **PopUps** zu öffnen und zu kontrollieren.

### Inhalt dieser Seite:

1. Seiteninformationen
2. Weiterleitung
3. PopUp-Fenster

## Seiteninformationen

Innerhalb des `window`-Objekts gibt es einige Eigenschaften, mit welchen wir Informationen über den Browser bzw. die angezeigte Seite abrufen können. Ein wichtiges Unterobjekt von `window` ist `location`. Die darin enthaltenen Informationen enthalten die URL bzw. deren Bestandteile der aktuellen Seite. Welche Informationen die Eigenschaften enthalten, lässt sich wohl am einfachsten feststellen, wenn man einfach den untenstehenden Code ausführt oder das untenstehende Beispiel aufruft:

```

1 document.write("innerHeight: " + window.innerHeight + "<br />");
2 document.write("innerWidth: " + window.innerWidth + "<br />");
3 document.write("<br />");
4 document.write("location.protocol: " + window.location.protocol + "<br />");
5 document.write("location.port: " + window.location.port + "<br />");
6 document.write("location.host: " + window.location.host + "<br />");
7 document.write("location.hostname: " + window.location.hostname + "<br />");
8 document.write("location.origin: " + window.location.origin + "<br />");
9 document.write("location.pathname: " + window.location.pathname + "<br />");
10 document.write("location.search: " + window.location.search + "<br />");
11 document.write("location.hash: " + window.location.hash + "<br />");
12 document.write("location.href: " + window.location.href + "<br />");
    
```

Die Bedeutung von `window.location.search` und `window.location.hash` geht aus dem obigen Beispiel nicht vor, weshalb wir diese hier nochmals aufgreifen möchten. `window.location.search` enthält die sogenannte **Suchanfrage**, welche mittels GET-Parameter übergeben wurde. Dieser Teil sieht z. B. wie folgt aus: `?name1=wert1&name2=werte`. `window.location.hash` hingegen enthält den **Ankername** inkl. dem `#`-Zeichen, welcher der Seite übergeben wurde (z. B. `#bild`). `window.location.port` gibt in den meisten Fällen eine leere Zeichenkette zurück, da Port 80 (für HTTP) bzw. Port 443 (für HTTPS) bereits aus dem Wert von `window.location.protocol` hervorgeht.

## Weiterleitung

Die Eigenschaften des `location`-Objekts lassen sich natürlich nicht nur abfragen, sondern auch zuweisen. Dadurch können wir mittels JavaScript eine Umleitung auf eine andere Seite durchführen.

```

1 window.location.href = "/";
    
```

**Wichtig:** Die Verwendung von `window.location.href` gilt als sicherste Variante zum Weiterleiten mit JavaScript. Natürlich könnten Sie aber auch `window.location.pathname` zum Weiterleiten verwenden, wenn Sie innerhalb der gleichen Domain bleiben. Als Wert für `window.location.href` sind neben relativen und absoluten Pfadangaben auch Weiterleitungen mittels einer URL-Angabe auf andere Domains möglich.

## PopUp-Fenster

Mit Hilfe der Funktion `open()` des `window`-Objekts ist es möglich, ein **neues Browserfenster** (und somit u. U. ein PopUp) zu öffnen. Dafür können der Funktion ein oder mehrere Parameter übergeben werden. Im ersten Parameter übergeben wir die URL der anzuzeigenden Seite. Wird hier lediglich eine leere Zeichenkette übergeben, so wird ein leeres Fenster geöffnet (Seite `about:blank`). Der zweite Parameter ersetzt das `target`-Attribut, welches wir bereits von [HTML-Links](#) kennen oder gibt dem Fenster einen Namen. Die Angabe dieses Parameters wird kaum gebraucht, muss jedoch als leere Zeichenkette angegeben werden, sofern der dritte Parameter benötigt wird. Im dritten Parameter können wir dem Fenster einige **Attribute** (in diesem Fall eine Art von Einstellungen) mitgeben, welche im Format `name1=wert1,=name2=wert2` angegeben werden müssen. Die Attribute `width` und `height` legen die Fenstergröße fest. Mit den Attributen `left` und `top` lässt sich das Fenster platzieren. Über das Attribut `scrollbars` und die Werte `yes` und `no` lässt sich die Anzeige von Scrollbalken steuern (diese werden in den meisten Browsern bei PopUps standardmäßig ausgeblendet).

```

1 function PopUpOeffnen()
2 {
3     window.open("/", "", "width=350,height=400,top=100,left=100,scrollbars=yes");
4 }
    
```

**Übrigens:** Die Funktion `open()` gibt das `window`-Objekt des PopUps zurück. Dadurch ist es möglich, auf das Fenster und dessen Elemente (DOM) vollständig zuzugreifen. Natürlich können wir auch Events im PopUp registrieren. Mittels der Funktion `close()` ist es zudem möglich, das **Fenster wieder zu schließen**. Dies funktioniert jedoch ausschließlich bei PopUps.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislöcher</p> <p><b>Web:</b> <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> <b>E-Mail:</b> <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Zeitsteuerung](#)

## Zeitsteuerung

Immer wieder kann es vorkommen, dass wir einen bestimmten JS-Code mit einer **Verzögerung** oder auch **zyklisch** ausführen wollen. Hierfür bietet JavaScript die Funktionen `setTimeout()` und `setInterval()`, welche innerhalb dieses Themas noch genauer erklärt werden.

### Inhalt dieser Seite:

1. Timeout
2. Intervall

### Timeout

Die Funktion `setTimeout()` ist eine Funktion des `window`-Objekts und erlaubt es, einen **Code mit einer Verzögerung** (nach Ablauf eines Timeouts) auszuführen. Der Funktion wird als Parameter eine Funktion (als Ausdruck oder Referenz) übergeben. Der zweite Parameter muss ebenfalls angegeben werden und legt die Verzögerungszeit in Millisekunden fest. Im untenstehenden Beispiel wechselt das ursprünglich rot gefärbte Rechteck nach dem Ablauf von 3 Sekunden (3000 Millisekunden) die Farbe nach blau.

```
1 window.setTimeout(function ()
2 {
3     document.getElementById("rechteck").style.backgroundColor = "blue";
4 }, 3000);
```

Manchmal kann es vorkommen, dass wir ein gestartetes Timeout wieder stoppen wollen. Hierzu müssen wir uns lediglich den Rückgabewert der `setTimeout()`-Funktion merken und diesen als Wert der Funktion `clearTimeout()` übergeben. Dadurch wird das gestartete **Timeout wieder beendet** und die Funktion wird nicht aufgerufen. Im folgenden Beispiel wird zusätzlich zu dem Rechteck ein Button angezeigt. Wird auf diesen geklickt, so wird das Timeout beendet (wodurch die rote Farbe des Rechtecks erhalten bleibt) und der Button ausgeblendet. Wird innerhalb der 3 Sekunden nicht auf den Button gedrückt, so ändert sich die Farbe des Rechtecks in blau und der Button wird anschließend ebenfalls ausgeblendet.

```
1 <div id="rechteck" style="width: 200px; height: 200px; background-color: red;"></div>
2 <form>
3   <input type="button" id="button" value="Timeout abbrechen" onclick="TimeoutBeenden()" />
4 </form>
5
6 var timer = window.setTimeout(function ()
7 {
8     document.getElementById("rechteck").style.backgroundColor = "blue";
9     document.getElementById("button").style.display = "none";
10 }, 3000);
11
12 function TimeoutBeenden()
13 {
14     window.clearTimeout(timer);
15     document.getElementById("button").style.display = "none";
16 }
```

### Intervall

Die Funktion `setInterval()` erlaubt es, einen **Code zyklisch auszuführen**. Die Funktion ist von den Parametern gleich aufgebaut wie die Funktion `setTimeout()`, wovon der zweite Parameter natürlich nicht das Timeout, sondern den Intervall festlegt.

```
1 var list = new Array("blue", "lime", "pink", "yellow", "green", "orange", "black", "red");
2 var idx = 0;
3
4 window.setInterval(function ()
5 {
6     document.getElementById("rechteck").style.backgroundColor = list[idx];
7     idx = (idx + 1) % list.length;
8 }, 1000);
```

Um einen Intervall zu beenden, gibt es die Funktion `clearInterval()`. Um diese Funktion aufrufen zu können, benötigen wir den Wert, welcher von `setInterval()` zurückgegeben wurde.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [AJAX](#)

## AJAX

AJAX (*Asynchronous JavaScript and XML*) ist eine Technologie, welche es erlaubt, asynchron Daten zwischen dem Browser und einem Webserver zu übertragen. AJAX wird eingesetzt, um Inhalte zur Laufzeit der Anwendung **nachzuladen** oder Informationen zu **aktualisieren** und somit den Seiteninhalt aktiv zu verändern. Des Weiteren können mittels der AJAX-Technologie auch **Daten an den Server** mittels GET oder POST gesendet werden.

### Inhalt dieser Seite:

1. Grundlagen
2. AJAX mit XML
3. AJAX mit JSON

## Grundlagen

Eine Anfrage (engl. *request*) mittels AJAX ist im Grunde nichts anderes als eine HTTP-Anfrage. In unserem ersten Beispiel wollen wir mittels der AJAX-Anfrage lediglich den Inhalt einer Textdatei abrufen. Um eine AJAX-Anfrage in JavaScript zu senden, müssen wir zuerst ein `XMLHttpRequest`-Objekt (Abkürzung XHR) instanzieren.

Die Eigenschaft `timeout` ermöglicht die Einstellung eines **Timeouts**. Nun gibt es noch eine weitere wichtige Eigenschaft, welcher eine Funktion zugewiesen wird: `onreadystatechange`. `readystatechange` ist ein Event, welches ausgelöst wird, sobald sich der Wert der Eigenschaft `readyState` ändert. Diese Eigenschaft ändert sich wiederum bei **Änderung des Anfragestatus**. Die Eigenschaft `readyState` enthält einen numerischen Wert zwischen 0 und 4: 0 = Anfrage noch nicht ausgeführt, 1 = Verbindung zum Server hergestellt, 2 = Anfrage gesendet, 3 = Empfangte Antwort (Response), 4 = Antwort erhalten. Über die Eigenschaft `status` lässt sich des Weiteren der HTTP-Statuscode abfragen. Dieser sollte `200` sein. Der Wert `404` deutet darauf hin, dass die Datei auf dem Server nicht existiert. Um die **Antwort** (engl. *response*), welche wir vom Server empfangen haben, abzurufen, können wir die Eigenschaft `responseText` verwenden.

Nachdem wir nun alle wichtigen Eigenschaften gesetzt und unsere Ereignisfunktion definiert haben, können wir nun die Anfrage senden. Hierfür rufen wir zuerst die Funktion `open()` und anschließend die Funktion `send()` auf. Der Funktion `open()` wird die HTTP-Methode (`GET` oder `POST`), die URL (bei Verwendung von `GET` evtl. mit Parametern) und `true` (für asynchrone Abarbeitung) oder `false` (für synchrone Abarbeitung) übergeben. Synchrone AJAX-Anfragen sollten nicht verwendet werden, da dadurch die Ausführung von anderem Code blockiert wird. Der Funktion `send()` werden die POST-Daten (falls es sich um ein POST-Request handelt) übergeben. Bei Verwendung von `GET` wird der Funktion kein Parameter übergeben.

### Text-Datei:

```
1 | Hallo! Dieser Inhalt wurde mittels AJAX abgerufen.
```

### HTML-/JS-Datei:

```
1 | var req = new XMLHttpRequest();
2 | req.onreadystatechange = function ()
3 | {
4 |     if (req.readyState == 4 && req.status == 200)
5 |         alert(req.responseText);
6 | };
7 | req.timeout = 3000;
8 | req.open("GET", "ajax-test.txt", true);
9 | req.send();
```

**Wichtig:** Bei der Verwendung von `POST` muss u. U. mittels der Funktion `setRequestHeader()` ein HTTP-Header gesetzt werden. Der Funktion wird zum einen der Header-Name und zum anderen der Header-Wert übergeben. In dem genannten Szenario wäre der Header-Name `Content-Type` und der dazugehörige Wert `application/x-www-form-urlencoded`. Die Kodierung der POST-Daten erfolgt dann über das bekannte Schema `name1=wert1&name2=wert2` etc..

## AJAX mit XML

Oft wird AJAX, wie man vom Namen auch schon vermuten kann, in **Verbindung mit XML-Dateien** verwendet. Dabei muss, abgesehen vom Abruf der Daten, nichts Besonderes beachtet werden. Die XML-Elemente und Attribute werden, nicht wie beim obigen Beispiel, mittels der Eigenschaft `responseText` abgerufen, sondern mittels `responseXML`. Die Eigenschaft `responseXML` enthält das **DOM der XML-Datei**. Damit ist es nun möglich, durch die Elemente zu navigieren und diese zu selektieren. Um den eigentlichen Inhalt (Text-Inhalt) eines XML-Elements abzurufen, benötigen wir die Eigenschaften `childNodes` und `nodeValue` (siehe Beispiel).

### XML-Datei:

```
1 | <?xml version="1.0" ?>
2 |
3 | <person>
4 |   <vorname>Peter</vorname>
5 |   <nachname>Meyer</nachname>
6 | </person>
```

### HTML-/JS-Datei:

```
1 | var req = new XMLHttpRequest();
2 | req.onreadystatechange = function ()
3 | {
4 |     if (req.readyState == 4 && req.status == 200)
5 |         alert("Name: " + req.responseXML.getElementsByTagName("vorname")[0].childNodes[0].nodeValue + " " +
6 | req.responseXML.getElementsByTagName("nachname")[0].childNodes[0].nodeValue);
7 | };
8 | req.timeout = 3000;
9 | req.open("GET", "ajax-test.xml", true);
10 | req.send();
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [AJAX](#)

## AJAX mit JSON

AJAX kann auch mit dem **JSON-Dateiformat** kombiniert werden. Vorteil von JSON im Vergleich zu XML ist zum einen die teilweise kürzere Darstellung / Schreibweise und zum anderen die **einfachere Kombination mit JavaScript**. JSON wird in einem an JavaScript angelehnten Objektsyntax angegeben. JSON erlaubt Datentypen wie `String` (in einfachen oder doppelten Anführungszeichen), `Number`, `Array` (mit eckigen Klammern) und `Object` (mit geschweiften Klammern). Die Notation erfolgt dabei ähnlich wie in JavaScript. Mittels der Funktion `JSON.parse()` lässt sich dann der JSON-Code in ein JS-Objekt umwandeln. Das folgende Beispiel soll lediglich demonstrieren wie mittels AJAX JSON-Daten verarbeitet werden können.

### JSON-Datei:

```

1  {
2    "vorname" : "Peter",
3    "nachname" : "Meyer"
4  }
```

### HTML-/JS-Datei:

```

1  var req = new XMLHttpRequest();
2  req.onreadystatechange = function ()
3  {
4    if (req.readyState == 4 && req.status == 200)
5    {
6      var daten = JSON.parse(req.responseText);
7
8      alert("Name: " + daten.vorname + " " + daten.nachname);
9    }
10 };
11 req.timeout = 3000;
12 req.open("GET", "ajax-test.json", true);
13 req.send();
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Canvas](#)

## Canvas

Bei einem Canvas (engl. für *Leinwand*) handelt es sich um ein HTML-Element, mit welchem wir eine **Zeichenfläche** (also sozusagen eine Leinwand) definieren können. Die Zeichenfläche wird dabei in HTML über das `canvas`-Element angelegt. Der Zeichenvorgang erfolgt dann mittels JavaScript.

Das `canvas`-Element ist zweiteilig, wovon zwischen den Tags ein Text angegeben werden kann, welcher angezeigt werden soll, wenn das Element nicht unterstützt wird. Die **Größe** der Zeichenfläche wird in Pixel durch die Attribute `width` und `height` (Angabe ohne `px`-Suffix) angegeben. Beide Attribute sollten dabei für ein korrektes Verhalten angegeben werden. Um das `canvas`-Element anzusprechen, können wir diesem z. B. eine ID geben (siehe Beispiel).

```
1 | <canvas id="zeichnung" width="400" height="200"></canvas>
```

Um in JavaScript nun auf die Zeichenfläche zuzugreifen, müssen wir das `Node`-Objekt des Elements laden (im Beispiel mittels `getElementById()`) und die Funktion `getContext()` mit der Zeichenkette `"2d"` als Parameter aufrufen.

```
1 | var ctx = document.getElementById("zeichnung").getContext("2d");
```

**Übrigens:** Der oft verwendete Variablenname `ctx`, im Bezug auf Zeichenflächen, stellt die Abkürzung für *context* dar.

Bevor wir im nächsten Schritt damit beginnen, unsere ersten Rechtecke zu zeichnen, wollen wir zuerst noch ein paar Eigenschaften vorstellen. Der Eigenschaft `strokeStyle` kann ein Farbwert zugewiesen werden und stellt somit die **Linienfarbe** dar. Als Angabe sind, so wie in CSS auch, Farbnamen, RGB-Werte, RGBA-Werte, Hex-Werte, HSL-Werte und HSLA-Werte möglich. Die Eigenschaft `fillStyle` legt die Farbe zum Ausfüllen (**Füllfarbe**) fest. Beide Eigenschaften können während des kompletten Zeichenvorgangs mehrmals geändert werden, sodass z. B. die eine Linie rot gefärbt wird und die andere blau. Mit Hilfe der Eigenschaft `lineWidth` lässt sich die Breite der Linie (**Strichstärke**) ändern. Die Angabe erfolgt als Pixel-Wert (ohne Einheit), wovon `1` die Standardeinstellung ist. Natürlich gibt es noch viele weitere Eigenschaften, diese sind für den Anfang jedoch nicht relevant und werden nur selten benutzt.

## Rechtecke

Um Rechtecke ohne Pfade zu zeichnen (dazu im nächsten Schritt mehr), gibt es die Funktionen `strokeRect()` und `fillRect()`. Die Funktion `strokeRect()` erzeugt ein **Rechteck ohne „Füllung“** ggf. jedoch mit einem Rahmen (je nach Einstellung von `strokeStyle` und `lineWidth`). Die Funktion `fillRect()` hingegen erzeugt ein **ausgefülltes Rechteck** (jedoch ohne Rahmen). Zur Einstellung der Farbe dient, wie oben bereits beschrieben, die Eigenschaft `fillStyle`. Beiden Funktionen werden 4 Parameter übergeben, wovon der erste die X-Position und der zweite Y-Position festlegen. Beide **Positionen** beziehen sich dabei auf die Ecke oben links. Der dritte und vierte Parameter legt die **Breite und Höhe** fest. Auch hier werden alle Parameterwerte in Pixel als Zahl angegeben.

```
1 | var ctx = document.getElementById("zeichnung").getContext("2d");
2 |
3 | ctx.strokeStyle = "red";
4 | ctx.strokeRect(50, 50, 50, 50);
5 | ctx.strokeStyle = "lime";
6 | ctx.lineWidth = 10;
7 | ctx.strokeRect(200, 10, 60, 80);
8 | ctx.fillStyle = "blue";
9 | ctx.fillRect(150, 110, 200, 80);
```

**Wichtig:** Das `canvas`-Element kann als eine Art Koordinatensystem angesehen werden. Der Nullpunkt (X=0 und Y=0) befindet sich hier jedoch in der Ecke oben links. Haben wir z. B. eine Zeichenfläche mit der Größe von 300x200px, so hat die Ecke unten rechts die Position X=300 und Y=200.

## Pfade

Um Linien oder andere Formen zu zeichnen, gibt es einige weitere Funktionen. Hier spricht man nun zumeist von Pfaden. Ein Pfad (engl. *path*) kann dabei aus ein oder mehreren Linien (diese können gerade oder gekrümmt sein) bestehen. Um das Zeichnen eines Pfades zu beginnen, müssen wir zuerst die Funktion `beginPath()` aufrufen. Am Ende der Pfad-Zeichnung rufen wir die Funktion `fill()` oder `stroke()` auf. `fill()` füllt dabei den „gewählten“ Bereich mit der Füllfarbe aus, `stroke()` hingegen zeichnet lediglich die Kontur.

Zum Zeichnen von **Linien** gibt es die Funktionen `moveTo()` und `lineTo()`. Beiden Funktionen werden als erster Parameter die X-Position und als zweiter Parameter die Y-Position übergeben. Die Funktion `moveTo()` positioniert lediglich den „Cursor“, wohingegen die Funktion `lineTo()` eine Linie von der letzten Position bis zur angegebenen Position zeichnet. Eine nützliche Funktion ist `closePath()`. Mit dieser ist es möglich, wieder an den Ausgangspunkt des Pfades zu springen, um somit eine Form zu schließen.

Die Funktion `arc()` erlaubt es, **einen Kreis bzw. eine Ellipse** oder einen Kreisteil bzw. Ellipsenteil zu zeichnen. Der erste Parameter stellt die X-Position des Kreismittelpunkts dar, der zweite Parameter stellt die Y-Position des Kreismittelpunkts dar und der dritte Parameter gibt den Kreisradius an. Des Weiteren benötigen wir noch zwei Parameter, welche die Winkelpositionen angeben. Die Angabe der Winkelpositionen erfolgt mittels der Konstante `PI`, welche im `Math`-Objekt hinterlegt ist. So entspricht z. B. `0.5 * Math.PI` dem Winkel 90° und `2 * Math.PI` 360°. Der vierte Funktionsparameter gibt den Startwinkel an, dabei entspricht 0 der 3-Uhr-Position der Ellipse. Der fünfte Parameter gibt den Endwinkel an.

Mit Hilfe der Funktion `arcTo()` ist es möglich, eine **Kurve** bzw. einen Bogen (engl. *arc*) zu zeichnen. Die Zeichnung erfolgt zwischen zwei Kontrollpunkten, welche als Funktionsparameter (Reihenfolge: `x1, y1, x2, y2`) übergeben werden müssen. Der fünfte Parameter gibt den Radius des Bogens an. Hier lohnt es sich, mit den Funktionsparametern etwas zu experimentieren, bis man die Funktionsweise der Funktion besser verstanden hat.

Um ein Rechteck mittels Pfade zu zeichnen (z. B. um Transformationen darauf anzuwenden), gibt es die Funktion `rect()`. Die Funktion `rect()` hat die gleichen Parameter wie `strokeRect()` und `fillRect()`.

```
1 | var ctx = document.getElementById("zeichnung").getContext("2d");
2 |
3 | // Zick-Zack-Linien zeichnen
4 | ctx.fillStyle = "red";
5 | ctx.beginPath();
```

### Inhalt dieser Seite:

1. Rechtecke
2. Pfade
3. Texte
4. Bilder
5. Transformationen

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Canvas](#)

```

6   ctx.moveTo(0, 0);
7   ctx.lineTo(50, 200);
8   ctx.lineTo(100, 0);
9   ctx.lineTo(150, 200);
10  ctx.lineTo(200, 0);
11  ctx.lineTo(250, 200);
12  ctx.lineTo(300, 0);
13  ctx.lineTo(350, 200);
14  ctx.lineTo(400, 0);
15  ctx.fill();
16
17  // Dreieck unten mitte
18  ctx.strokeStyle = "blue";
19  ctx.beginPath();
20  ctx.moveTo(175, 200);
21  ctx.lineTo(200, 100);
22  ctx.lineTo(225, 200);
23  ctx.closePath();           // entspricht => ctx.lineTo(175, 200);
24  ctx.stroke();
25
26  // Ellipse unten links
27  ctx.fillStyle = "lime";
28  ctx.beginPath();
29  ctx.arc(100, 150, 25, 0.6 * Math.PI, 1.8 * Math.PI);
30  ctx.fill();
31
32  // kurvige Linie unten rechts
33  ctx.strokeStyle = "lime";
34  ctx.beginPath();
35  ctx.moveTo(280, 200);
36  ctx.lineTo(290, 150);
37  ctx.arcTo(300, 130, 310, 150, 10);
38  ctx.lineTo(320, 200);
39  ctx.stroke();
40
41  // Rechteck oben mitte
42  ctx.fillStyle = "yellow";
43  ctx.beginPath();
44  ctx.rect(150, 25, 100, 50);
45  ctx.fill();

```



## Texte

Zum Zeichnen eines Texts gibt es die Funktion `strokeText()` und `fillText()`. Als Parameter wird der Text sowie die X- und Y-Position übergeben. Zum Einstellen der Schriftart, Schriftgröße etc. können wir die Eigenschaft `font` nutzen. Dabei muss eine Zeichenkette gesetzt werden, welche mehrere Einstellungen umfassen kann. Hier muss vor allem auf die Reihenfolge geachtet werden: `font-style`, `font-variant`, `font-weight`, `font-size` und `font-family`. Die Eigenschaft-Namen und Werte sind dabei von [CSS](#) abzuleiten.

```

1   var ctx = document.getElementById("zeichnung").getContext("2d");
2
3   ctx.fillStyle = "red";
4   ctx.font = "30px Times New Roman";
5   ctx.fillText("Herzlich Willkommen auf", 20, 75);
6
7   ctx.strokeStyle = "blue";
8   ctx.font = "bold 35px Arial";
9   ctx.strokeText("Homepage-Webhilfe", 40, 150);

```



## Bilder

Mit Hilfe der Funktion `drawImage()` ist es möglich, ein Bild innerhalb des `canvas`-Elements zu zeichnen. Das Bild selbst muss dabei bereits auf der Seite verwendet werden (mittels `img`-Element). Wollen wir das Bild ausschließlich im `canvas`-Element anzeigen, muss das Bild zwar ebenfalls mittels des `img`-Elements eingebunden werden, jedoch können wir das Bild selbst mit der CSS-Eigenschaft `display` und dem Wert `none` (siehe Beispiel) ausblenden. Zusätzlich zu dem Bild (Angabe als `Node`-Objekt) muss der Funktion noch die X- und Y-Position übergeben werden. Bei Bedarf kann noch zusätzlich die Breite und Höhe angegeben werden.

```

1   
2   <canvas id="zeichnung" width="400" height="200"></canvas>
3
4   window.onload = function ()
5   {
6     var ctx = document.getElementById("zeichnung").getContext("2d");
7     ctx.drawImage(document.getElementById("logo"), 100, 0, 200, 200);
8   };

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Canvas](#)

**Wichtig:** In dem obigen Beispiel ist es nochmals besonders wichtig, dass die Canvas-Funktion `drawImage()` erst aufgerufen wird, wenn die Seite (und somit auch das Bild) vollständig geladen ist.

### Transformationen

Transformationen erlauben es, zu zeichnende Bestandteile, wie z. B. Rechtecke, Linien, Kreise, Texte oder Bilder, zu rotieren, verschieben, skalieren und neigen. Dafür stehen uns die Funktionen `rotate()`, `translate()`, `scale()`, `transform()` und `setTransform()` zur Verfügung, welche wir Ihnen gleich noch genauer erklären werden.

Davor wollen wir uns jedoch mit der Funktionsweise aller Transformationsfunktionen sowie den Funktionen `save()` und `restore()` beschäftigen: Alle **ausgeführten Transformationen werden gespeichert** (und dies nicht nur für den aktuellen Pfad). Dies hat zur Folge, dass wenn wir z. B. zwei Rechtecke zeichnen und beide mittels der `rotate()`-Funktion und einer Angabe von 1° rotieren, dass das erste Rechteck wie gewollt um 1° rotiert wird, das zweite jedoch schon um 2°. Dies kann u. U. gewünscht sein oder eben auch nicht. Um diesem Szenario entgegenzuwirken, gibt es die Funktionen `save()` und `restore()`. Beim Aufruf der Funktion `save()` werden **Transformations-Einstellungen** sowie viele andere Parameter (u. a. `strokeStyle`, `fillStyle`, `lineWidth` und `font`) **gespeichert**. Um diese **Einstellungen wieder zurückzuholen**, gibt es die Funktion `restore()`. Innerhalb eines Zeichenvorgangs kann natürlich mehrmals die Funktion `save()` und `restore()` aufgerufen werden. `save()` baut dabei einen **Einstellungsstapel** von unten nach oben auf, welcher von `restore()` von oben nach unten wieder abgebaut wird. Für alle Transformationsfunktionen gilt zusätzlich noch, dass die Transformationen nur für Bestandteile gelten, welche nach dem Aufruf der Transformationsfunktion gezeichnet wurden.

Die Funktion `rotate()` kann zum **Rotieren** verwendet werden. Der Funktion muss dabei ein Wert in der Einheit Radiant übergeben werden. Um Grad in Radiant umzuwandeln, gilt die Formel `grad * (Math.PI / 180)`. Die Rotation selbst erfolgt immer vom Nullpunkt der Zeichenfläche aus.

Die Funktion `translate()` **verschiebt den Nullpunkt** der Zeichenfläche, wovon zu beachten ist, dass Überstände nicht gezeichnet werden. Als Parameter werden der `translate()`-Funktion die neue X- und Y-Position übergeben.

Mit Hilfe der Funktion `scale()` lässt sich eine **Zeichnung skalieren**. Hierfür werden der Funktion zwei Parameter übergeben, welche den Skalierungsfaktor (z. B. 1 = 100%, 1.5 = 150%) für die X- und Y-Achse festlegen. Bei Verwendung der `scale()`-Funktion wird neben der Größe (Breite und Höhe) auch die Positionsangabe (X- und Y-Position) skaliert, wovon sich ein Teil des Ergebnisses ähnlich wie eine Verschiebung auswirkt.

Im folgenden Beispiel wird bewusst die Transformation innerhalb der Schleife nicht zurückgesetzt. Das Beispiel zeigt ein blaues Rechteck, welches immer mehr im Uhrzeigersinn rotiert wird, ein rotes Rechteck, welches immer weiter verschoben wird, und ein grünes Rechteck, welches immer größer wird (und somit auch immer weiter verschoben wird).

```

1  var ctx = document.getElementById("zeichnung").getContext("2d");
2
3  ctx.save();
4
5  ctx.fillStyle = "blue";
6  for (var i = 0; i < 30; i++)
7  {
8      ctx.beginPath();
9      ctx.rotate(1 * (Math.PI / 180));
10     ctx.rect(100, 50, 100, 50);
11     ctx.fill();
12 }
13
14 ctx.restore();
15 ctx.save();
16
17 ctx.fillStyle = "red";
18 for (var i = 0; i < 30; i++)
19 {
20     ctx.beginPath();
21     ctx.translate(1, 1);
22     ctx.rect(225, 25, 100, 50);
23     ctx.fill();
24 }
25
26 ctx.restore();
27 ctx.save();
28
29 ctx.fillStyle = "lime";
30 for (var i = 0; i < 5; i++)
31 {
32     ctx.beginPath();
33     ctx.scale(1.1, 1.1);
34     ctx.rect(195, 95, 25, 25);
35     ctx.fill();
36 }
    
```

Neben den zwei bisher vorgestellten Funktionen zur Transformation gibt es noch die Funktionen `transform()` und `setTransform()`. Beiden Funktionen werden 6 Parameter übergeben, mit welchen eine **Skalierung, Neigung und Verschiebung** angegeben werden kann. Dabei muss vor allem auf die Reihenfolge geachtet werden: horizontale Skalierung, horizontale Neigung, vertikale Neigung, vertikale Skalierung, horizontale Verschiebung und vertikale Verschiebung. `transform()` verhält sich wie die bereits vorgestellten Transformationsfunktionen, d. h. die Transformationseinstellungen werden „gespeichert“ und summieren sich u. U. bei mehrmaligem aufrufen. `setTransform()` hingegen setzt zuerst die Transformationseinstellungen zurück und führt dann die Transformation aus. Im Beispiel ist dieser Unterschied klar zu erkennen.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Canvas](#)

```

1  var ctx = document.getElementById("zeichnung").getContext("2d");
2
3  ctx.save();
4
5  ctx.fillStyle = "blue";
6
7  for (var i = 0; i < 15; i++)
8  {
9      ctx.beginPath();
10     ctx.transform(1, 0, 0, 1, i, i);
11     ctx.rect(25, 25, 50, 50);
12     ctx.fill();
13 }
14
15 for (var i = 0; i < 15; i++)
16 {
17     ctx.beginPath();
18     ctx.setTransform(1, 0, 0, 1, i, i);
19     ctx.rect(150, 25, 50, 50);
20     ctx.fill();
21 }
22
23 ctx.restore();
24
25 ctx.fillStyle = "red";
26
27 for (var i = 0; i < 15; i++)
28 {
29     ctx.beginPath();
30     ctx.transform(1, 0, (i / 100), 1, 0, 0);
31     ctx.rect(250, 25, 50, 50);
32     ctx.fill();
33 }
34
35 for (var i = 0; i < 15; i++)
36 {
37     ctx.beginPath();
38     ctx.setTransform(1, 0, (i / 100), 1, 0, 0);
39     ctx.rect(250, 125, 50, 50);
40     ctx.fill();
41 }

```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung





Sie befinden sich hier: [Homepage-Webhilfe](#) » [JavaScript](#) » [Abschluss](#)

## Abschluss

Nun ist das JavaScript-Tutorial auch schon wieder am Ende angelangt. In diesem Kapitel haben Sie alle Grundlagen sowie fortgeschrittenere Programmiertechnologien (z. B. Objektorientierung, AJAX und Canvas) kennengelernt. Nachdem Sie nun vermutlich bereits HTML und CSS gelernt haben, haben Sie den **ersten und wichtigsten Schritt für Webseiten in Bezug auf die Programmierung erreicht**.

Auf Grund der **Komplexität des DOMs** ist es immer wieder nützlich, ein Nachschlagewerk zu haben. Auch hier empfehlen wir wieder die [Referenz von W3Schools](#). Für die gängigsten Eigenschaften und Funktionen können Sie natürlich auch unsere Karteikarten sowie dieses Tutorial nutzen.

Mit dem aktuellen Wissenstand ist es uns nun möglich, statische Seiten zu erstellen und diese mittels JavaScript zur Laufzeit aktiv zu verändern. Unter Umständen haben Sie jetzt noch Interesse an weiteren clientseitigen Programmiersprachen wie [ActionScript \(Flash\)](#) oder [Java \(Applet\)](#). Andernfalls wäre es nun bei Bedarf an der Zeit, unsere **Seite dynamisch erstellen zu lassen**. Hierfür dienen die serverseitigen Skript- und Programmiersprachen [PHP](#), [Perl](#), [ASP.NET](#) oder [Java \(EE\)](#). Zu den genannten Sprachen bieten wir hier ebenfalls Tutorials an und würden uns sehr über einen Besuch freuen.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

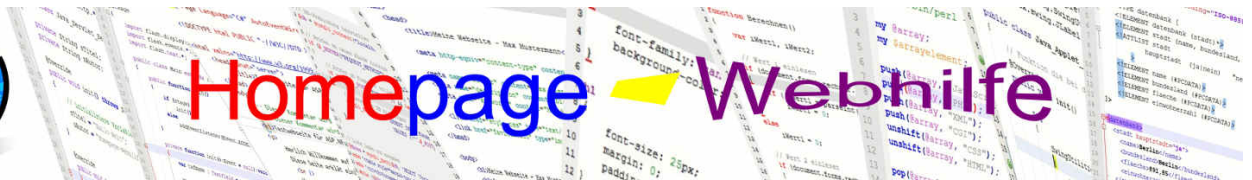
Java EE

XML

# E-Book

## ActionScript





Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Einführung](#)

## Einführung



ActionScript (kurz AS) ist eine Programmiersprache von Adobe, welche dazu verwendet wird, **Flash-Anwendungen** zu entwickeln. Bei Flash-Anwendungen kann es sich um eine Desktop-Anwendung, eine „Rich Internet Application“ (kurz RIA) oder eine App (für Android, Apple iOS oder BlackBerry OS) handeln.

ActionScript **basiert auf ECMAScript** und ist somit **JavaScript-ähnlich**, weshalb Vorkenntnisse in JavaScript für diesen Kurs vorteilhaft sind. Die Sprache ist **objektorientiert** und verfügt über eine starke und statische Typisierung. Somit sind deklarierte Variablen an einen bestimmten Typ gebunden, was in JavaScript nicht der Fall ist.

ActionScript-Applikationen müssen **kompiliert** werden und werden an den Browser im Binärformat geschickt (Dateiendung swf). Die AS-Anwendung kann mittels einem Link in HTML (siehe [unten](#)) eingebunden werden. Die Ausführung der Flash-Anwendung erfolgt nicht durch den Webbrowser, sondern durch den **Adobe Flash Player**.

Die Programmierung von ActionScript kann in jedem Editor erfolgen, jedoch sollte vorzugsweise eine integrierte **Entwicklungsumgebung** (kurz IDE für Integrated Development Environment) verwendet werden. Hier gibt es zum einen die kostenpflichtigen IDEs Adobe Animate (früher Adobe Flash Professional) und Adobe Flash Builder sowie die kostenlose IDE FlashDevelop. Für den Kompilierungsvorgang benötigen wir noch zusätzlich ein „Software Development Kit“ (kurz SDK). Hier gibt es zum einen Adobe Animate, welcher nur in der gleichnamigen kostenpflichtigen IDE verfügbar ist, und zum anderen **Apache Flex** (früher Adobe Flex). Das Apache Flex SDK ist kostenlos verfügbar.

Der Schwerpunkt in diesem Tutorial liegt auf den Grundlagen der Sprache sowie auf der Entwicklung von Webanwendungen. Alle Beispiele wurden mit der IDE FlashDevelop (Beispiel-Projekte können mit dem Download-Button heruntergeladen werden) und dem SDK Apache Flex erstellt.

## Geschichte

ActionScript wurde ursprünglich von Macromedia entworfen. 1999 erschien die 1. Version der Sprache. Im Jahre 2004 wurde ActionScript 2 zusammen mit dem Flash Player 7 herausgegeben, welche nun eine **klassenlose Objektorientierung** erlaubte.

Macromedia wurde im Dezember 2005 von Adobe Systems übernommen. Dabei wurden die bekannten Produkte wie Flash und Dreamweaver weitergeführt. 2006 wurde ActionScript 3 (kurz AS3) vorgestellt, welche ab dem Flash Player 9 unterstützt wird.

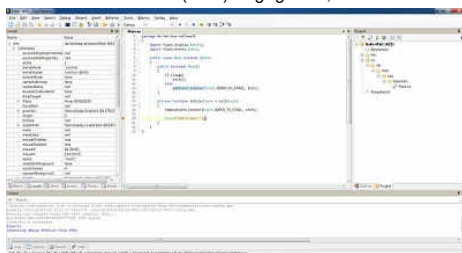
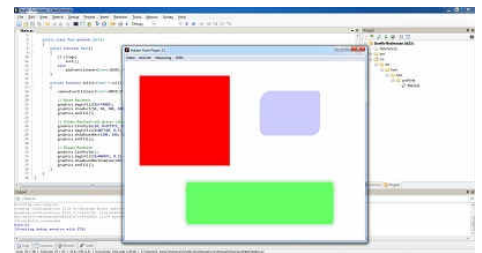
ActionScript 3 ist eine große, jedoch auf Grund der vielen Änderungen, nicht kompatible Version zu den Vorgängern ActionScript 1 und ActionScript 2. AS3 basiert auf Entwürfen der eingestellten **ECMAScript Version 4** und ist somit ähnlich wie JavaScript. Anders wie bei JavaScript sind AS3-Anwendungen **klassenbasiert** (Objektorientierung) und verfügen über eine **starke und statische Typisierung** (kurz: Datentypen müssen im Voraus festgelegt sein).

Dieses Tutorial beschäftigt sich lediglich mit der ActionScript Version 3.

## SDK und IDE

Wie bereits oben schon erwähnt, müssen AS-Anwendungen kompiliert werden. Vormalig war Adobe Animate (früher Adobe Flash Professional) die einzige IDE (mit integriertem SDK) zur Entwicklung von ActionScript-Anwendungen. Doch in der Zwischenzeit gibt es ein paar weitere Entwicklungsumgebungen. Dazu zählen z. B. Adobe Flash Builder und die kostenlose IDE (*Integrated Development Environment*) FlashDevelop. **FlashDevelop** bietet dabei eine gute Syntax-Hervorhebung und Code-Vervollständigung.

Zusätzlich zur Entwicklungsumgebung ist noch ein sogenanntes SDK (*Software Development Kit*) notwendig. Das SDK enthält die notwendigen Libraries, einen Compiler und weitere Hilfsprogramme (sogenannte Dienstprogramme). Das bekannteste SDK (abgesehen von Adobe Animate) ist **Apache Flex**, welches kostenlos erhältlich ist und ideal in Kombination mit FlashDevelop verwendet werden kann. Vormalig wurde dieses SDK ebenfalls von Adobe entwickelt. 2011 hat Adobe das Flex SDK an die Apache Software Foundation (ASF) abgegeben, von welcher das SDK noch heute aktiv weiterentwickelt wird.



Zusammen mit der FlashDevelop IDE ist Apache Flex eine der besten **kostenlosen Alternativen** zu Adobe Animate. FlashDevelop kann mittels einem speziellen Flash Player um eine **Debugging-Funktion** erweitert werden. Dadurch können dann auch Haltepunkte (engl. *breakpoints*) gesetzt werden, der Programmablauf schrittweise durchgegangen werden sowie Variablen überwacht werden.

Die folgende Aufzählung zeigt den Ablauf der Installation und Konfiguration von FlashDevelop sowie dem Apache Flex SDK:

1. Download der IDE von <http://www.flashdevelop.org/>
2. Installation der IDE
3. Download der SDK (SDK Installer) von <http://flex.apache.org/>
4. Installation der SDK
5. Download des „Standalone Debug Players“ (projector content debugger) von [http://www.adobe.com/support/flashplayer/debug\\_downloads.html](http://www.adobe.com/support/flashplayer/debug_downloads.html)
6. Kopieren des heruntergeladenen Flash Players in den Ordner `runtimes/player` des Verzeichnisses der SDK (Ordner muss ggf. erstellt werden)
7. Starten der IDE
8. Eintragen des SDK-Installationspfads in der IDE unter `Tools` → `Program Settings` → `AS3Context` → `Installed Flex SDKs`
9. Eintragen des „Standalone Debug Players“ in der IDE unter `Tools` → `Program Settings` → `FlashViewer` → `External Player Path`

**Übrigens:** Zur Erstellung der in diesem Kurs enthaltenen Beispiele bzw. Projekte wurde ausschließlich die IDE FlashDevelop 5.1.1 in Kombination mit dem Apache Flex SDK 4.15 (Adobe Flash Player 20 oder höher) verwendet. Bei der Verwendung einer neueren SDK in Kombination mit unseren Beispielprojekten muss ggf. die Plattform-Version unter den Projekteinstellungen geändert werden.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



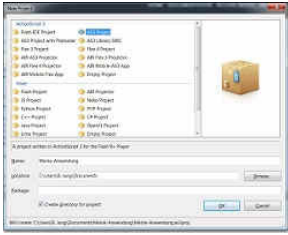
Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Einführung](#)

## Erstes Programm



Nachdem wir die IDE und das SDK eingerichtet haben, können wir endlich unser erstes Programm erstellen. Dazu klicken wir bei FlashDevelop im Menü auf **Project** und anschließend auf **New Project**. Es erscheint ein weiteres Fenster, in welchem wir nun als erstes den Typ **AS3 Project** auswählen. Nun müssen wir noch einen Namen und einen Speicherort festlegen. Die Angabe des „Package“ ist vorerst unwichtig und nicht unbedingt notwendig. Wir werden uns im [Thema Objektorientierung](#) jedoch nochmals genauer damit beschäftigen.

Nach dem Klick auf OK wird das **Projekt erstellt**. Auf der rechten Seite finden Sie die Projektansicht. Dort sollte sich auch der Ordner **src** befinden, in welchem sich Ihre Quellcode-Dateien befinden. Dies ist bei einem neu erstellten Projekt dieses Typs lediglich die Datei **Main.as**. Um diese zu öffnen, müssen wir lediglich einen Doppelklick ausführen. Anschließend wird der Inhalt der Datei angezeigt.

Dieses Projekt könnte nun bereits ausgeführt werden. Dazu muss lediglich auf den blauen Pfeil (**Test Project**) geklickt werden. Das Programm wird nun kompiliert und anschließend vom ausgewählten Flash Player geöffnet, jedoch sehen wir bisher noch nicht viel (abgesehen von einem leeren Fenster). Um dies zu ändern, fügen wir innerhalb der **init()**-Funktion (an der Stelle, wo sich der Kommentar *// entry point* befindet) den Code `trace("Hallo Welt!");` ein. Dabei handelt es sich um einen Funktionsaufruf der Funktion **trace()**, welche eine Ausgabe im Ausgabefenster der Entwicklungsumgebung erzeugt. Anschließend starten wir das Programm erneut und wir sollten die Ausgabe „Hallo Welt!“ im Ausgabefenster (**Output**) unten sehen. Der Quellcode der Datei **Main.as** sollte bisher ungefähr wie folgt aussehen:

```

1 package de.hwh.bsp.hallowelt
2 {
3     import flash.display.Sprite;
4     import flash.events.Event;
5
6     public class Main extends Sprite
7     {
8         public function Main()
9         {
10            if (stage)
11                init();
12            else
13                addEventListener(Event.ADDED_TO_STAGE, init);
14        }
15
16        private function init(e:Event = null):void
17        {
18            removeEventListener(Event.ADDED_TO_STAGE, init);
19
20            trace("Hallo Welt!");
21        }
22    }
23 }

```

**Ausgabe:**

1 | Hallo Welt!



## Syntax

Der Syntax von ActionScript ist auf Grund der Basierung auf ECMAScript ähnlich wie in JavaScript, jedoch gibt es auch ein paar Unterschiede. Der Vollständigkeit halber wollen wir hier jedoch den kompletten Syntax von ActionScript vorstellen.

Bei dem Syntax einer Sprache handelt es sich um einen **Satz von Regeln** zur Notation (also zum Schreiben) von einem Code einer Sprache. So gibt es in AS u. a. **Anweisungen** (engl. *statements*), welche am Ende mit einem Semikolon `;` abgeschlossen werden müssen. Bei Anweisungen kann es sich z. B. um Deklarationen, Wertzuweisungen oder Funktionsaufrufe handeln. Um einen bestimmten Code zu gruppieren, gibt es in ActionScript sogenannte **Blöcke**. Der Block beginnt mit einer öffnenden geschweiften Klammer `{` und endet mit einer schließenden geschweiften Klammer `}`. Der zum Block gehörende Code muss dabei innerhalb der Klammern angegeben werden. Um auf Konstanten, Eigenschaften oder Funktionen einer Klasse oder eines Objekts und auf Packages zuzugreifen, wird der Punktoperator `.` verwendet. Runde Klammern kommen bei der Deklaration sowie beim Aufruf von **Funktionen** zum Einsatz. Innerhalb des runden Klammerspaars können die Übergabewerte (auch als **Parameter** bezeichnet) kommagetrennt angegeben werden. Des Weiteren können runde Klammern bei Berechnungen verwendet werden. Bei Abfragen oder Schleifen wird ebenfalls ein rundes Klammerspaar angegeben, in welchem die **Bedingung** notiert wird. Zum Schluss muss noch erwähnt werden, dass ActionScript zwischen **Groß- und Kleinschreibung** unterscheidet, weshalb Sie immer auf eine korrekte Schreibweise achten sollten. In AS werden Variablenamen, Funktionsnamen und Klassennamen standardmäßig in der **Camel-Case-Schreibweise** notiert. Dabei wird der Anfangsbuchstabe von zusammengesetzten Wörtern immer groß geschrieben. Bei Variablenamen und Funktionsnamen wird zudem der erste Buchstabe zumeist klein geschrieben. Diese Regelung ist jedoch lediglich ein Vorschlag und trägt nicht zur Funktionalität des Programms bei. Eine detaillierte Erklärung zu den genannten Regeln sowie dessen Praxisbezug werden Sie innerhalb dieses Kurses finden.

Nun wollen wir noch auf die Bestandteile des obigen Beispiels eingehen: Jede Klasse befindet sich in einem sogenannten Package. Ein **Package** erlaubt es, Klassen zu strukturieren und zu gruppieren. Packages bilden immer einen Block und beginnen mit dem Schlüsselwort `package` gefolgt von dem Namen des Packages (im unteren Beispiel ist dies `de.hwh.bsp.hallowelt`). Haben Sie beim Erstellen Ihres Projekts kein Package angegeben, so befinden sich die Klassen in einem namenlosen Package. Eine Strukturierung und Gruppierung ist nur bei Verwendung von Namen möglich.

```

1 package de.hwh.bsp.hallowelt
2 {
3
4 }

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Einführung](#)

Die `import`-Anweisungen dienen dazu, **Packages und Klassen einzubinden**. Die hier eingebundenen Klassen `flash.display.Sprite` und `flash.events.Event` gehören zu den Libraries der SDK.

```
1 | import flash.display.Sprite;
2 | import flash.events.Event;
```

Durch das Schlüsselwort `class` wird eine **Klasse** deklariert. Eine Klasse stellt eine Art „Bauplan“ dar. Objekte können in ActionScript nur durch die Vorlage einer Klasse erzeugt werden. Der dahinter folgende Name `Main` ist der Klassenname. Das Schlüsselwort `extends` kennzeichnet, dass die Klasse die sogenannte Kindklasse einer anderen Klasse (der sogenannten Basisklasse) ist. Das Schlüsselwort `public` ist ein sogenannter Zugriffsmodifizierer. Eine Klassendeklaration bildet, wie das Package auch, immer einen eigenständigen Block, welcher sich im Package-Block befindet.

```
1 | public class Main extends Sprite
2 | {
3 |
4 | }
```

Der folgende Code zeigt die sogenannte **Konstruktorfunktion**. Die Konstruktorfunktion hat immer den gleichen Namen wie die Klasse. Alle Funktionen bilden einen eigenen Block, in welchem der zur Funktion gehörende Code angegeben wird. An dem Code der Konstruktorfunktion werden wir vorerst nichts ändern, weshalb wir diesem auch erstmals keine weitere Beachtung schenken. Funktionen sind immer durch das Schlüsselwort `function` zu erkennen.

```
1 | public function Main()
2 | {
3 |     if (stage)
4 |         init();
5 |     else
6 |         addEventListener(Event.ADDED_TO_STAGE, init);
7 | }
```

Die **Initialisierungsfunktion** ist eine weitere Funktion, welche beim Erstellen eines neuen Projekts automatisch erstellt wird. Auch hier ist wieder das Schlüsselwort `function` zu finden. Innerhalb der Funktion befindet sich neben dem Aufruf der Funktion `removeEventListener()` der Kommentar `// entry point`. Hier können wir also nun unseren Code platzieren, welcher beim „Start“ ausgeführt wird. Der Programmcode aller unserer Beispiele (sofern nicht anders angegeben) befindet sich innerhalb dieser Funktion. Für unser erstes Programm haben wir den Kommentar entfernt und durch den Code `trace("Hallo Welt!");` ersetzt. Hier wird die Funktion `trace()` aufgerufen, mit welcher wir eine Ausgabe im Ausgabefenster erzeugen können.

```
1 | private function init(e:Event = null):void
2 | {
3 |     removeEventListener(Event.ADDED_TO_STAGE, init);
4 |
5 |     trace("Hallo Welt!");
6 | }
```

**Kommentare** können in AS auf zwei verschiedene Arten erstellt werden. Generell dienen Kommentare zur **Dokumentation** innerhalb des Quellcodes. Oft werden Kommentare aber auch einfach zum „Auskommentieren“ von Quellcode verwendet, um so z. B. fehlerhaften Code zu finden oder einen Programmteil „auszuhängen“. Einen einzeiligen Kommentar können wir mit den Zeichen `//` erstellen. Einzeilige Kommentare gelten ab der angegebenen Stelle bis zum Ende der Zeile. Mehrzeilige Kommentare werden mit den Zeichen `/*` und `*/` erstellt. Der auszukommentierende Bereich muss sich dabei zwischen den zwei Zeichensequenzen befinden.

```
1 | // Einzeiliger Kommentar
2 |
3 | /* Mehrzeiliger
4 |    Kommentar */
```

## HTML-Einbindung

Um Flash-Anwendungen in HTML einzubinden, gibt es das leere Element `embed`. Im `embed`-Element geben wir nun die Attribute `type`, `src`, `width` und `height` an. Als Wert des `type`-Attributs geben wir den MIME-Typ von Flash-Anwendungen an, bei welchem es sich um `application/x-shockwave-flash` handelt. Das `src`-Attribut gibt die URL zu der `swf`-Datei an. Die Attribute `width` und `height` legen die Größe fest. Als Werte sind ausschließlich Pixel-Werte (ohne Einheit) erlaubt.

```
1 | <embed type="application/x-shockwave-flash" src="Animation.swf" width="800" height="600" />
```

**Wichtig:** In den ersten Anwendungen arbeiten wir ausschließlich mit der `trace()`-Funktion zur Ausgabe. Das Zeichnen von Inhalten und verwenden von Steuerelementen (wodurch Inhalte in der Anwendung „sichtbar“ werden) werden wir erst später behandeln. Den Code zur Einbindung in HTML benötigen wir also erst später.

**Wichtig:** Dieses Thema war zwar für den Anfang sehr theorielastig und mit vielen Fachbegriffen gefüllt, jedoch werden wir die meisten der hier angeschnitten Themen wie Packages, Klassen, Objekte und Funktionen später nochmals aufgreifen und genauer erklären.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Variablen und Datentypen](#)

## Variablen und Datentypen

### Inhalt dieser Seite:

1. Zeichenkette
2. Nummer
3. Wahrheitswert
4. Array
5. Vektor
6. Datum

Variablen werden in AS über das Schlüsselwort `var` deklariert. Hinter dem Schlüsselwort muss ein **Name** angegeben werden, welcher innerhalb des aktuellen Blocks und der darin verschachtelten folgenden Blöcke **eindeutig** sein muss. Hinter dem Namen muss nun noch ein Doppelpunkt `:` sowie der Datentyp angegeben werden. Da es sich bei dieser sogenannten **Deklaration** (also dem Reservieren bzw. Anlegen der Variable) um eine Anweisung handelt, muss am Ende noch ein Semikolon `;` angegeben werden.

```
1 | var NAME:TYP;
```

Um einer Variable einen Wert zuzuweisen (**Wertzuweisung**), notieren wir den Namen der Variable gefolgt von einem Gleichheitszeichen `=` (Zuweisungsoperator). Anschließend wird der zuzuweisende Wert angegeben. Dabei kann es sich um eine Konstante, den Wert einer anderen Variablen oder um den Rückgabewert einer Funktion handeln. Auch hier darf das Semikolon `;` nicht vergessen werden. Der Wert von Variablen kann mehrmals zugewiesen werden, jedoch sollten Sie Variablen immer nur zu einem bestimmten Zweck verwenden, welchen Sie im Variablennamen zum Ausdruck bringen sollten.

```
1 | NAME = WERT;
```

Das erste Zuweisen einer Variablen wird als **Initialisierung** bezeichnet. Die Variableninitialisierung kann in einer Wertzuweisungs-Anweisung oder direkt bei der Deklaration (siehe folgendes Beispiel) vorgenommen werden.

```
1 | var NAME:TYP = WERT;
```

## Zeichenkette

Bei einer Zeichenkette (engl. *String*) handelt es sich um die Aneinanderreihung von Zeichen. Eine Zeichenkette kann dabei kein, ein oder mehrere Zeichen enthalten. Zeichenketten zeichnen sich durch den Typ `String` aus. Werte von Zeichenketten müssen in **Anführungszeichen** angegeben werden (doppelte oder einfache Anführungszeichen). Um eine Zeichenkette mit einer anderen zu verbinden, können wir das Pluszeichen `+` verwenden.

```
1 | var meinName:String = "Peter";
```

Zum Vereinfachen der Arbeit mit Zeichenketten gibt es einige sogenannte String-Funktionen. Um diese **String-Funktionen** zu nutzen, benötigen wir den im vorherigen Thema angesprochenen Punktoperator. Dabei wird der Variablenname oder der Wert von einem Punkt gefolgt angegeben. Anschließend wird die Eigenschaft oder die Funktion notiert. Die Eigenschaft `length` enthält einen lesbaren Wert, welcher die **Länge der Zeichenkette** enthält. Die Funktionen `substring()` und `substr()` erlauben das **Extrahieren einer Teilzeichenkette** aus einer Zeichenkette. Beiden Funktionen muss mindestens ein Parameter mitgegeben werden, bei welchem es sich um den Startindex handelt. Dieser stellt die Nummer des Zeichens dar (erstes Zeichen entspricht dem Index 0, zweites Zeichen dem Index 1 usw.). Der Funktion `substring()` kann optional ein zweiter Parameter übergeben werden, wobei es sich um den Endindex handelt. `substr()` verfügt ebenfalls über einen zweiten optionalen Parameter, welcher die Länge der Teilzeichenkette angibt. Die Funktion `indexOf()` erlaubt die **Suche innerhalb einer Zeichenkette**. Als Parameter wird der Funktion eine Zeichenkette übergeben, welche gesucht werden soll. Als Rückgabe gibt die Funktion den Index des ersten Vorkommens der angegebenen Zeichenkette zurück. Bleibt die Suche erfolglos, so gibt die Funktion `-1` zurück. Mit der Funktion `lastIndexOf()` kann eine Suche vom Ende zum Anfang durchgeführt werden, um somit das letzte Vorkommen zu finden.

```
1 | var ausgabeMeldung:String;
2 | var vorname:String = "Peter";
3 | var nachname:String = "Meyer";
4 |
5 | ausgabeMeldung = "Hallo, mein Name ist " + vorname + " " + nachname + "!";
6 |
7 | trace(ausgabeMeldung);
8 | trace("Länge: " + ausgabeMeldung.length);
9 | trace("Teil 7-32: " + ausgabeMeldung.substring(7, 32));
10 | trace("Teil 0-L5: " + ausgabeMeldung.substr(0, 5));
11 | trace("1. Komma: " + ausgabeMeldung.indexOf(", "));
12 | trace("letztes Leerzeichen: " + ausgabeMeldung.lastIndexOf(" "));
```

### Ausgabe:

```
1 | Hallo, mein Name ist Peter Meyer!
2 | Länge: 33
3 | Teil 7-32: mein Name ist Peter Meyer
4 | Teil 0-L5: Hallo
5 | 1. Komma: 5
6 | letztes Leerzeichen: 26
```



## Nummer

Für numerische Werte gibt es die drei Datentypen: `Number`, `int` und `uint`. Die Datentypen `int` und `uint` werden für **32-bit große Ganzzahlen** verwendet. Variablen des Typs `uint` sind **vorzeichenlos** und können dadurch Werte zwischen 0 und 4.294.967.295 besitzen. Werte vom Typ `int` hingegen von -2.147.483.648 bis +2.147.483.647. Der Datentyp `Number` kann neben Ganzzahlen auch **Gleitkommazahlen mit doppelter Genauigkeit** (64-bit nach IEEE-754) enthalten. Es ist jedoch zu empfehlen, den Datentyp `Number` nur für Gleitkommazahlen sowie Ganzzahlen größer 32bit zu verwenden. Zu beachten ist, dass Gleitkommazahlen mit einem Punkt dargestellt werden (für 2,4 also `2.4`).

```
1 | var innenTemperatur:Number = 22.3;
```

Um die einfachen mathematischen **Grundrechenarten** zu nutzen, können die dafür standardisierten Zeichen `+` (Addition), `-` (Subtraktion), `*` (Multiplikation), `/` (Division) und `%` (Modulo) genutzt werden.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Variablen und Datentypen](#)

```
1 | a = (b + c) * d;
```

**Wichtig:** Bei der „Addition“ einer Zahl mit einer Zeichenkette wird die Zahl automatisch in eine Zeichenkette umgewandelt (man spricht hier von der sogenannten Typkonvertierung). So wird z. B. aus "CO" + 2 "CO2" und aus "1" + 2 "12".

Für die meisten Operatoren sind zusätzlich einige Kurzoperatoren verfügbar, welche in der folgenden Tabelle mit Beispielen aufgelistet sind.

Name	Formel	Formel (kurz)	Formel (schrittweise)
Addition	$x = y + z$	$x += y$ $\Delta x = x + y$	$x++$ $\Delta x += 1$ $\Delta x = x + 1$
Subtraktion	$x = y - z$	$x -= y$ $\Delta x = x - y$	$x--$ $\Delta x -= 1$ $\Delta x = x - 1$
Multiplikation	$x = y * z$	$x *= y$ $\Delta x = x * y$	-
Division	$x = y / z$	$x /= y$ $\Delta x = x / y$	-
Modulo	$x = y \% z$	$x \% = y$ $\Delta x = x \% y$	-

**Übrigens:** Das Erhöhen einer Variablen mittels dem ++-Operator wird als **Inkrementierung** bezeichnet. Beim --Operator wird von der **Dekrementierung** gesprochen.

**Wichtig:** An Stelle von x++ und x-- können wir auch ++x und --x notieren. Der Unterschied besteht darin, dass bei x++ die Inkrementierung nach der Zuweisung stattfindet. Bei ++x wird zuerst inkrementiert und dann der Wert zugewiesen. Das Gleiche gilt natürlich auch für x-- und --x. Die Verwendung von ++x oder ++x bzw. x-- oder --x spielt also nur dann eine Rolle, wenn diese Anweisung einer anderen Variablen zugewiesen wird.

Auch für Zahlen sind ein paar Hilfsfunktionen verfügbar. Mit der Funktion toFixed() können wir die anzuzeigenden **Nachkommastellen für Gleitkommazahlen begrenzen**. Zur **Konvertierung** einer Zeichenkette in eine Zahl können wir die Funktionen parseInt() und parseFloat() nutzen. Schlägt die Konvertierung von parseInt() und parseFloat() fehl, so geben die Funktionen den Wert NaN zurück. Um zu überprüfen, ob eine Variable den Wert NaN hat, können wir die Funktion isNaN() verwenden. Einige weitere Hilfsfunktionen befinden sich in der Math-Klasse. Um auf diese zuzugreifen, notieren wir den Klassennamen Math gefolgt von einem Punkt . und dem Funktionsnamen. Zum **Runden** von Zahlen stehen die Funktionen ceil(), floor() und round() zur Verfügung. floor() rundet auf die nächstkleinere Ganzzahl, ceil() auf die nächstgrößere und round() nach dem kaufmännischem Prinzip (2,4 zu 2 und 2,6 zu 3). Den Funktionen min() und max() kann eine Liste von Zahlen übergeben werden, aus welchem das **Minimum oder Maximum** ermittelt werden soll.

```
1 | var ergebnis:int;
2 | var temperatur1:Number = 25.64;
3 | var temperatur2:Number = 21.91;
4 | var temperaturDifferenz:Number;
5 |
6 | ergebnis = parseInt("23") + parseInt("5");
7 | temperaturDifferenz = temperatur1 - temperatur2;
8 |
9 | trace("23 + 5 = " + ergebnis);
10 | trace("Temperatur-Differenz: " + temperaturDifferenz.toFixed(1) + " - " + Math.floor(temperaturDifferenz) + " - " +
    Math.ceil(temperaturDifferenz) + " - " + Math.round(temperaturDifferenz));
11 | trace("Minimum: " + Math.min(12, 43, 23, 8, 37, 49, 15, 6, 31, 28));
12 | trace("Maximum: " + Math.max(12, 43, 23, 8, 37, 49, 15, 6, 31, 28));
```

**Ausgabe:**

```
1 | 23 + 5 = 28
2 | Temperatur-Differenz: 3.7 - 3 - 4 - 4
3 | Minimum: 6
4 | Maximum: 49
```



## Wahrheitswert

Ein Wahrheitswert kennzeichnet sich durch den Typ Boolean und kennt lediglich zwei Zustände: wahr (true) und unwahr (false).

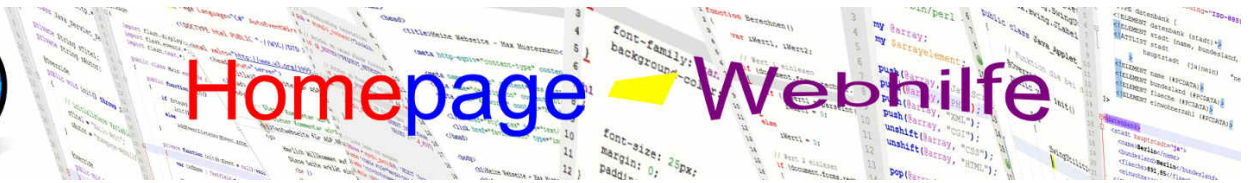
```
1 | var benutzerAngemeldet:Boolean = false;
```

Neben der Verwendung bei Variablen wird ein solcher Wahrheitswert bei Abfragen und Schleifen für die sogenannte **Bedingung** benötigt. Hier wird oftmals nicht eine Variable vom Typ Boolean abgefragt, sondern über bestimmte Operatoren ein Wert vom Typ Boolean erzeugt. Die folgende Tabelle zeigt eine Auflistung von diesen Vergleichsoperatoren:

a == b	Wert a ist gleich b
a === b	Wert und Typ a ist gleich b
a != b	Wert a ist ungleich b
a !== b	Wert und Typ a ist ungleich b
a > b	Wert a ist größer als b
a >= b	Wert a ist größer als oder gleich b
a < b	Wert a ist kleiner als b
a <= b	Wert a ist kleiner als oder gleich b

**Wichtig:** Beim Vergleich mittels === und !== wird im Gegensatz zu anderen Operatoren keine automatische Konvertierung durchgeführt, so ergibt z. B. 123 ==

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Variablen und Datentypen](#)

"123" true, die Bedingung 123 === "123" hingegen false.

**Übrigens:** Um mehrere Teilbedingung zu verbinden, gibt es das logische Und (Zeichen &&) und das logische Oder (Zeichen ||). Auch eine Kombination beider Operatoren ist möglich. Hier müssen dann jedoch evtl. Klammern gesetzt werden, um Teilbedingungen zu gruppieren.

```
1 | a > b && a < c
```

**Wichtig:** Natürlich können die mit diesen Operatoren erstellten Werte vom Typ `Boolean` nicht nur bei Bedingungen verwendet werden, sondern auch einer Variablen zugewiesen werden.

## Array

Bei einem Array handelt es sich um einen sogenannten komplexen Datentyp, welcher es erlaubt, **mehrere Werte innerhalb einer Variablen** zu speichern. Die Deklaration eines Arrays ist mit der normalen Variablendeklaration zu vergleichen. Bei der Zuweisung eines Arrays wird als Wert das Schlüsselwort `new` sowie der Datentypname `Array`, gefolgt von einem runden Klammernpaar, angegeben. Innerhalb der Klammer können wir nun (falls gewünscht) das Array bereits mit Werten füllen. Diese können dabei von einem beliebigen Datentyp sein und müssen wie Funktionsparameter mit einem Komma getrennt angegeben werden.

```
1 | var namensListe:Array = new Array("Peter", "Lisa", "Kai", "Anna");
```

Um auf ein einzelnes Element des Arrays zuzugreifen, notieren wir den Variablennamen gefolgt von einem Paar von eckigen Klammern. Innerhalb der Klammern wird der sogenannte **Index** angegeben. Der erste Wert hat den Index 0, der zweite Wert den Index 1, der dritte Wert den Index 2 usw.. Der Wert kann über diese sogenannte **Indexierung** lediglich gelesen werden (z. B. zur Ausgabe), aber auch verändert werden.

```
1 | trace(namensListe[1]); // Gibt "Lisa" aus
```

Ein Array kann jederzeit verändert werden. Hierfür dienen die Funktionen `push()`, `pop()`, `unshift()` und `shift()`. `push()` und `unshift()` erlauben das **Hinzufügen eines Wertes** zum Array. `pop()` und `shift()` hingegen entfernen jeweils einen Wert. `push()` und `pop()` beziehen sich auf das Ende des Arrays, d. h. die Funktion `push()` fügt einen Wert (auch als Element bezeichnet) an das Ende an und `pop()` entfernt das letzte Element. Im Gegensatz dazu bezieht sich `unshift()` und `shift()` auf den Array-Anfang. Die Funktionen `pop()` und `shift()` geben jeweils den entfernten Wert des Arrays zurück. Die Eigenschaft `length` enthält einen Wert, welcher die Länge des Arrays (also die Anzahl der darin enthaltenen Werte) widerspiegelt. Mit Hilfe der Funktion `indexOf()` und `lastIndexOf()` lassen sich **Elemente im Array suchen**. Dabei sind die Funktionen mit den bereits bekannten String-Funktionen `indexOf()` und `lastIndexOf()` zu vergleichen. Auch hier muss beachtet werden, dass das erste Element den Index 0 besitzt (nicht 1). Die Funktion `join()` fügt **alle Werte in einer Zeichenkette** zusammen. Als Parameter wird der Funktion das dafür zu verwendete Trennzeichen übergeben. Die Funktion `sort()` erlaubt das Sortieren des Arrays. Wird der Funktion kein Parameter übergeben, so erfolgt die Sortierung nach den darin enthaltenen Zeichen (mit Beachtung der Groß- und Kleinschreibung). Eine numerische Sortierung kann mit der Konstante `Array.NUMERIC` als Parameter durchgeführt werden.

```
1 | var zahlenListe:Array = new Array(12, 43, 23, 8, 37, 49, 15, 6, 31, 28);
2 |
3 | // Array verändern
4 | trace("Erstes Element (entfernt): " + zahlenListe.shift());
5 | trace("Letztes Element (entfernt): " + zahlenListe.pop());
6 | zahlenListe.unshift(13);
7 | zahlenListe.push(24);
8 |
9 | // Informationen abrufen
10 | trace("Länge: " + zahlenListe.length);
11 | trace("Position von 12: " + zahlenListe.indexOf(12)); // Wird nicht gefunden, da vorher entfernt
12 | trace("Position von 37: " + zahlenListe.indexOf(37));
13 | trace("Position von 24: " + zahlenListe.indexOf(24)); // Wird gefunden, da vorher hinzugefügt
14 |
15 | // Listenausgabe und sortieren
16 | trace("Liste (Ursprung): " + zahlenListe.join(", "));
17 | zahlenListe.sort();
18 | trace("Liste (Sort.): " + zahlenListe.join(", "));
19 | zahlenListe.sort(Array.NUMERIC);
20 | trace("Liste (Num. Sort.): " + zahlenListe.join(", "));
```

### Ausgabe:

```
1 | Erstes Element (entfernt): 12
2 | Letztes Element (entfernt): 28
3 | Länge: 10
4 | Position von 12: -1
5 | Position von 37: 4
6 | Position von 24: 9
7 | Liste (Ursprung): 13, 43, 23, 8, 37, 49, 15, 6, 31, 24
8 | Liste (Sort.): 13, 15, 23, 24, 31, 37, 43, 49, 6, 8
9 | Liste (Num. Sort.): 6, 8, 13, 15, 23, 24, 31, 37, 43, 49
```



## Vektor

Bei dem Datentyp `Vector` handelt es sich um eine Abwandlung des Array-Datentyps. Vektoren (engl. *vector*) sind **an einen bestimmten Datentyp** gebunden und können zudem „fixiert“ werden. Die Deklaration von Vektoren ist etwas komplexer, da in der Deklaration noch der Datentyp angegeben wird. Innerhalb der Klammern können optional zwei Parameter angegeben werden, wodurch die Länge (1. Parameter) und die Fixierung (2. Parameter) angegeben werden kann. Wird ein Vektor fixiert, so umfasst der Datentyp immer die festgelegte Anzahl an Werten. Das dynamische Hinzufügen oder Entfernen von Werten ist dann nicht möglich.

```
1 | var namensListe:Vector.<String> = new Vector.<String>();
2 | var temperaturWerte:Vector.<Number> = new Vector.<Number>(100, true); // Fixierter Vektor mit 100 Werten
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--





Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Variablen und Datentypen](#)

Des Weiteren bleiben alle oben beschriebenen Array-Eigenschaften und -Funktionen erhalten. Jedoch gilt zu beachten, dass die Funktionen `push()`, `pop()`, `shift()` und `unshift()` bei einem fixierten Vektor nicht verwendet werden darf. Ob ein Vektor fixiert ist, kann mit der Eigenschaft `fixed` abgerufen werden. Der Zugriff mittels der Indexierung wird auch beim Vektor benutzt.

```

1  var zahlenListe:Vector.<Number> = new Vector.<Number>(10, true);
2
3  zahlenListe[0] = 12;
4  zahlenListe[1] = 43;
5  zahlenListe[2] = 23;
6  zahlenListe[3] = 8;
7  zahlenListe[4] = 37;
8  zahlenListe[5] = 49;
9  zahlenListe[6] = 15;
10 zahlenListe[7] = 6;
11 zahlenListe[8] = 31;
12 zahlenListe[9] = 28;
13
14 // Informationen abrufen
15 trace("Fixiert: " + zahlenListe.fixed);
16 trace("Länge: " + zahlenListe.length);
17 trace("Position von 43: " + zahlenListe.indexOf(43));
18 trace("Position von 15: " + zahlenListe.indexOf(15));
19
20 // Listenausgabe
21 trace("Liste: " + zahlenListe.join(", "));
    
```

**Ausgabe:**

```

1  Fixiert: true
2  Länge: 10
3  Position von 43: 1
4  Position von 15: 6
5  Liste: 12, 43, 23, 8, 37, 49, 15, 6, 31, 28
    
```



**Datum**

Um in AS einen Datumswert (ggf. mit Uhrzeit) darzustellen, benötigen wir den Datentyp `Date`. Die Erzeugung eines `Date`-Datentyps erfolgt ebenfalls mit dem Schlüsselwort `new`, dem Datentypname und einem runden Klammernpaar. Wird hier kein Parameter angegeben, so enthält die `Date`-Variable das aktuelle Datum mit Uhrzeit. Alternativ kann dieser sogenannten Konstruktorkfunktion ein **Millisekundenwert** (UNIX-Zeitstempel: Millisekunden seit dem 1.1.1970 00:00:00 Uhr) übergeben werden. Des Weiteren ist es auch möglich, der Funktion das Jahr, den Monat, den Tag (des Monats), die Stunden, die Minuten, die Sekunden und die Millisekunden als Parameter in der gerade genannten Reihenfolge zu übergeben.

Zum **Auslesen und Setzen der Datums- und Uhrzeitbestandteile** gibt es einige Funktionen und Eigenschaften, die in den folgenden Tabellen aufgelistet sind:

Lokalzeit			
Eigenschaft	Lese-Funktion	Setz-Funktion	Beschreibung
fullYear	getFullYear()	setFullYear()	Jahr
month	getMonth()	setMonth()	Monat (0 für Januar, 1 für Februar, ..., 11 für Dezember)
date	getDate()	setDate()	Tag des Monats (0 - 31)
day	getDay()	-	Tag der Woche (0 für Sonntag, 1 für Montag, ..., 6 für Samstag)
hours	getHours()	setHours()	Stunde (0 - 23)
minutes	getMinutes()	setMinutes()	Minute (0 - 59)
seconds	getSeconds()	setSeconds()	Sekunde (0 - 59)
milliseconds	getMilliseconds()	setMilliseconds()	Millisekunde (0 - 999)

UTC-Zeit			
Eigenschaft	Lese-Funktion	Setz-Funktion	Beschreibung
fullYearUTC	getUTCFullYear()	setUTCFullYear()	Jahr
monthUTC	getUTCMonth()	setUTCMonth()	Monat (0 für Januar, 1 für Februar, ..., 11 für Dezember)
dateUTC	getUTCDate()	setUTCDate()	Tag des Monats (0 - 31)
dayUTC	getUTCDay()	-	Tag der Woche (0 für Sonntag, 1 für Montag, ..., 6 für Samstag)
hoursUTC	getUTCHours()	setUTCHours()	Stunde (0 - 23)
minutesUTC	getUTCMinutes()	setUTCMinutes()	Minute (0 - 59)
secondsUTC	getUTCSeconds()	setUTCSeconds()	Sekunde (0 - 59)
millisecondsUTC	getUTCMilliseconds()	setUTCMilliseconds()	Millisekunde (0 - 999)

Die Funktionen `toDateString()`, `toTimeString()`, `toString()` und `toUTCString()` geben eine Zeichenkette mit dem Datum bzw. der Uhrzeit zurück.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Variablen und Datentypen](#)

Das Format der Ausgabe kann nicht beeinflusst werden und ist von Adobe festgelegt. Ist ein anderes Format gewünscht, so müssen Sie das Zusammenbauen der Bestandteile selbst übernehmen.

```

1 | var aktuellesDatum:Date = new Date();
2 | var webseitenGruendung:Date = new Date(2013, 0, 29, 20, 0, 0, 0);
3 |
4 | trace("Aktuelles Datum: " + aktuellesDatum.toString());
5 | trace("Aktuelle Uhrzeit: " + aktuellesDatum.getTimeString());
6 | trace("Aktuelles Datum mit Uhrzeit (UTC): " + aktuellesDatum.toUTCString());
7 |
8 | trace("Die Webseite wurde am " + webseitenGruendung.date + "." + (webseitenGruendung.month + 1) + "." +
9 | webseitenGruendung.fullYear + " um " + webseitenGruendung.hours + " Uhr gegründet!");
10| webseitenGruendung.fullYear = aktuellesDatum.fullYear;
11| trace("Die Webseite wird / wurde am " + webseitenGruendung.date + "." + (webseitenGruendung.month + 1) + "." +
12| webseitenGruendung.fullYear + " um " + webseitenGruendung.hours + " Uhr " + (aktuellesDatum.fullYear - 2013) + " Jahre
    alt!");
    
```

**Ausgabe:**

```

1 | Aktuelles Datum: Wed Aug 10 2016
2 | Aktuelle Uhrzeit: 12:11:33 GMT+0200
3 | Aktuelles Datum mit Uhrzeit (UTC): Wed Aug 10 10:11:33 2016 UTC
4 | Die Webseite wurde am 29.1.2013 um 20 Uhr gegründet!
5 | Die Webseite wird / wurde am 29.1.2016 um 20 Uhr 3 Jahre alt!
    
```



**Übrigens:** Sie können in ActionScript auch sogenannte Konstanten definieren. **Konstanten** müssen bei der Deklaration zugewiesen werden und können danach nicht mehr verändert werden. Bei Konstanten wird an Stelle des Schlüsselworts `var` `const` verwendet.

```

1 | const mwstFaktor = 0.19;
    
```

**Wichtig:** Uniinitialisierte Variablen besitzen je nach Typ einen anderen Wert. `false` für Boolean, `0` für `int` und `uint`, `NaN` für Number und `null` für Object und `String`. `null` ist ein Wert, welcher eine fehlende Referenz angibt. Objekte (dazu zählen auch `Array`, `Vector` und `Date`) werden auf dem sogenannten Heap gespeichert. In der Variablen selbst wird daher nicht das ganze Objekt, sondern lediglich eine Referenz auf einen Speicher gespeichert. Fehlt diese Referenz (wie es der Fall ist, wenn eine Objekt-Variablen deklariert jedoch nicht zugewiesen ist) so ist der Wert eben `null`.

**Über uns**

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

**Community**

- Blog
- Forum
- News

**Nachschlagewerk**

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislöcher



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Bedingungen](#)

## Bedingungen

Mit Bedingungen ist es möglich, **Abfragen durchzuführen** und somit auf sogenannte Fälle abzutprüfen (**Fallunterscheidung**). Wir können also damit Rückgabewerte von Funktionen oder Werte von Variablen prüfen und darauf individuell reagieren. AS bietet hier die einfache Verzweigung mittels `if` und die mehrfache Verzweigung mittels `switch-case` an.

### Inhalt dieser Seite:

1. Einfache Verzweigung
2. Mehrfache Verzweigung

### Einfache Verzweigung

Einfache Verzweigungen (einfache Abfragen) werden durch das Schlüsselwort `if` gekennzeichnet. Nach dem Schlüsselwort wird in runden Klammern die Bedingung (zur Erinnerung: es muss sich dabei um einen Wert vom Typ `Boolean` handeln) notiert. Anschließend wird ein Block angegeben, welcher nur ausgeführt wird, wenn die **Bedingung zutrifft**.

```
1  if (a == b)
2  {
3      // Anweisungen wenn der Fall zutrifft
4  }
```

Natürlich ist es auch möglich, darauf zu reagieren, wenn die **Bedingung nicht zutrifft**. So ist es neben der Verwendung des NOT-Operators (z. B. `!benutzerAngemeldet`) zum „Wechseln“ des `Boolean`-Werts (aus `!true` wird `false` und aus `!false` wird `true`) auch möglich, beide Fälle (Bedingung trifft zu und Bedingung trifft nicht zu) abzutprüfen. Hierzu notieren wir nach dem `if`-Block das Schlüsselwort `else` mit einem weiteren Block.

```
1  if (a == b)
2  {
3      // Anweisungen wenn der Fall zutrifft
4  }
5  else
6  {
7      // Anweisungen wenn der Fall nicht zutrifft
8  }
```

Sowohl innerhalb des `if`-Blocks als auch des `else`-Blocks können weitere Verzweigungen folgen:

```
1  if (a == b)
2  {
3      // Anweisungen wenn der Fall zutrifft
4  }
5  else
6  {
7      if (c == d)
8      {
9          // Anweisungen wenn der Fall zutrifft
10     }
11     else
12     {
13         // Anweisungen wenn der Fall nicht zutrifft
14     }
15 }
```

Eine verkürzte Schreibweise für **weitere Verzweigungen** im `else`-Block können mittels `else if` erreicht werden. Das obige Beispiel mit Verwendung von `else if` sieht dann wie folgt aus:

```
1  if (a == b)
2  {
3      // Anweisungen wenn der 1. Fall zutrifft
4  }
5  else if (c == d)
6  {
7      // Anweisungen wenn der 2. Fall zutrifft
8  }
9  else
10 {
11     // Anweisungen wenn beide Fälle nicht zutreffen
12 }
```

Des Weiteren ist es möglich, das geschweifte Klammernpaar wegzulassen, sofern sich innerhalb einer der Blöcke nur eine auszuführende Anweisung befindet. Das folgende Beispiel zeigt die Überprüfung von zwei Variablen, bei welcher je nach Wert eine bestimmte Meldung ausgegeben wird:

```
1  var a:int = 12;
2  var b:int = 7;
3
4  if (a > b)
5      trace("a ist größer als b");
6  else if (a == b)
7      trace("a ist gleich b");
8  else
9      trace("a ist kleiner als b");
```

**Ausgabe:**

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Bedingungen](#)

1 | a ist größer als b



Wie auch in vielen anderen Programmiersprachen, ist es möglich, eine **In-Line-Abfrage** mittels `if-else` durchzuführen. Vorteil dieser Variante ist, dass diese wesentlich kürzer ist, jedoch muss darauf geachtet werden, dass diese Variante nur in bestimmten Fällen verwendet kann: Sowohl der `if`-Zweig als auch der `else`-Zweig müssen einen Wert zurückgeben. Dieser Wert kann von einem Funktionsaufruf zurückgegeben worden sein oder auch von einer anderen Variablen sowie einer Konstante kommen. Die In-Line-Verzweigung kann dann einer anderen Funktion übergeben oder einer Variablen zugewiesen werden. Zum Syntax: Zuerst wird die Bedingung angegeben, anschließend ein Fragezeichen `?` gefolgt von dem Wert, wenn die Bedingung zutrifft, einem Doppelpunkt `:` und dem Wert, wenn die Bedingung nicht zutrifft. Hierzu als erstes folgendes „ausführliches“ Beispiel:

```
1 | if (a == b)
2 |     c = "Wert gleich";
3 | else
4 |     c = "Wert ungleich"
```

In der In-Line-Variante sieht das obige Beispiel dann wie folgt aus:

```
1 | c = (a == b) ? "Wert gleich" : "Wert ungleich";
```

## Mehrfache Verzweigung

Mehrfache Verzweigungen werden mittels des Schlüsselworts `switch` gekennzeichnet. Dafür wird in dem folgenden runden Klammernpaar der zu **überprüfende Wert** angegeben (keine Bedingung!).

```
1 | switch (a)
2 | {
3 |
4 | }
```

Die einzelnen Fälle werden nun mit dem Schlüsselwort `case` gekennzeichnet. Danach folgt der Wert, welcher mit dem Wert von oben (siehe `switch`) verglichen werden soll. Zum Schluss folgt noch ein Doppelpunkt `:`. Nun können wir den auszuführenden Code notieren. Dabei ist die Verwendung eines Blocks mit geschweiften Klammern nicht notwendig. Um den `case`-Block zu beenden und dem Compiler damit mitzuteilen, dass wir am Ende der Anweisungen für diesen `case`-Block sind, benötigen wir das Schlüsselwort `break` mit darauffolgendem Semikolon `;`.

```
1 | case 1:
2 |     // Anweisungen welche ausgeführt werden sollen
3 |     break;
```

Von den `case`-Blöcken können wir selbstverständlich mehrere notieren. Wollen wir zusätzlich den Fall abprüfen, dass **keine der Bedingungen zutrifft**, so können wir den `default`-Block verwenden:

```
1 | default:
2 |     // Anweisungen welche ausgeführt werden sollen
3 |     break;
```

Hier nun folgendes vollwertiges Beispiel einer Verzweigung mittels `switch-case`:

```
1 | switch ((new Date()).month + 1)
2 | {
3 |     case 1:
4 |         trace("Januar");
5 |         break;
6 |     case 2:
7 |         trace("Februar");
8 |         break;
9 |     case 3:
10 |        trace("März");
11 |        break;
12 |     case 4:
13 |        trace("April");
14 |        break;
15 |     case 5:
16 |        trace("Mai");
17 |        break;
18 |     case 6:
19 |        trace("Juni");
20 |        break;
21 |     case 7:
22 |        trace("Juli");
23 |        break;
24 |     case 8:
25 |        trace("August");
26 |        break;
27 |     case 9:
28 |        trace("September");
29 |        break;
30 |     case 10:
31 |        trace("Oktober");
32 |        break;
33 |     case 11:
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Bedingungen](#)

```

34     trace("November");
35     break;
36     case 12:
37         trace("Dezember");
38         break;
39 }

```

## Ausgabe:

1 | August



**Wichtig:** Mit der `switch-case`-Verzweigung werden lediglich einzelne Werte überprüft. Komplexe Abfragen mit mehreren Variablen oder Wertebereichen werden daher mit `if`-Verzweigungen gelöst.

**Übrigens:** Jeder Code einer `switch-case`-Verzweigung könnte sich mittels einer `if-else`-Verzweigung ersetzen lassen. Auch wenn die `switch-case`-Verzweigung einen längeren Code erzeugt, ist diese wesentlich **übersichtlicher** und gilt für solche Fälle als **besserer Programmierstil**.

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: Homepage-Webhilfe » ActionScript » Schleifen

## Schleifen

Mit Hilfe von Schleifen können Sie bestimmte **Vorgänge mehrmals ausführen** bzw. eine Liste (z. B. Array oder Vektor) durchgehen. In AS gibt es 5 verschiedene Schleifen: `for`, `while`, `do-while`, `for-in` und `for-each`.

Jede Schleife besitzt einen Block, welcher u. U. mehrmals ausgeführt und als **Schleifenrumpf** bezeichnet wird. Die Bedingung, welche ebenfalls zu einer Schleife gehört, mit allen dazu gehörigen Schlüsselwörtern und Anweisungen wird als **Schleifenkopf** oder **Schleifenfuß** bezeichnet.

Um eine Schleife frühzeitig zu verlassen, notieren wir das Schlüsselwort `break` (mit Semikolon). Die Anweisung `continue;` führt dazu, dass die Ausführung des Schleifenrumpfs beendet und erneut zum Schleifenkopf gesprungen wird. Einer erneuten Ausführung des Schleifenrumpfs steht dabei nichts im Wege, sofern die Bedingung noch zutrifft.

**Inhalt dieser Seite:**

1. Zählschleife
2. Kopfgesteuerte Schleife
3. Fußgesteuerte Schleife
4. Array-Schleife

## Zählschleife

Eine Zählschleife wird dazu verwendet, einen bestimmten **Vorgang n mal auszuführen**. Gekennzeichnet ist diese Schleife durch das Schlüsselwort `for`. Hinter dem Schlüsselwort folgt ein rundes Klammernpaar, in welchem drei Teile angegeben werden müssen und durch Semikolon getrennt werden: Variablendeklaration oder -zuweisung, Bedingung und Anweisung nach dem Ausführen des Schleifenkopfs. Die Anweisung im 3. Teil des Schleifenkopfs wird bei der klassischen Zählschleife zur **Inkrementierung einer Zählervariablen** (meistens mit dem Name `i`, `j` etc.) verwendet und nach jedem Durchlauf des Schleifenrumpfs ausgeführt. Die Bedingung wird vor jedem Durchlauf der Schleife geprüft, weshalb die `for`-Schleife auch zu den sogenannten kopfgesteuerten Schleifen gehört.

```

1 | for (var i:int = 0; i < 10; i++)
2 |     trace(i);

```

**Ausgabe:**

```

1 | 0
2 | 1
3 | 2
4 | 3
5 | 4
6 | 5
7 | 6
8 | 7
9 | 8
10| 9

```



## Kopfgesteuerte Schleife

Eine kopfgesteuerte Schleife ist eine einfache Schleife, deren **Bedingung vor der Ausführung des Schleifenrumpfs geprüft** wird. Der Schleifenkopf besteht aus dem Schlüsselwort `while` und der Bedingung (angegeben in runden Klammern). Das untere Beispiel zeigt, wie das obige Beispiel mit der `for`-Schleife mittels einer `while`-Schleife ersetzt werden kann.

```

1 | var i:int = 0;
2 |
3 | while (i < 10)
4 | {
5 |     trace(i);
6 |
7 |     i++;
8 | }

```

**Ausgabe:**

```

1 | 0
2 | 1
3 | 2
4 | 3
5 | 4
6 | 5
7 | 6
8 | 7
9 | 8
10| 9

```



**Wichtig:** In der Praxis sollten Sie für einen solchen Zählvorgang ausschließlich die `for`-Schleife und nicht die `while`-Schleife verwenden.

## Fußgesteuerte Schleife

Bei der `do-while`-Schleife wird die **Bedingung am Ende geprüft**, weshalb es sich um eine fußgesteuerte Schleife handelt. Daraus folgt, dass der Schleifenrumpf mindestens einmal ausgeführt wird, da die Bedingung vor der ersten Ausführung nicht geprüft wird. Vor dem Block notieren wir das Schlüsselwort `do`. Nach der schließenden geschweiften Klammer wird der Schleifenfuß, bestehend aus dem Schlüsselwort `while`, der Bedingung in runden Klammern und dem Semikolon, angegeben.

```

1 | var i:int = 0;
2 |

```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Schleifen](#)

```

3 | do
4 | {
5 |     trace(i);
6 |
7 |     i++;
8 | } while (i < 10);

```

**Ausgabe:**

```

1 | 0
2 | 1
3 | 2
4 | 3
5 | 4
6 | 5
7 | 6
8 | 7
9 | 8
10| 9

```



**Wichtig:** Auch hier möchten wir nochmals darauf hinweisen, dass die `do-while`-Schleife nicht als Ersatz für die `for`-Schleife verwendet werden soll und zudem kein Ersatz für die `while`-Schleife darstellt, da die Bedingung hier am Ende und bei den anderen zwei Schleifen am Anfang geprüft wird.

## Array-Schleife

Die sogenannte „Array-Schleife“ (auch Iterationsschleife genannt) ermöglicht das „durchgehen“ der Elemente eines Arrays, Vektors oder der Eigenschaften eines Objekts. Als Schleifen stehen uns hier die `for-in`-Schleife und die `for-each-in`-Schleife zur Verfügung. Bei der `for-in`-Schleife setzt sich der Schleifenkopf aus dem Schlüsselwort `for` und einem runden Klammernpaar, in welchem eine „temporäre“ Variable deklariert wird sowie das Schlüsselwort `in` und der Name des Arrays, Vektors oder Objekts angegeben wird, zusammen. Die „temporäre“ Variable enthält dabei den **Index für das Array oder den Vektor** bzw. den **Eigenschaftsnamen für das Objekt**.

```

1 | var sprachen:Array = new Array("HTML", "CSS", "JavaScript", "ActionScript", "PHP", "Perl", "ASP.NET");
2 | for (var sprache:int in sprachen)
3 |     trace(sprachen[sprache]);

```

Eine ähnliche Funktion bietet die `for-each-in`-Schleife. Hier wird nach dem Schlüsselwort `for` zusätzlich das Schlüsselwort `each` angegeben. Des Weiteren enthält die „temporäre“ Variable nicht den Index sondern den **Wert des Elements bzw. der Eigenschaft**. Das Beispiel von oben lässt sich mittels der `for-each-in`-Schleife also wie folgt darstellen.

```

1 | var sprachen:Array = new Array("HTML", "CSS", "JavaScript", "ActionScript", "PHP", "Perl", "ASP.NET");
2 |
3 | for each (var sprache:String in sprachen)
4 |     trace(sprache);

```

**Ausgabe:**

```

1 | HTML
2 | CSS
3 | JavaScript
4 | ActionScript
5 | PHP
6 | Perl
7 | ASP.NET

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Funktionen](#)

## Funktionen

### Inhalt dieser Seite:

1. Funktion mit Parametern
2. Funktion mit Rückgabewert

Funktionen dienen dazu, **Quellcode zu gliedern und zu strukturieren**. Des Weiteren werden Funktionen auch dazu verwendet, den dort notierten **Code mehrmals nutzen** zu können. Einige Funktionen (z. B. `parseInt()`, `parseFloat()`) und einige Klassen (z. B. `Math`, `Date`) sind in den sogenannten **Libraries** (zu Deutsch Bibliotheken) des SDKs enthalten. Diese können ohne weiteres verwendet werden. Wie man diese Funktionen bzw. Funktionen im Allgemeinen aufruft, haben wir ja bereits gelernt.

Nun wollen wir uns aber damit beschäftigen, wie man **eigene Funktionen definiert**. Funktionen befinden sich immer innerhalb einer Klasse (`class`-Block). Die Deklaration besteht dabei aus 5 Teilen: dem Zugriffsmodifizierer, dem Schlüsselwort `function`, dem Funktionsnamen, einem runden Klammernpaar (für Funktionsparameter) und dem Rückgabotyp. Zwischen dem runden Klammernpaar und dem Rückgabotyp muss zudem ein Doppelpunkt notiert werden. Auf die Funktionsparameter und den Rückgabotyp (aktuell verwenden wir hier `void`) gehen wir später noch genauer ein. Der Zugriffsmodifizierer kann `private`, `public` oder `protected` sein. Auf deren genaue Bedeutung gehen wir im [Thema Objektorientierung](#) nochmals ein. Der Funktionsname muss innerhalb der Klasse eindeutig sein und wird meistens ebenfalls in der Camel-Case-Schreibweise (mit kleinem Buchstaben am Anfang) angegeben.

Das folgende Beispiel zeigt den Aufruf der Funktion und darunter die Deklaration der Funktion:

```
1 | gebeHalloAus();
```

**Funktion gebeHalloAus():**

```
1 | private function gebeHalloAus():void
2 | {
3 |     trace("Hallo Welt!");
4 | }
```

**Ausgabe:**

```
1 | Hallo Welt!
```



## Funktion mit Parametern

Um Funktionen mit Parametern zu definieren, müssen wir bei der Funktionsdeklaration innerhalb der Klammern **Variablen deklarieren**. Dabei wird der Name gefolgt von einem Doppelpunkt `:` und dem Typ angegeben. Mehrere Variablen bzw. Parameter können mittels Komma `,` (wie beim Aufruf) getrennt werden. Um einen Parameter als **optional** zu kennzeichnen, notieren wir nach dem Datentyp ein Gleichheitszeichen und den für den Parameter geltenden **Standardwert**. Dieser Standardwert gilt immer dann, wenn kein Wert für den Parameter übergeben wurde. Eine Funktion kann selbstverständlich mehrere Parameter haben, wovon auch mehrere optional sein können, jedoch müssen sich alle optionale Parameter am Ende befinden. Beim Aufruf ist das Auslassen von Parametern nicht möglich. Lediglich das Weglassen von Parametern am Ende (von rechts nach links) ist möglich. Die Namen der Funktionsparameter dürfen sich nicht mit den Variablennamen innerhalb der Funktion oder untereinander überschneiden.

```
1 | gebeNamenAus("Kai");
2 | gebeNamenAus("Lisa");
3 | gebeNamenAus();
```

**Funktion gebeNamenAus():**

```
1 | private function gebeNamenAus(name:String = "Peter"):void
2 | {
3 |     trace("Hallo " + name + "!");
4 | }
```

**Ausgabe:**

```
1 | Hallo Kai!
2 | Hallo Lisa!
3 | Hallo Peter!
```



## Funktion mit Rückgabewert

Der Rückgabotyp ist bei der Funktionsdeklaration der letzte Teil und wird hinter dem Doppelpunkt angegeben. Bisher hatten alle unsere definierten Funktionen den Rückgabotyp `void`. `void` ist ein spezieller Datentyp, welcher angibt, dass die Funktion keinen Wert zurückgibt. Dieser kann jedoch (falls benötigt) durch einen anderen beliebigen Datentyp (z. B. eine Zahl, eine Zeichenkette, ein Objekt) ersetzt werden.

Um von der Funktion aus einen Wert zurückzugeben, müssen wir als Anweisung das Schlüsselwort `return`, gefolgt von dem zurückzugebenden Wert, angeben. Anweisungen, welche hinter der `return`-Anweisung stehen, werden nicht ausgeführt.

```
1 | trace("7 + 5 = " + addiereZahlen(7, 5));
2 | trace("1.2 + 8.9 = " + addiereZahlen(1.2, 8.9).toFixed(1));
```

**Funktion addiereZahlen():**

```
1 | private function addiereZahlen(a:Number, b:Number):Number
2 | {
3 |     return a + b;
4 | }
```

**Ausgabe:**

```
1 | 7 + 5 = 12
2 | 1.2 + 8.9 = 10.1
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Funktionen](#)



**Übrigens:** Bei Funktionen mit dem Rückgabtyp `void` können wir mittels der Anweisung `return;` ebenfalls aus der Funktion herausspringen.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislöcher



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Objektorientierung](#)

## Objektorientierung

Mit Hilfe von Klassen ist es möglich, eigene **komplexe Datentypen** zu definieren. Eine Klasse stellt das Modell bzw. den Bauplan für Objekte dar. Eine Klasse kann mehrere Variablen (die sogenannten globalen Variablen), Eigenschaften (dazu weiter unten mehr) und Funktionen enthalten.

Objekte werden zu Laufzeit mittels des `new`-Schlüsselworts erzeugt. Dieser Vorgang wird als sogenannte **Instanziierung** bezeichnet. Bei der Deklaration von Objekt-Variablen wird als Datentyp der Klassenname angegeben. Die instanziierte Objekt-Variable wird als **Instanz** bezeichnet. Klassen werden innerhalb des `package`-Blocks angegeben. Jede Klasse muss sich dabei in einer **eigenen Datei** befinden. Vor dem Klassenblock wird der Zugriffsmodifizierer (dazu gleich mehr), das Schlüsselwort `class` und der eindeutige **Name der Klasse** angegeben. Dabei wird zumeist ebenfalls die Camel-Case-Schreibweise verwendet, jedoch mit einem großen Buchstaben am Anfang des Namens. Innerhalb der Klasse können nun Variablen und Funktionen deklariert werden.

Sowohl für Klassen als auch für deren globale Variablen und Funktionen benötigen wir sogenannte **Zugriffsmodifizierer**. Hier stehen uns die Schlüsselwörter `private`, `protected` und `public` zur Verfügung. Bei der Deklaration von Klassen wird stets `public` verwendet. Die Verwendung von „privaten Klassen“ ist nicht zu empfehlen und wird von dem Flex Compiler nicht mehr unterstützt, weshalb wir darauf nicht genauer eingehen. Doch nun zurück zu den Unterschieden zwischen den Zugriffsmodifizierern. Mittels `public` ist ein Zugriff von der Klasse selbst sowie von „außen“ möglich. `private` erlaubt den Zugriff auf die Variable bzw. Funktion **nur von innerhalb der Klasse**. Der Zugriffsmodifizierer `protected` ist mit `private` zu vergleichen, jedoch ist bei `protected` ein Zugriff von der Kindklasse auf die Variable oder Funktion der Elternklasse (siehe Vererbung) möglich.

In AS3 und anderen Programmiersprachen hat es sich eingebürgert, bei privaten Variablen einen Unterstrich `_` voranzustellen (siehe Beispiel). Das folgende Beispiel zeigt die Klasse `Computer`, welche eine Variable für das Betriebssystem (`betriebsSystem`) und zwei Funktionen zum Hoch- und Runterfahren des Computers hat. Über die private Variable `_gestartet`, kann innerhalb der Klasse festgestellt werden, ob der Computer bereits gestartet ist. Der erste Code zeigt die Objektinstanziierung sowie einige Zugriffe auf das Objekt. Im zweiten Code ist der Inhalt der Datei für die `Computer`-Klasse zu sehen.

```
1 var meinComputer:Computer = new Computer();
2 meinComputer.betriebsSystem = "Windows 7";
3 meinComputer.starteComputer();
4 meinComputer.starteComputer(); // Computer läuft bereits
5 meinComputer.stoppeComputer();
6 meinComputer.stoppeComputer(); // Computer ist bereits heruntergefahren
7 meinComputer.starteComputer();
```

### Computer.as

```
1 package de.hwh.bsp.obj
2 {
3     public class Computer
4     {
5         private var _gestartet:Boolean = false;
6         public var betriebsSystem:String;
7
8         public function starteComputer():void
9         {
10            if (!_gestartet)
11            {
12                trace(betriebsSystem + " wird gestartet ...");
13                _gestartet = true;
14            }
15            else
16                trace("Computer bereits gestartet!");
17        }
18
19        public function stoppeComputer():void
20        {
21            if (_gestartet)
22            {
23                trace(betriebsSystem + " wird heruntergefahren ...");
24                _gestartet = false;
25            }
26            else
27                trace("Computer läuft nicht!");
28        }
29    }
30 }
```

### Ausgabe:

```
1 Windows 7 wird gestartet ...
2 Computer bereits gestartet!
3 Windows 7 wird heruntergefahren ...
4 Computer läuft nicht!
5 Windows 7 wird gestartet ...
```

**Wichtig:** In der Praxis sollten Variablen niemals mittels des Zugriffsmodifizierers `public` von „außen“ zugänglich gemacht werden. Zum Ändern eines Variablenwertes sollten stets Funktionen (z. B. `getX()` und `setX()`) oder Eigenschaften (siehe weiter unten) verwendet werden.

**Übrigens:** Sollten Sie innerhalb einer Klasse einen globalen Variablennamen haben, welcher den gleichen Namen wie eine lokale Variable oder ein

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Objektorientierung](#)

Funktionsparameter hat, dann können wir mittels des `this`-Schlüsselworts eine Unterscheidung durchführen. Das `this`-Schlüsselwort verweist immer auf das aktuelle Objekt. Eine Verwendung von `this` können Sie in den zwei folgenden Beispielen unten sehen.

## Konstruktor

Der sogenannte Konstruktor ist eine spezielle Funktion, welche bei der Instanziierung ausgeführt wird. Deklariert wird die Funktion, so wie jede andere auch, jedoch muss beachtet werden, dass der Name der Konstruktorfunktion immer **dem Klassennamen entsprechen** muss. Des Weiteren fallen bei der Deklaration der Rückgabebetyp sowie der vorangestellte Doppelpunkt weg. Das Verwenden von Parametern (und ggf. optionalen Parametern) bei der Konstruktorfunktion ist ebenfalls möglich (siehe Beispiel). Übergeben werden die Werte für die Konstruktorfunktion direkt in dem runden Klammernpaar bei Objektinstanziierung.

```
1 var meinComputer:Computer = new Computer("Windows 7");
2 meinComputer.starteComputer();
3 meinComputer.stoppeComputer();
4 meinComputer.starteComputer();
```

### Computer.as

```
1 package de.hwh.bsp.objkonstruktor
2 {
3     public class Computer
4     {
5         private var _gestartet:Boolean = false;
6         public var betriebsSystem:String;
7
8         public function Computer(betriebsSystem:String)
9         {
10            this.betriebsSystem = betriebsSystem;
11        }
12
13        public function starteComputer():void
14        {
15            if (!_gestartet)
16            {
17                trace(betriebsSystem + " wird gestartet ...");
18                _gestartet = true;
19            }
20            else
21                trace("Computer bereits gestartet!");
22        }
23
24        public function stoppeComputer():void
25        {
26            if (_gestartet)
27            {
28                trace(betriebsSystem + " wird heruntergefahren ...");
29                _gestartet = false;
30            }
31            else
32                trace("Computer läuft nicht!");
33        }
34    }
35 }
```

### Ausgabe:

```
1 Windows 7 wird gestartet ...
2 Windows 7 wird heruntergefahren ...
3 Windows 7 wird gestartet ...
```



## Vererbung

Bei der Vererbung wird eine **bestehende Klasse um weitere Variablen, Eigenschaften und Funktionen erweitert**. Bei der Klasse, die Bestandteile von einer anderen Klassen erben soll, wird bei der Deklaration am Ende das Schlüsselwort `extends` und der Name der sogenannten Elternklasse angegeben. Bei der **Elternklasse** handelt es sich um die Klasse, von welcher Variablen, Eigenschaften und Funktionen geerbt werden sollen. Bei der Klasse, die von einer anderen erbt, spricht man von der **Kindklasse**. Über das Schlüsselwort `super` können wir von der Kindklasse auf die Elternklasse zugreifen (z. B. um den Konstruktor der Elternklasse aufzurufen). Des Weiteren ist es möglich, **Funktionen der Elternklasse zu überschreiben**. Hierfür notieren wir das Schlüsselwort `override` vor dem Zugriffsmodifizierer. Auch hier ist weiterhin ein Aufrufen der überschriebenen Funktion (die zur Elternklasse gehört) möglich (siehe Beispiel). Im folgenden Beispiel handelt es sich bei der Klasse `Computer` um die Elternklasse (auch als Basisklasse bezeichnet). Die Klasse `Notebook` hingegen ist die Kindklasse.

```
1 var meinNotebook:Notebook = new Notebook("Windows 7", 15.6);
2 meinNotebook.starteComputer();
3 meinNotebook.stoppeComputer();
```

### Computer.as

```
1 package de.hwh.bsp.objvererbung
2 {
3     public class Computer
```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Objektorientierung](#)

```

4     {
5         private var _gestartet:Boolean = false;
6         public var betriebsSystem:String;
7
8         public function Computer(betriebsSystem:String)
9         {
10            this.betriebsSystem = betriebsSystem;
11        }
12
13        public function starteComputer():void
14        {
15            if (!_gestartet)
16            {
17                trace(betriebsSystem + " wird gestartet ...");
18                _gestartet = true;
19            }
20            else
21                trace("Computer bereits gestartet!");
22        }
23
24        public function stoppeComputer():void
25        {
26            if (_gestartet)
27            {
28                trace(betriebsSystem + " wird heruntergefahren ...");
29                _gestartet = false;
30            }
31            else
32                trace("Computer läuft nicht!");
33        }
34    }
35 }

```

## Notebook.as

```

1 package de.hwh.bsp.objvererbung
2 {
3     public class Notebook extends Computer
4     {
5         public var displayGroesse:Number;
6
7         public function Notebook(betriebsSystem:String, displayGroesse:Number)
8         {
9             super(betriebsSystem);
10
11            this.displayGroesse = displayGroesse;
12        }
13
14        override public function starteComputer():void
15        {
16            super.starteComputer();
17
18            trace("Ihr Bildschirm mit " + displayGroesse.toFixed(1) + " Zoll wird initialisiert ...");
19        }
20    }
21 }

```

## Ausgabe:

```

1 | Windows 7 wird gestartet ...
2 | Ihr Bildschirm mit 15.6 Zoll wird initialisiert ...
3 | Windows 7 wird heruntergefahren ...

```



## Eigenschaften

Eigenschaften sind die „besseren Variablen“. Wie bereits oben erwähnt, sollten Variablen nicht mittels `public` von außen zugänglich gemacht werden. Eigenschaften lösen dieses Problem, denn in Eigenschaften können wir einen Programmcode (z. B. zum Überprüfen des zu setzenden Wertes) notieren, wie wenn es eine Funktion wäre. Vorteil gegenüber von Funktionen ist jedoch, dass sowohl zum Lesen als auch zum Setzen der gleiche Name verwendet werden kann. Zudem ist die Zuweisung von außen mittels des Zuweisungsoperators einfacher zu lesen als der Aufruf einer Funktion. Zur Speicherung des Wertes einer Eigenschaft wird eine (interne / private) Variable benötigt.

Eigenschaften sehen bei der Deklaration sehr ähnlich wie Funktionen aus. Zwischen dem Schlüsselwort `function` und dem Funktionsnamen bzw. Eigenschaftsnamen wird das Schlüsselwort `get` oder `set` notiert. Dadurch ist es möglich, nur lesbare, nur änderbare und les- und änderbare Eigenschaften zu notieren. Die `get`-Funktion der Eigenschaft hat immer einen Rückgabewert, wohingegen die `set`-Funktion immer einen Übergabeparameter hat.

```

1 | var meinComputer:Computer = new Computer();
2 | meinComputer.betriebsSystem = "";
3 | meinComputer.starteComputer();

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Objektorientierung](#)

```
4 meinComputer.stoppeComputer();
5 meinComputer.betriebsSystem = "Windows 7";
6 meinComputer.starteComputer();
```

## Computer.as

```
1 package de.hwh.bsp.objgest
2 {
3     public class Computer
4     {
5         private var _gestartet:Boolean = false;
6         private var _betriebsSystem:String = "Windows 10";
7
8         public function get betriebsSystem():String
9         {
10            return _betriebsSystem;
11        }
12        public function set betriebsSystem(betriebsSystem:String):void
13        {
14            if (betriebsSystem != "")
15                _betriebsSystem = betriebsSystem;
16        }
17
18        public function starteComputer():void
19        {
20            if (!_gestartet)
21            {
22                trace(betriebsSystem + " wird gestartet ...");
23                _gestartet = true;
24            }
25            else
26                trace("Computer bereits gestartet!");
27        }
28
29        public function stoppeComputer():void
30        {
31            if (_gestartet)
32            {
33                trace(betriebsSystem + " wird heruntergefahren ...");
34                _gestartet = false;
35            }
36            else
37                trace("Computer läuft nicht!");
38        }
39    }
40 }
```

## Ausgabe:

```
1 Windows 10 wird gestartet ...
2 Windows 10 wird heruntergefahren ...
3 Windows 7 wird gestartet ...
```



## Packages

Mit Packages (oder auch als Paket bezeichnet) ist es möglich, **Klassen zu gruppieren** und zu bündeln. Ein Package ist immer der „oberste“ Block in einer Datei.

```
1 package de.hwh.bsp.oop
2 {
3
4 }
```

Die Namen von Packages bestehen zumeist aus mehreren Teilen und werden mittels des Punktoperators getrennt. So baut sich der Paketname von links nach rechts zusammen. Alle unsere Beispiele befinden sich z. B. im Package `de.hwh.bsp.*`. Dieser Name baut sich aus den Teilnamen `de` für Deutschland, `hwh` für Homepage-Webhilfe und `bsp` für Beispiel zusammen. Die Gruppierung von Klassen in unterschiedliche Packages wird lediglich für größere Projekte verwendet.

Mittels des Schlüsselworts `import` ist es möglich, **Packages oder sogar einzelne Klassen einzubinden**. Befindet sich z. B. Klasse `A` in einem anderen Package wie Klasse `B`, so muss in der Datei von Klasse `A` das Package von Klasse `B` importiert werden, sofern die Klasse `B` innerhalb der Klasse `A` verwendet wird. Die `import`-Anweisungen befinden sich direkt innerhalb des `package`-Blocks oberhalb der Klassendefinition. Über das Sternzeichen `*` lassen sich alle Klassen eines Packages einbinden.

```
1 import de.hwh.bsp.oop.A; // Importiert die Klasse A des Packages de.hwh.bsp.oop
1 import de.hwh.bsp.oop.*; // Importiert alle Klassen des Packages de.hwh.bsp.oop
```

**Übrigens:** Der Zugriff auf Variablen und Funktionen einer Klasse sind nur möglich, wenn wir ein Objekt daraus instanziiert haben. Um dies zu ändern, gibt es das Schlüsselwort `static`. Dieses Schlüsselwort wird zwischen Zugriffsmodifizierer und dem Schlüsselwort `var` (bei Variablen), `const` (bei Konstanten) oder `function` (bei Funktionen) angegeben. Geeignet sind **statische Klassenbestandteile** meistens nur für Konstanten (z. B. `Math.PI`) oder für Funktionen, die nicht

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

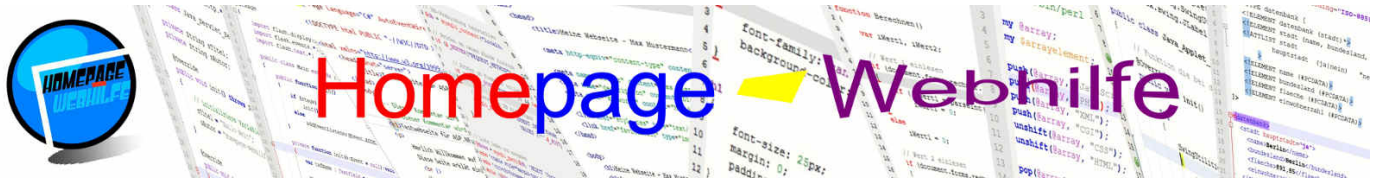
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Objektorientierung](#)

direkt mit einem Objekt zusammenhängen bzw. Hilfsfunktionen sind (z. B. `Math.ceil()`).

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Steuerelemente](#)

## Steuerelemente

Steuerelemente sind Bestandteile einer **Benutzeroberfläche** und dienen zur **Interaktion mit dem Bediener**. Da uns bei der Verwendung des Flex SDKs das Package `fl` und deren untergeordnete Packages und Klassen fehlen, sind die Möglichkeiten der Steuerelemente sehr begrenzt. Trotzdem lassen sich mit den hier vorgestellten Elementen einiges anstellen. Des Weiteren sollte beachtet werden, dass in Flash-Anwendungen vor allem grafische Bestandteile und Bilder verwendet werden, welche wir in diesem und im nächsten Thema ebenfalls vorstellen werden.

Um Steuerelemente (zumeist die Basisklasse `InteractiveObject`) zu erzeugen, wird ein Objekt der jeweiligen Klasse instanziiert. Um das Element einem „Container“ (z. B. der Klasse `Sprite`) hinzuzufügen, müssen wir die Funktion `addChild()` aufrufen. Zur Erinnerung: Unsere `Main`-Klasse ist von der Klasse `Sprite` abgeleitet. Steuerelemente können mittels der Eigenschaften `x` und `y` **positioniert** werden. Beide Werte sind auf den Wert `0` voreingestellt. Die Position `0|0` befindet sich in der Ecke oben links. Die Einstellung der **Größe** kann mittels der Eigenschaften `width` und `height` erfolgen. Unsere Anwendung hat bisher immer die Größe von `800x600` Pixeln. Diese Einstellung kann in FlashDevelop in den Projekteinstellungen (`Project` → `Properties`) jedoch jederzeit geändert werden. Eine weitere nützliche Eigenschaft ist `visible`, welche einen Wert vom Typ `Boolean` erwartet und es ermöglicht, das Element auszublenden. Mittels der Eigenschaft `name` können wir dem Steuerelement einen **Namen** geben. Diese Eigenschaft wird zumeist nur dann gesetzt, wenn wir später auf das Element mittels der Funktion `getChildByName()` des sogenannten **Anzeigeobjekt-Containers** (z. B. `Sprite`) zugreifen möchten. Das Entfernen eines Steuerelementes aus dem Container kann mittels der Funktion `removeChild()` durchgeführt werden. Dieser Funktion muss dabei das Objekt des zu entfernenden Steuerelements übergeben werden.

### Inhalt dieser Seite:

1. Textfeld
2. Button
3. Bild

## Textfeld

Ein Textfeld können wir mittels der Klasse `TextField` darstellen und erzeugen. Dieses Textfeld kann dabei nur zur Anzeige dienen (sozusagen ein „Label“), nicht änderbar jedoch selektierbar sein oder zur Eingabe durch den Benutzer verwendet werden. Über die Eigenschaft `text` kann der **anzuweisende Text** bzw. der **eingabegebene Text** abgerufen werden. Mittels der Eigenschaft `selectable` lässt sich steuern, ob der **Text markiert** werden kann. Standardmäßig ist dies erlaubt (Wert `true`). Mit der Eigenschaft `type` und dem Wert der statischen Konstante `INPUT` der `TextFieldType`-Klasse kann festgelegt werden, dass der **Text änderbar** ist.

Um die **Schrifteinstellungen** eines Textfelds zu ändern, müssen wir ein Objekt der `TextFormat`-Klasse erzeugen. Dem Konstruktor können dabei unter anderem die Schriftart (Eigenschaft `font`), die Schriftgröße (Eigenschaft `size`) und die Schriftfarbe (Eigenschaft `color`) übergeben werden. Weitere Eigenschaften sind u. a. `bold` (Fettdruck), `italic` (kursiver Text), `underline` (unterstrichener Text) und `align` (horizontale Ausrichtung). Zeichenketten-Konstanten für die Textausrichtung können aus der Klasse `TextFormatAlign` abgerufen werden. Um dem `TextField`-Objekt das `TextFormat`-Objekt zuzuweisen, können wir der Eigenschaft `defaultTextFormat` das `TextFormat`-Objekt zuweisen. Alternativ können wir auch die Funktion `setTextFormat()` verwenden. Hierbei muss jedoch beachtet werden, dass dieser Aufruf immer nach der Zuweisung der `text`-Eigenschaft erfolgen muss. Des Weiteren ist es möglich, der Funktion zwei weitere Parameter (Startindex und Endindex) zu übergeben, um somit die Formatierung nur auf einen **bestimmten Teil des Texts** anzuwenden.

```
1 var meinTextFormat:TextFormat = new TextFormat("Times", 30, 0xFF0000); // 0xFF0000 = rot
2 meinTextFormat.bold = true; // Fettdruck
3 meinTextFormat.italic = true; // kursiver Text
4 meinTextFormat.align = TextFormatAlign.RIGHT; // rechtsbündiger Text
```

Mittels der Eigenschaften `background` (Datentyp `Boolean`) und `backgroundColor` (Datentyp `uint`) ist es möglich, dem Textfeld eine **Hintergrundfarbe** zu geben. Durch die Eigenschaft `border` (Datentyp `Boolean`) und `borderColor` (Datentyp `uint`) kann ein **Rahmen** eingestellt werden. `displayAsPassword` legt fest, ob es sich bei dem Textfeld um ein Eingabefeld für Passwörter handelt, um somit die **Zeichen „verschlüsselt“ darzustellen**. Wollen wir mittels der `TextField`-Klasse einen **mehrzeiligen Text** erzeugen, so muss der Wert der Eigenschaft `multiline` auf `true` gesetzt werden.

```
1 // Überschrift "Formular"
2 var ueberschrift:TextField = new TextField();
3 var ueberschriftFormat:TextFormat = new TextFormat("Arial", 50);
4 ueberschriftFormat.bold = true;
5 ueberschriftFormat.align = TextFormatAlign.CENTER;
6 ueberschrift.y = 20;
7 ueberschrift.width = 800;
8 ueberschrift.height = 60;
9 ueberschrift.selectable = false;
10 ueberschrift.text = "Formular";
11 ueberschrift.setTextFormat(ueberschriftFormat);
12 addChild(ueberschrift);
13
14 // Beschriftungsfelder
15 var beschriftungVorname:TextField = new TextField();
16 var beschriftungNachname:TextField = new TextField();
17 var beschriftungFormat:TextFormat = new TextFormat("Arial", 20);
18 beschriftungFormat.align = TextFormatAlign.RIGHT;
19 beschriftungVorname.y = 100;
20 beschriftungNachname.y = 140;
21 beschriftungVorname.width = beschriftungNachname.width = 390;
22 beschriftungVorname.height = beschriftungNachname.height = 30;
23 beschriftungVorname.selectable = beschriftungNachname.selectable = false;
24 beschriftungVorname.text = "Vorname:";
25 beschriftungVorname.setTextFormat(beschriftungFormat);
26 beschriftungNachname.text = "Nachname:";
27 beschriftungNachname.setTextFormat(beschriftungFormat);
28 addChild(beschriftungVorname);
29 addChild(beschriftungNachname);
30
31 // Eingabefelder
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Steuerelemente](#)

```

32 var eingabeVorname:TextField = new TextField();
33 var eingabeNachname:TextField = new TextField();
34 var eingabeFormat:TextFormat = new TextFormat("Arial", 20);
35 eingabeVorname.x = eingabeNachname.x = 410;
36 eingabeVorname.y = 100;
37 eingabeNachname.y = 140;
38 eingabeVorname.width = eingabeNachname.width = 150;
39 eingabeVorname.height = eingabeNachname.height = 30;
40 eingabeVorname.border = eingabeNachname.border = true;
41 eingabeVorname.type = eingabeNachname.type = TextFieldType.INPUT;
42 eingabeVorname.defaultTextFormat = eingabeFormat;
43 eingabeNachname.defaultTextFormat = eingabeFormat;
44 addChild(eingabeVorname);
45 addChild(eingabeNachname);
    
```



## Button

Ein Button wird in AS mittels der Klasse `SimpleButton` dargestellt. Die Klasse `SimpleButton` enthält keinen darstellbaren Bereich, jedoch können dem Button verschiedene Steuerelemente zugewiesen werden, welche angezeigt werden sollen, wenn der Button sich in einem bestimmten Status befindet. Hierfür besitzt die `SimpleButton`-Klasse die vier Eigenschaften `hitTestState`, `overState`, `downState` und `upState`. `overState` ist aktiv, wenn sich unser **Mauszeiger innerhalb des Buttons** bzw. innerhalb des „Erkennungsrechtecks“ befindet, die Maustaste jedoch nicht gedrückt ist. `downState` ist mit `overState` zu vergleichen, jedoch mit dem Unterschied, dass beim `downState` die Maustaste und somit der **Button gedrückt** wird. Um das Drücken des Buttons zu erkennen, müssen wir das Klick-Event registrieren (dazu jedoch später mehr). Der Status `upState` ist aktiv, wenn sich der Mauszeiger **außerhalb der Schaltfläche** (engl. *button*) befindet. `hitTestState` legt das Anzeigebereich bzw. Steuerelement fest, welches zur sogenannten **Kollisionserkennung** dient. Das Element, welches in der `hitTestState`-Eigenschaft festgelegt ist, wird dabei nicht durch die `SimpleButton`-Klasse grafisch dargestellt (außer es wurde manuell mittels `addChild()` zum Container hinzugefügt). Aus diesem Grund wird der Eigenschaft `hitTestState` oft das gleiche Objekt wie der Eigenschaft `upState` zugewiesen (siehe Beispiel). Mit der Eigenschaft `useHandCursor` wird festgelegt, ob der **Hand-Cursor an Stelle des Pfeil-Cursors** angezeigt werden soll. Die Standardeinstellung dieser Eigenschaft ist `true`.

```

1 var textAnzeigeA:TextField = new TextField();
2 var textAnzeigeB:TextField = new TextField();
3 var textAnzeigeC:TextField = new TextField();
4 var textFormat:TextFormat = new TextFormat("Arial", 80);
5
6 // Text
7 textAnzeigeA.text = "Komm her ...";
8 textAnzeigeB.text = "Drück mich ...";
9 textAnzeigeC.text = "Lass los ...";
10
11 // Hintergrund und Rahmen
12 textAnzeigeA.backgroundColor = 0xFF0000;
13 textAnzeigeB.backgroundColor = 0x00FF00;
14 textAnzeigeC.backgroundColor = 0x0000FF;
15 textAnzeigeA.background = textAnzeigeB.background = textAnzeigeC.background = true;
16 textAnzeigeA.border = textAnzeigeB.border = textAnzeigeC.border = true;
17
18 // Position und Größe
19 textAnzeigeA.x = textAnzeigeB.x = textAnzeigeC.x = 100;
20 textAnzeigeA.y = textAnzeigeB.y = textAnzeigeC.y = 50;
21 textAnzeigeA.width = textAnzeigeB.width = textAnzeigeC.width = 600;
22 textAnzeigeA.height = textAnzeigeB.height = textAnzeigeC.height = 100;
23
24 // Textformatierung
25 textFormat.align = TextFormatAlign.CENTER;
26 textFormat.bold = true;
27 textAnzeigeA.setTextFormat(textFormat);
28 textAnzeigeB.setTextFormat(textFormat);
29 textAnzeigeC.setTextFormat(textFormat);
30
31 // Button
32 var button:SimpleButton = new SimpleButton();
33 button.hitTestState = textAnzeigeA;
34 button.overState = textAnzeigeB;
35 button.downState = textAnzeigeC;
36 button.upState = textAnzeigeA;
37 addChild(button);
    
```



## Bild

Um ein Bild (JPG-, PNG- oder GIF-Dateiformat) innerhalb einer AS3-Anwendung darzustellen, benötigen wir die `Loader`-Klasse. Über die Funktion `load()` kann das Bild geladen werden. Als Parameter wird der Pfad zu der Datei mittels eines `URLRequest`-Objekts (siehe Beispiel) angegeben. Im Beispiel wird ein Event registriert (Zeile 11), welches auslöst, wenn das Bild erfolgreich geladen wurde. In der Event-Funktion (`imageLoaded()`) wird dann das Bild skaliert und erst im

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Steuerelemente](#)

Anschluss dem Anzeigeobjekt-Container hinzugefügt.

```

1 private var bildLader:Loader;
2
3 private function init(e:Event = null):void
4 {
5     removeEventListener(Event.ADDED_TO_STAGE, init);
6
7     // Bild Ladevorgang starten
8     bildLader = new Loader();
9     bildLader.x = 300;
10    bildLader.y = 200;
11    bildLader.contentLoaderInfo.addEventListener(Event.COMPLETE, imageLoaded);
12    bildLader.load(new URLRequest("Bild-Test.jpg"));
13 }
14
15 private function imageLoaded(e:Event):void
16 {
17     // Bild von 250x250px auf 200x200px skalieren
18     bildLader.width = bildLader.height = 200;
19
20     // Bild anzeigen
21     addChild(bildLader);
22 }
    
```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Grafik-API](#)

## Grafik-API

Mit der `graphics`-Eigenschaft des `Sprite`-Objekts können wir ein Objekt der `Graphics`-Klasse abrufen, welches einige **Funktionen zum Zeichnen von Formen und Pfaden** zur Verfügung stellt. Mit Hilfe der Funktion `beginFill()` können wir die **Füllfarbe** einer Form festlegen. Als Parameter wird der Funktion der Farbwert (RGB-Wert) und optional der Alphawert (0.0 für 0% Deckkraft bis 1.0 für 100% Deckkraft) übergeben. Eine Füllung kann mit der Funktion `endFill()` beendet werden. Mit Hilfe der Funktion `lineStyle()` ist es möglich, die zu zeichnende **Line** für eine Form oder Linie festzulegen. Als Parameter können der Funktion die Breite, die Farbe und der Alphawert übergeben werden. Alle Parameter sind dabei optional. Ein Aufruf der Funktion ohne Parameter setzt die LinienEinstellung zurück. Um alle gezeichneten Grafiken sowie Füll- und LinienEinstellungen zurückzusetzen, können wir die Funktion `clear()` aufrufen.

### Inhalt dieser Seite:

1. Rechtecke
2. Kreise und Ellipsen
3. Pfade
4. Shape

## Rechtecke

Um Rechtecke zu zeichnen, stehen uns die drei Funktionen `drawRect()`, `drawRoundRect()` und `drawRoundRectComplex()` zur Verfügung. Allen Funktionen müssen vier Parameter übergeben werden: X-Position, Y-Position, Breite und Höhe. Der Funktion `drawRoundRect()` muss zusätzlich ein weiterer Parameter übergeben werden, in welchem die **Abrundung der Ecken** festgelegt wird. Als 6. optionaler Parameter kann die Höhe der Eckenabrundung festgelegt werden. Wird der Parameter weggelassen, so entspricht er der Breite der abgerundeten Ecken. Der Funktion `drawRoundRectComplex()` müssen zusätzlich zu den Positions- und Größenparameter die Radien aller 4 Ecken übergeben werden. Dabei gilt folgende Reihenfolge: oben links, oben rechts, unten links, unten rechts.

```

1 // Rotes Rechteck
2 graphics.beginFill(0xFF0000);
3 graphics.drawRect(50, 50, 300, 300);
4 graphics.endFill();
5
6 // Grünes Rechteck mit grauer Linie
7 graphics.lineStyle(10, 0xEEEEEE, 0.8);
8 graphics.beginFill(0x00FF00, 0.6);
9 graphics.drawRoundRect(200, 400, 500, 150, 25, 50);
10 graphics.endFill();
11
12 // Blaues Rechteck
13 graphics.lineStyle();
14 graphics.beginFill(0x0000FF, 0.2);
15 graphics.drawRoundRectComplex(450, 100, 200, 150, 40, 10, 20, 30);
16 graphics.endFill();

```



## Kreise und Ellipsen

Einen Kreis können wir mittels der Funktion `drawCircle()` zeichnen. Der Funktion werden die X- und Y-Position sowie der Radius übergeben. Die Positionen beziehen sich dabei auf den Mittelpunkt des Kreises. Die Funktion `drawEllipse()` zeichnet eine Ellipse. Der Funktion werden die X- und Y-Position sowie die Breite und Höhe übergeben. Die Positionen beziehen sich hier, anders als bei der Funktion `drawCircle()`, auf die Ecke oben links des umliegenden Rechtecks.

```

1 graphics.beginFill(0xFF0000);
2 graphics.drawCircle(250, 250, 200);
3 graphics.endFill();
4
5 graphics.beginFill(0x0000FF);
6 graphics.drawEllipse(450, 400, 300, 150);
7 graphics.endFill();

```



## Pfade

Um komplexe Formen (dabei spricht man auch von Pfaden) zu zeichnen, können wir die Funktionen `lineTo()` und `moveTo()` benutzen. Die Funktion `lineTo()` zeichnet dabei eine Linie von dem vorherigen Punkt bis zu dem angegebenen Punkt. `moveTo()` hingegen springt lediglich von der alten zur neuen Position, ohne dabei eine Linie zu zeichnen. Beiden Funktionen wird als Parameter die X- und Y-Position übergeben. Eine Reihe von X- und Y-Positionen kann der Funktion `drawPath()` übergeben werden. Der Funktion wird ein Vektor mit Zeichen-Kommandos (Konstanten aus der `GraphicsPathCommand`-Klasse) und ein Vektor mit Positionen (X- und Y-Position jeweils nacheinander) übergeben. Die `endFill()`-Funktion zeichnet übrigens automatisch eine Linie zum Ausgangspunkt (siehe Beispiel).

Kurven können mittels den Funktionen `curveTo()` und `cubicCurveTo()` gezeichnet werden. Die Funktion `curveTo()` zeichnet eine **quadratische Bézierkurve**. Der Funktion wird die X- und Y-Position des **Kontrollpunkts** und des sogenannten **Ankerpunkts** übergeben. Der **Ankerpunkt** stellt dabei den Punkt dar, wo die Linie / Kurve endet. Die Funktion `cubicCurveTo()` zeichnet eine sogenannte **kubische Bézierkurve**. Als Parameter werden die Positionen zweier Kontrollpunkte sowie eines Ankerpunkts übergeben.

```

1 // Achteck oben links
2 graphics.beginFill(0xFF0000);
3 graphics.moveTo(50, 25);
4 graphics.lineStyle(5, 0x00FF00);
5 graphics.lineTo(100, 25);
6 graphics.lineStyle(5, 0x0000FF);
7 graphics.lineTo(125, 50);
8 graphics.lineStyle(5, 0x00FF00);
9 graphics.lineTo(125, 75);

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Grafik-API](#)

```

10 graphics.lineStyle(5, 0x0000FF);
11 graphics.lineTo(100, 100);
12 graphics.lineStyle(5, 0x00FF00);
13 graphics.lineTo(50, 100);
14 graphics.lineStyle(5, 0x0000FF);
15 graphics.lineTo(25, 75);
16 graphics.lineStyle(5, 0x00FF00);
17 graphics.lineTo(25, 50);
18 graphics.lineStyle(5, 0x0000FF);
19 //graphics.lineTo(50, 25);    --> nicht notwendig, wird von endFill() automatisch ausgeführt
20 graphics.endFill();
21
22 // Achteck oben rechts
23 var achteckKommandos:Vector.<int> = new Vector.<int>(8, true);
24 var achteckPositionen:Vector.<Number> = new Vector.<Number>(16, true);
25 achteckKommandos[0] = GraphicsPathCommand.MOVE_TO;
26 for (var i:int = 1; i < 8; i++)
27     achteckKommandos[i] = GraphicsPathCommand.LINE_TO;
28 achteckPositionen[0] = 250;    // X1
29 achteckPositionen[1] = 25;    // Y1
30 achteckPositionen[2] = 300;    // X2
31 achteckPositionen[3] = 25;    // Y2
32 achteckPositionen[4] = 325;    // X3
33 achteckPositionen[5] = 50;    // Y3
34 achteckPositionen[6] = 325;    // X4
35 achteckPositionen[7] = 75;    // Y4
36 achteckPositionen[8] = 300;    // X5
37 achteckPositionen[9] = 100;    // Y5
38 achteckPositionen[10] = 250;    // X6
39 achteckPositionen[11] = 100;    // Y6
40 achteckPositionen[12] = 225;    // X7
41 achteckPositionen[13] = 75;    // Y7
42 achteckPositionen[14] = 225;    // X8
43 achteckPositionen[15] = 50;    // Y8
44 graphics.lineStyle();
45 graphics.beginFill(0x0000FF);
46 graphics.drawPath(achteckKommandos, achteckPositionen);
47 graphics.endFill();
48
49 // Wellenlinie mitte mitte
50 graphics.lineStyle(1, 0x00FF00);
51 graphics.moveTo(100, 300);
52 graphics.curveTo(125, 325, 150, 300);
53 graphics.curveTo(175, 275, 200, 300);
54 graphics.curveTo(225, 325, 250, 300);
55 graphics.curveTo(275, 275, 300, 300);
56 graphics.curveTo(325, 325, 350, 300);
57 graphics.curveTo(375, 275, 400, 300);
58 graphics.curveTo(425, 325, 450, 300);
59 graphics.curveTo(475, 275, 500, 300);
60 graphics.curveTo(525, 325, 550, 300);
61 graphics.curveTo(575, 275, 600, 300);
62
63 // Wellenlinie unten mitte
64 graphics.lineStyle(1, 0x00FF00);
65 graphics.moveTo(100, 500);
66 graphics.cubicCurveTo(110, 525, 130, 520, 150, 500);
67 graphics.cubicCurveTo(170, 475, 190, 480, 200, 500);
68 graphics.cubicCurveTo(210, 525, 230, 530, 250, 500);
69 graphics.cubicCurveTo(270, 475, 290, 470, 300, 500);
70 graphics.cubicCurveTo(310, 525, 330, 520, 350, 500);
71 graphics.cubicCurveTo(370, 475, 390, 480, 400, 500);
72 graphics.cubicCurveTo(410, 525, 430, 530, 450, 500);
73 graphics.cubicCurveTo(470, 475, 490, 470, 500, 500);
74 graphics.cubicCurveTo(510, 525, 530, 520, 550, 500);
75 graphics.cubicCurveTo(570, 475, 590, 480, 600, 500);

```



## Shape

Die Klasse `Shape` stellt ein einfaches Element zum Zeichnen dar. Die `Shape`-Klasse enthält, wie die `Sprite`-Klasse auch, die `graphics`-Eigenschaft. Dort können alle bekannte Funktionen verwendet werden. Der Vorteil beim Zeichnen auf ein `Shape`-Element an Stelle direkt auf das `Sprite`-Element zu zeichnen ist, dass wir mittels dieses Elements **Zeichnungen gruppieren** können. Wollen wir dann z. B. nur eine bestimmte Gruppe von Zeichnungen zurücksetzen, so rufen wir lediglich die `clear()`-Funktion dieses `Shape`-Elements auf. Alle anderen Zeichnungen, die einem anderen `Shape`-Element zugewiesen sind, bleiben dabei erhalten.

```
1 | var form:Shape = new Shape();
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

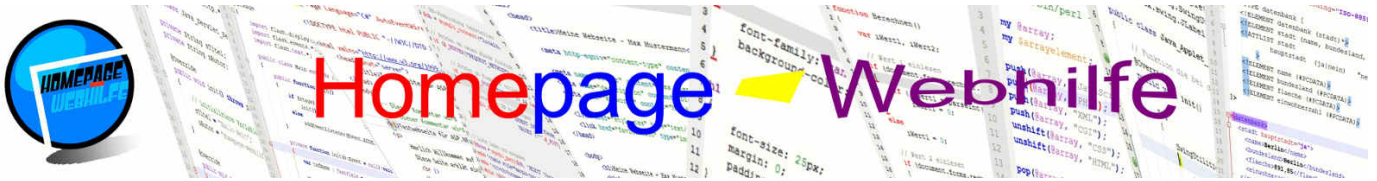
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Grafik-API](#)

```
2 form.graphics.beginFill(0xFF0000);
3 form.graphics.drawRect(50, 50, 300, 300);
4 form.graphics.endFill();
5 addChild(form);
```



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislödingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Event-Handling](#)

## Event-Handling

### Inhalt dieser Seite:

1. Maus
2. Tastatur

Unter Event-Handling versteht man das **Reagieren auf verschiedene Ereignisse**. Bei einem Ereignis kann es sich z. B. um einen Mausklick, eine Mausbewegung, einen Tastendruck oder etwas anderes handeln. Registriert werden diese Events mittels der Funktion `addEventListener()` des jeweiligen Objekts. Vielleicht erinnern Sie sich noch an folgende Zeile, welche sich im Konstruktor der Main-Klasse befindet:

```
1 | addEventListener(Event.ADDED_TO_STAGE, init);
```

Dabei handelt es sich um die sogenannte **Registrierung** des Events. Der Funktion `addEventListener()` wird eine Zeichenkette übergeben, welche den Namen des Ereignisses trägt. Dabei wird zumeist nicht die Zeichenkette direkt angegeben, sondern lediglich eine statische Konstante verwendet. Der zweite Parameter gibt eine Referenz zu einer Funktion an. Bei einer **Funktionsreferenz** wird lediglich der Name der Funktion (ohne runde Klammern) angegeben. Aus diesem Grund haben die Event-Funktionen (auch als **Event-Handler** bezeichnet) und auch andere Funktionen, die mit einer Referenz angegeben bzw. übergeben werden, immer einen festgelegten Aufbau an Parametern und dem Rückgabewert. Alle Event-Funktionen haben keinen Rückgabewert und besitzen einen Parameter, der sich je nach Event vom Typ unterscheidet. Bei allen Event-Funktionen handelt es sich jedoch um ein Objekt der Basisklasse `Event`. In dem Objekt der Klasse `Event` lässt sich über die Eigenschaft `target` das Element, welches das Ereignis ausgelöst hat, abfragen.

Um ein Event zur Laufzeit zu **deregistrieren**, können wir die Funktion `removeEventListener()` aufrufen. Als Parameter werden der Funktion die gleichen Parameter wie der Funktion `addEventListener()` übergeben. Auch einen solchen Aufruf haben wir bereits in der Event-Funktion `init()` gesehen:

```
1 | removeEventListener(Event.ADDED_TO_STAGE, init);
```

**Wichtig:** Wird z. B. für einen Button ein Klick-Event registriert, so löst das Ereignis nur aus, wenn auf den Button geklickt wurde, nicht aber, wenn außerhalb des Buttons geklickt wird.

## Maus

Die Maus bietet einige Events, welche sich generell in zwei Gruppen aufteilen lassen: Maustasten und Mausbewegung. Für die **Maustasten** gibt es die Events `CLICK`, `MOUSE_DOWN` und `MOUSE_UP`. Alle diese genannten Konstanten sind statische Konstanten der Klasse `MouseEvent`, von welcher auch ein Objekt des Event-Handlers als Parameter übergeben wird. `MOUSE_DOWN` löst aus, sobald die **Taste einer Maus gedrückt** wurde. `MOUSE_UP` löst beim **Loslassen der Maustaste** aus. Das `CLICK`-Ereignis tritt nach diesen beiden Ereignissen auf und wird am häufigsten verwendet (z. B. bei Buttons). Alle diese drei Ereignisse lösen lediglich bei Verwendung der **linken Maustaste** aus. Für die **rechte Maustaste** gibt es stattdessen die Events `RIGHT_CLICK`, `RIGHT_MOUSE_DOWN` und `RIGHT_MOUSE_UP`. Für die **mittlere Maustaste** stehen uns die Events `MIDDLE_CLICK`, `MIDDLE_MOUSE_DOWN` und `MIDDLE_MOUSE_UP` zur Verfügung.

Um die **Bewegung der Maus** zu erkennen, stehen uns die Events `MOUSE_MOVE` und `MOUSE_OUT` zur Verfügung. Das Ereignis des `MOUSE_MOVE`-Events tritt ein, wenn Sie den **Mauszeiger innerhalb eines Elements bewegen**. `MOUSE_OUT` tritt ein, wenn der **Cursor das Element verlässt**. `MOUSE_OUT` kann im unteren Beispiel nicht verwendet werden, da wir im Beispiel die Events an die `stage`-Variable knüpfen. Die `stage`-Variable verweist auf das Objekt der `Stage`-Klasse und stellt die Präsentationsoberfläche bzw. die sogenannte **Bühne** dar. Um festzustellen, ob unsere **Maus die Bühne verlassen** hat, können wir das Event `MOUSE_LEAVE` registrieren. Das `MOUSE_LEAVE`-Event gehört zu den Basis-Events (Klasse `Event`). Das `MOUSE_OUT`-Event kann also lediglich dann verwendet werden, wenn es in Kombination mit einem Element verwendet wird, welches nicht die gesamte Fläche der Bühne ausfüllt. Somit könnte das Event auch nicht an die `Main`-Klasse registriert werden, da dieses `Sprite`-Element die gesamte Bühne ausfüllt.

Bei allen Maus-Events (ausgenommen `MOUSE_LEAVE`) wird dem Event-Handler ein `MouseEvent`-Objekt als Parameter übergeben. Mittels dessen Eigenschaften `stageX` und `stageY` können wir die **Position der Maus** innerhalb der Bühne feststellen. Über die Eigenschaften `altKey`, `ctrlKey` und `shiftKey` kann der Status der Tasten ALT, CTRL / STRG und SHIFT abgefragt werden.

```
1 | private function init(e:Event = null):void
2 | {
3 |     removeEventListener(Event.ADDED_TO_STAGE, init);
4 |
5 |     stage.addEventListener(MouseEvent.MOUSE_MOVE, onMouseMove);
6 |     stage.addEventListener(MouseEvent.MOUSE_OUT, onMouseOut);
7 |     stage.addEventListener(Event.MOUSE_LEAVE, onMouseLeave);
8 |     stage.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
9 |     stage.addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
10 |    stage.addEventListener(MouseEvent.CLICK, onMouseClick);
11 | }
12 |
13 | private function onMouseMove(e:MouseEvent):void
14 | {
15 |     trace("Maus wurde bewegt (X: " + e.stageX + ", Y: " + e.stageY + ")");
16 | }
17 |
18 | private function onMouseOut(e:MouseEvent):void
19 | {
20 |     // Dieses Ereignis tritt im obigen Fall niemals ein!
21 |     trace("Maus hat Bereich verlassen");
22 | }
23 |
24 | private function onMouseLeave(e:Event):void
25 | {
26 |     trace("Maus hat Applikation verlassen");
27 | }
28 |
29 | private function onMouseDown(e:MouseEvent):void
30 | {
31 |     trace("Maustaste wurde gedrückt");
32 | }
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Event-Handling](#)

```

33
34 private function onMouseUp(e:MouseEvent):void
35 {
36     trace("Maustaste wurde losgelassen");
37 }
38
39 private function onMouseClick(e:MouseEvent):void
40 {
41     trace("Mausklick wurde ausgeführt");
42 }

```

## Ausgabe:

```

1 Maus wurde bewegt (X: 2, Y: 5)
2 Maus wurde bewegt (X: 7, Y: 7)
3 [...]
4 Maustaste wurde gedrückt
5 Maustaste wurde losgelassen
6 Mausclick wurde ausgeführt
7 Maus wurde bewegt (X: 57, Y: 22)
8 Maus wurde bewegt (X: 58, Y: 22)
9 Maus wurde bewegt (X: 60, Y: 22)
10 [...]
11 Maus wurde bewegt (X: 90, Y: 5)
12 Maus wurde bewegt (X: 92, Y: 3)
13 Maus hat Applikation verlassen

```



## Tastatur

Für die Tastatur bzw. deren Tasten gibt es zwei wichtige Ereignisse: `KEY_DOWN` und `KEY_UP`. `KEY_DOWN` tritt ein, wenn eine **Taste gedrückt wird**, `KEY_UP` hingegen, wenn diese wieder **losgelassen wird**. Wie Ihnen ja sicherlich schon aufgefallen ist, werden Zeichen doppelt in einem Dokument eingegeben, wenn Sie auf der Taste bleiben (ohne diese zwischenzeitlich loszulassen). Dies ist nicht nur beim Schreiben von Dokumenten so, sondern es handelt sich vielmehr um eine Standardfunktion vom Betriebssystem. Man spricht hier auch von der **Anschlagverzögerung**. Dies wendet sich auch auf die Ereignisse aus, so wird z. B. das `KEY_DOWN`-Event auch öfters ausgelöst, wenn Sie auf der Taste bleiben. `KEY_UP` dagegen löst lediglich einmal aus (nachdem die Taste losgelassen wird).

Bei beiden Ereignissen wird als Event-Objekt ein Objekt der Klasse `KeyboardEvent` übergeben. Die Eigenschaft `charCode` enthält den **Code des Zeichens**, welches eingegeben wurde. Die Eigenschaft `keyCode` hingegen enthält den **Code der Taste**, welche gedrückt bzw. losgelassen wurde. Mittels der Eigenschaften `altKey`, `ctrlKey` und `shiftKey` lässt sich der Status der Tasten ALT, CTRL / STRG und SHIFT abfragen.

```

1 private function init(e:Event = null):void
2 {
3     removeEventListener(Event.ADDED_TO_STAGE, init);
4
5     stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyDown);
6     stage.addEventListener(KeyboardEvent.KEY_UP, onKeyUp);
7 }
8
9 private function onKeyDown(e:KeyboardEvent):void
10 {
11     trace("onKeyDown: " + String.fromCharCode(e.charCode) + " (" + e.charCode + ") " + (e.ctrlKey ? "CTRL " : "") + (e.altKey
12 ? "ALT " : "") + (e.shiftKey ? "SHIFT " : ""));
13 }
14 private function onKeyUp(e:KeyboardEvent):void
15 {
16     trace("onKeyUp: " + String.fromCharCode(e.charCode) + " (" + e.charCode + ") " + (e.ctrlKey ? "CTRL " : "") + (e.altKey ?
17 "ALT " : "") + (e.shiftKey ? "SHIFT " : ""))
18 }

```

## Ausgabe:

```

1 onKeyDown: H (72) SHIFT
2 onKeyUp: H (72) SHIFT
3 onKeyUp: (0)
4 onKeyDown: a (97)
5 onKeyUp: a (97)
6 onKeyDown: l (108)
7 onKeyUp: l (108)
8 onKeyDown: l (108)
9 onKeyUp: l (108)
10 onKeyDown: o (111)
11 onKeyUp: o (111)
12 onKeyDown: (0) SHIFT
13 onKeyDown: ! (33) SHIFT
14 onKeyUp: ! (33) SHIFT
15 onKeyUp: (0)

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Timer](#)

## Timer

Ein Timer ermöglicht das **verzögerte und ggf. mehrmalige Ausführen einer bestimmten Funktion**. Ein Timer wird in AS3 mittels der Klasse `Timer` dargestellt (Package `flash.utils`). Dem Konstruktor wird die **Wartezeit** bzw. Dauer zwischen den Timerereignissen in Millisekunden (Eigenschaft `delay`) und die Anzahl der **Wiederholungen** (Eigenschaft `repeatCount`) übergeben. Der zweite Parameter ist dabei jedoch optional. Beim Zuweisen des Wertes `0` der Eigenschaft `repeatCount`, läuft der Timer sozusagen endlos. `0` ist dabei der Standardwert. Auf Grund einer technischen Begrenzung kann der Timer jedoch trotzdem maximal ca. 24 Tage laufen (bei der schnellstmöglichen Ausführung von 1ms). Über die Eigenschaft `currentCount` kann abgerufen werden, wie oft der Timer bereits ausgeführt wurde. Mittels der `running`-Eigenschaft kann zudem abgerufen werden, ob der Timer aktuell läuft oder nicht. Um einen **Timer zu starten**, rufen wir lediglich die Funktion `start()` auf. Ein Aufruf der `stop()`-Funktion **hält den Timer an**. Wollen wir den Timer anhalten und dabei zusätzlich den Zähler für die Ausführungen zurücksetzen, so können wir die Funktion `reset()` aufrufen. Der Timer verfügt über zwei Events, welche mittels der bekannten Funktion `addEventListener()` registriert werden können: `TIMER` und `TIMER_COMPLETE`. Das `TIMER`-Event tritt nach jedem **Ablauf der angegebenen Zeit** auf. Das Ereignis des `TIMER_COMPLETE`-Events tritt auf, wenn der **Timer „abgelaufen“** ist. Dies ist der Fall, wenn der Timer so oft wie angegeben ausgeführt wurde.

```

1 private function init(e:Event = null):void
2 {
3     removeEventListener(Event.ADDED_TO_STAGE, init);
4
5     var zaehler:Timer = new Timer(50, 25);
6     zaehler.addEventListener(TimerEvent.TIMER, onTick);
7     zaehler.addEventListener(TimerEvent.TIMER_COMPLETE, onTimerComplete);
8     zaehler.start();
9 }
10
11 private function onTick(e:TimerEvent):void
12 {
13     graphics.beginFill(0xFF0000);
14     graphics.drawRect(20 * e.target.currentCount, 20 * e.target.currentCount, 80, 80);
15     graphics.endFill();
16 }
17
18 private function onTimerComplete(e:TimerEvent):void
19 {
20     graphics.beginFill(0x0000FF, 0.25);
21     graphics.drawRect(20, 20, 560, 560);
22     graphics.endFill();
23 }
    
```



<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ActionScript](#) » [Abschluss](#)

## Abschluss

Das Tutorial zu der Programmiersprache ActionScript und der Einführung in das Erstellen von Flash-Anwendungen mittels des Flex SDKs und der IDE FlashDevelop ist nun auch schon zu Ende. Sie haben in diesem Kurs die **Sprachbestandteile der Programmiersprache** sowie ein paar **Steuerelemente** und **Schnittstellen zum Zeichnen und der Interaktion mit dem Benutzer** kennengelernt.

Auch in ActionScript ist es nicht möglich, alle Eigenschaften, Funktionen und Klassen die zur Verfügung sind vorzustellen. Eine sehr gute **Referenz** bietet hier Adobe selbst mit dem [ActionScript 3.0 Referenzhandbuch](#) an.

Trotzdem sollte es Ihnen mit den hier vorgestellten Klassen und dessen Funktionen und Eigenschaften möglich sein, einige **nützliche Anwendungen** sowie **kleinere Spiele** entwickeln zu können. Falls Sie noch Interesse an einer weiteren Technologie für clientseitige Anwendungen haben, könnte Sie vielleicht das Tutorial zu [Java SE \(Applet\)](#) interessieren. Falls Sie stattdessen eher an einer serverseitigen Skript- bzw. Programmiersprache Interesse haben, finden Sie auf unserer Website auch Tutorials zu [PHP](#), [Perl](#), [ASP.NET](#) und [Java EE](#).

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislöcher



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

Java EE

XML

# E-Book

## Java SE Applet





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java SE](#) » [Applet](#) » [Einführung](#)

## Einführung



Java Applets sind Java-Programme, welche im Browser ausgeführt und in eine Webseite integriert werden. Die Applikation wird dabei **über die JVM (Java Virtual Machine) ausgeführt**. Zur Kommunikation zwischen dem Browser und der Java Laufzeitumgebung (kurz JRE für Java Runtime Environment) wird ein **Plug-in** benötigt.

Java Applets sind vollwertige Java-Anwendungen, in welchen alle Packages und Klassen von Java SE (Java Platform, Standard Edition) verfügbar sind. So stehen zur Programmierung der Benutzeroberfläche auch die **AWT- und Swing-Komponenten** von Java zur Verfügung.

Java Applets müssen, wie andere Java Programme auch, kompiliert werden. Die Verbreitung der Applets erfolgt mittels einer Java Archiv-Datei (JAR für Java Archive). Das Programm kann somit problemlos aus mehreren Klassen bestehen, da diese lediglich in die Archiv-Datei gepackt werden.

Der Einsatz von Applets findet sich hauptsächlich in firmeninternen Netzwerken. Auf Grund der **Sicherheitsproblematik** und der immer kleiner werdenden **Browserunterstützung** sollte von Applets, falls möglich, jedoch abgesehen werden. Oracle hat angekündigt, **die Applet-Klassen sowie das Browser Plug-in in zukünftigen Versionen zu entfernen**. Sollten Sie also nicht an die Verwendung von Applets gebunden sein, ist es durchaus sinnvoller, auf neue modernere Technologien zu setzen. Dazu zählt vor allem die Kombination von HTML5, CSS3 und JavaScript.

Zum Verständnis dieses Tutorials sind Vorkenntnisse in Java erforderlich. Für einen schnellen Wiedereinstieg können Sie unseren [Crashkurs](#) besuchen.

**Wichtig:** Google Chrome unterstützt seit Version 45 die Plug-in-Architektur NPAPI nicht mehr. Dies hat zur Folge, dass Java Applets in Google Chrome nicht mehr angezeigt werden. Weitere Browser werden dieser Entscheidung u. U. folgen.

## Schnittstellen

Applets werden von der Basisklasse `Applet` (Package `java.applet`) abgeleitet, welche sich wiederum aus der Klasse `Panel` (Package `java.awt`) ableitet. Im Gegensatz zu anderen Java-Anwendungen verfügen Applets standardmäßig über keine `main()`-Funktion. Das Hinzufügen einer `main()`-Funktion hat keine Auswirkung auf das Applet selbst, erlaubt jedoch das direkte Ausführen der jar-Datei mittels Doppelklick. Man spricht hier auch von einem **hybriden Applet**.

In der `Applet`-Klasse gibt es folgende vier wichtige Funktionen: `init()`, `start()`, `stop()` und `destroy()`. Alle diese Funktionen müssen überschrieben werden, um dort einen bestimmten Programmcode hinterlegen zu können. Der Aufruf der Funktionen geschieht automatisch durch den Browser oder Applet Viewer. Die Funktion `init()` wird einmalig aufgerufen, sobald die **Applet-Ausführung vorbereitet** wurde. `start()` wird aufgerufen, sobald die **Ausführung des Applets gestartet** wurde. Die Funktion wird mindestens einmal, direkt nach dem Aufruf der `init()`-Funktion, ausgeführt. Die Funktion `stop()` hingegen wird aufgerufen, sobald die **Ausführung des Applets gestoppt** wird. Die Ausführung dieser Funktion erfolgt ebenfalls auf jeden Fall einmal, jedoch vor dem Aufruf der `destroy()`-Funktion. `destroy()` ist die letzte der vier Funktionen und wird **vor der Deinitialisierung des Applets** aufgerufen. `init()` und `destroy()` können zur **Allokierung und Freigabe von Ressourcen** verwendet werden. Wann `start()` und `stop()` aufgerufen wird ist browserabhängig. So wird z. B. `start()` und `stop()` nicht bei jedem Browser ausgeführt, wenn ein Tab-Wechsel erfolgt. Natürlich müssen nur die Funktionen überschrieben werden, welche Sie auch wirklich benötigen.

Zum **Austausch von Daten zwischen der HTML-Seite und dem Applet** können beim Initialisieren des Applets Parameter übergeben werden. Wie diese Übergabe und deren Abarbeitung aussieht, werden wir im Thema [Parameterübergabe](#) behandeln.

## Sicherheit

Java Applets selbst stellen nur ein **geringeres Sicherheitsrisiko** dar. Applets werden zwar lokal ausgeführt, jedoch erfolgt die Ausführung in einer geschlossenen Laufzeitumgebung (auch als **Sandbox** bezeichnet), wodurch ein Zugriff auf das lokale System nicht möglich ist.

Ein größeres Problem hingegen stellen die Laufzeitumgebung, die virtuelle Maschine, die Klassenbibliotheken und das Browser-Plug-in dar. **Sicherheitslücken** in diesen Programteilen erlauben bössartigen Applets einen **direkten Zugriff auf das lokale System** bzw. auf Teile des lokalen Systems. Auf Grund der immer wieder auftretenden Sicherheitslücken in der Java-Technologie haben einige Browser-Hersteller und Oracle selbst darauf reagiert.

Neben der **abnehmenden Browser-Unterstützung von Applets** (u. a. in Google Chrome) wurden auch die **Sicherheitsregeln von Java verschärft**. So werden vom Browser nur noch Applets, welche **signiert** sind, als „sicher“ markiert. Um „unsichere“ Applets auszuführen, muss neuerdings eine Ausnahmeregel im Java Control Panel hinzugefügt werden. Auch selbst-signierende Applets gelten standardmäßig nicht mehr als „sicher“.

**Wichtig:** Alle unsere Beispiele sind nicht signiert. Wenn Sie sich unsere Beispiele anschauen wollen, dann fügen Sie bitte die folgende URL als Ausnahme in Ihrem Java Control Panel hinzu: <https://ftp.homepage-webhilfe.de>.

**Übrigens:** Die Ausführung von lokalen Applets gilt ebenfalls nicht mehr als sicher. Wenn Sie also Applets im Browser testen wollen, dann müssen Sie auch das lokale Laufwerk als Ausnahme hinzufügen. Befindet sich das Applet z. B. auf dem Laufwerk D, so müssen Sie eine Ausnahme für `file:///D:/` hinzufügen.

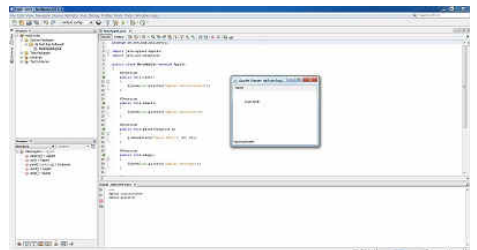
## NetBeans IDE

Zur Entwicklung von Java Applets benötigen wir eine IDE, mit welcher Java-Applikationen entwickelt werden können. Zudem benötigen wir natürlich noch das JDK (Java Development Kit). Die in diesem Kurs enthaltenen **Beispiele wurden mit Hilfe der IDE NetBeans erstellt**. Die jeweiligen Projekte können über den Download-Button heruntergeladen werden.

Um in NetBeans ein Applet zu erstellen, klicken wir auf `File` → `New Project`. Als nächstes wählen wir `Java` → `Java Class Library`. Jetzt haben wir ein leeres Projekt. Nun können wir ggf. ein Package und anschließend unsere **erste Klasse erstellen**.

Für das erste Beispiel erstellen wir eine einfache Klasse und fügen bei der Klassendeklaration die Vererbung hinzu: `extends java.applet.Applet`. Da unser Projekt lediglich eine Library ist und über keine `main()`-Funktion bzw. Hauptklasse verfügt, können wir das Projekt nicht direkt starten. Mit einem Rechtsklick auf die Datei erscheint ein Kontextmenü, in welchem wir `Run File (Umschalt+F6)` oder `Debug File (Strg+Umschalt+F5)` auswählen können. Zur Anzeige von Applets in der NetBeans IDE wird der sogenannte **Applet Viewer** verwendet.

Um aus dem Projekt eine jar-Datei erstellen zu lassen, wählen wir `Build Project (F11)` oder `Clean and Build Project (Umschalt+F11)`.



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java SE](#) » [Applet](#) » [Einführung](#)

## Erstes Applet

In unserem ersten Beispiel erstellen wir ein leeres Projekt (wie oben beschrieben) mit einer Klasse, bei welcher wir die Vererbung zu `java.applet.Applet` festlegen. Nun überschreiben wir die vier oben beschriebenen Funktionen `init()`, `start()`, `stop()` und `destroy()`. Des Weiteren überschreiben wir die `paint()`-Funktion, um einen **Begrüßungstext** auf das Panel zu zeichnen.

Um die Ausgaben (Funktion `System.out.println()`) zu sehen, empfiehlt sich die Ausführung des Projekts in einer IDE, wie z. B. NetBeans. Alternativ ist es auch möglich, im Java Control Panel unter der Registerkarte **Erweitert**, die Einstellungen **Tracing aktivieren** und **Debugging aktivieren** zu aktivieren. Des Weiteren sollte die Option **Konsole anzeigen** ausgewählt werden.

```

1 package de.hwh.bsp.hallowelt;
2
3 import java.applet.Applet;
4 import java.awt.Graphics;
5
6 public class MeinApplet extends Applet
7 {
8     @Override
9     public void init()
10    {
11        System.out.println("Applet initialisiert");
12    }
13
14    @Override
15    public void start()
16    {
17        System.out.println("Applet gestartet");
18    }
19
20    @Override
21    public void paint(Graphics g)
22    {
23        g.drawString("Hallo Welt!", 50, 50);
24    }
25
26    @Override
27    public void stop()
28    {
29        System.out.println("Applet gestoppt");
30    }
31
32    @Override
33    public void destroy()
34    {
35        System.out.println("Applet deinitialisiert");
36    }
37 }

```



## HTML-Einbindung

Die Einbindung in HTML erweist sich als etwas schwerer, wie es z. B. bei Flash-Anwendungen der Fall ist. Grund dafür ist, die mangelnde Browserunterstützung sowie die nicht klare Definition zum Einbinden von Applets seitens des W3Cs.

Die beste Lösung zum Einbinden von Applets ist daher die Verwendung des **Java Deployment Toolkits** (kurz Java DT). Bei diesem Toolkit handelt es sich um eine JavaScript-Library, welche das Einbinden von Applets je nach Browser durchführt und uns somit die Arbeit vereinfacht. Die Library kann auf der Webseite von Java gefunden werden und direkt von dort eingebunden werden. Die Einbindung erfolgt ganz normal mittels `script`-Element.

```
1 <script type="text/javascript" src="http://java.com/js/deployJava.js"></script>
```

Für Webseiten, welche verschlüsselt (HTTPS) sind, kann das Skript auch über HTTPS geladen werden.

```
1 <script type="text/javascript" src="https://java.com/js/deployJava.js"></script>
```

**Übrigens:** Eine für den Entwickler lesbare Version (bei welcher Leerzeichen und Kommentare nicht entfernt wurden) kann unter <https://www.java.com/js/deployJava.txt> bezogen werden.

Über die Funktion `runApplet()` des Objekts `deployJava` ist es möglich, ein Applet anzuzeigen. Der Funktion werden dabei zwei Objekte sowie eine Zeichenkette für die Minimalversion von Java mitgegeben. Im ersten Objekt werden die **Einstellungen für das Applet**, wie z. B. die URL zur Archiv-Datei, die URL zur Codebasis, der Pfad zur Hauptklasse und die Größe übergeben. Dem zweiten Parameter der `runApplet()`-Funktion wird ein Objekt für die **Applet-Parameter** übergeben. Darauf gehen wir im [nächsten Thema](#) nochmals genauer ein. Sind keine Parameter vorhanden, so können wir hier `null` übergeben. Die übergebene Minimalversion entspricht der Version des JDKs, mit welchem das Archiv erstellt wurde.

Der folgende JavaScript-Code zeigt, wie das Beispiel „Hallo Welt“ von oben in die HTML-Seite eingebunden werden kann:

```

1 deployJava.runApplet(
2 {
3     codebase:    "./",
4     archive:     "Hallo-Welt.jar",
5     code:        "de/hwh/bsp/hallowelt/MeinApplet.class",
6     width:      "150",

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

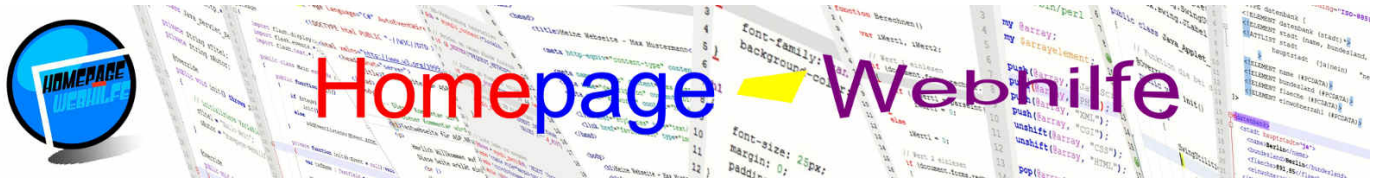
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java SE](#) » [Applet](#) » [Einführung](#)

```
7 | height: "100"  
8 | }, null, "1.8");
```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java SE](#) » [Applet](#) » [Parameterübergabe](#)

## Parameterübergabe

Wie bereits schon im vorherigen Thema angesprochen, ist es möglich, dem Applet **Parameter vor dem Start zu übergeben**. Diese Parameter können zur Laufzeit des Java-Programms mittels der Funktion `getParameter()` abgerufen werden. Die Funktion `getParameter()` erwartet als Übergabewert den **Namen des Parameters**. Als Wert gibt die Funktion eine Zeichenkette zurück, welche den **Wert des Parameters** widerspiegelt. Wurde der angefragte Parameter nicht übergeben, so gibt die Funktion `null` zurück.

In unserem JavaScript-Code müssen wir nun im zweiten Parameter der `runApplet()`-Funktion ein Objekt übergeben. Das Objekt kann nun einige Eigenschaften und Werte besitzen. Als Datentyp sollten vorzugsweise **nur Zeichenketten** verwendet werden. Zahlen oder Wahrheitswerte werden automatisch in Zeichenketten umgewandelt. Arrays werden ebenfalls in eine Zeichenkette (Elemente mittels Komma getrennt) umgewandelt. Die Übergabe von Objekten ist jedoch keinesfalls möglich.

```

1 package de.hwh.bsp.parameter;
2
3 import java.applet.Applet;
4 import java.awt.Graphics;
5
6 public class MeinApplet extends Applet
7 {
8     @Override
9     public void paint(Graphics g)
10    {
11        g.drawString("Vorname:", 50, 50);
12        if (getParameter("Vorname") != null)
13            g.drawString(getParameter("Vorname"), 150, 50);
14        else
15            g.drawString("k. A.", 150, 50);
16
17        g.drawString("Nachname:", 50, 100);
18        if (getParameter("Nachname") != null)
19            g.drawString(getParameter("Nachname"), 150, 100);
20        else
21            g.drawString("k. A.", 150, 100);
22    }
23 }
24
25 deployJava.runApplet(
26 {
27     codebase:    "./",
28     archive:     "Parameter.jar",
29     code:        "de/hwh/bsp/parameter/MeinApplet.class",
30     width:       "250",
31     height:      "150"
32 },
33 {
34     Vorname:     "Peter",
35     Nachname:    "Meyer"
36 }, "1.8");

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java SE](#) » [Applet](#) » [Applets mit AWT](#)

## Applets mit AWT

Applets können ohne weiteres AWT-Komponenten (Abstract Window Toolkit), wie z. B. die Steuerelemente `Label`, `TextField`, `Checkbox` und `Button` enthalten. Die Basisklasse bleibt dabei die Gleiche.

NetBeans ermöglicht das **Designen der AWT-Oberfläche** mittels dem integrierten Designer. Beim Erstellen einer solchen Oberfläche, sollten Sie bei der Dateierstellung `Applet Form` aus der Kategorie `AWT GUI Forms` wählen.

Das folgende Beispiel zeigt ein Textfeld zum Eintragen einer E-Mail-Adresse, ein Kontrollkästchen zum Akzeptieren der „Teilnahmebedingung“ sowie einen Button. Der Button wird erst freigeschaltet, sobald die Felder ordnungsgemäß ausgefüllt wurden. Beim Klick auf den Button wird die eingetragene E-Mail-Adresse in einem darunterliegenden Feld (`Label`) angezeigt und die Eingaben zurückgesetzt.

```

1  package de.hwh.bsp.awtapp;
2
3  public class MeinApplet extends java.applet.Applet
4  {
5      @Override
6      public void init()
7      {
8          try
9          {
10             java.awt.EventQueue.invokeLater(new Runnable()
11             {
12                 @Override
13                 public void run()
14                 {
15                     initComponents();
16                 }
17             });
18         }
19         catch (Exception ex)
20         {
21             ex.printStackTrace();
22         }
23     }
24
25     private void initComponents()
26     {
27         // [...]
28     }
29
30     private void txtEmailTextValueChanged(java.awt.event.TextEvent evt)
31     {
32         btnAnmelden.setEnabled(!txtEmail.getText().isEmpty() && cbTeilnahme.getState());
33     }
34
35     private void cbTeilnahmeItemStateChanged(java.awt.event.ItemEvent evt)
36     {
37         btnAnmelden.setEnabled(!txtEmail.getText().isEmpty() && cbTeilnahme.getState());
38     }
39
40     private void btnAnmeldenMouseClicked(java.awt.event.MouseEvent evt)
41     {
42         lblErgebnis.setText("E-Mail-Adresse " + txtEmail.getText() + " angemeldet!");
43
44         txtEmail.setText("");
45         cbTeilnahme.setState(false);
46     }
47
48     // Variables declaration - do not modify
49     private java.awt.Button btnAnmelden;
50     private java.awt.Checkbox cbTeilnahme;
51     private java.awt.Label lblEmail;
52     private java.awt.Label lblErgebnis;
53     private java.awt.Label lblTeilnahme;
54     private java.awt.TextField txtEmail;
55     // End of variables declaration
56 }

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java SE](#) » [Applet](#) » [Applets mit Swing](#)

## Applets mit Swing

Als Alternative zur Verwendung von AWT-Oberflächen können Applets auch **Swing-Steuerelemente** enthalten. Hierfür benötigen wir natürlich eine Swing-Oberfläche als Basis. Um Swing-Komponenten benutzen zu können, müssen wir deshalb die Basisklasse von `java.applet.Applet` in `javax.swing.JApplet` ändern. Dadurch stehen uns nun alle Komponenten und Steuerelemente von Swing zur Verfügung.

Auch für Swing-Oberflächen bietet NetBeans ein Tool (auch GUI-Builder genannt) zum **Designen der grafischen Oberfläche**. Beim Erstellen der Datei wählen wir `JApplet Form` aus der Kategorie `Swing GUI Forms`.

Das untere Beispiel zeigt ein Formular mit einigen Formular-Steuerelementen wie z. B. `JLabel`, `JTextField`, `JTextArea`, `JRadioButton`, `JCheckBox`, `JComboBox` und `JButton`. Beim Klick auf den Button wird ein Meldungsfenster angezeigt. Nach Bestätigung der Meldung wird das Formular zurückgesetzt.

```

1  package de.hwh.bsp.swingapp;
2
3  import javax.swing.JOptionPane;
4
5  public class MeinApplet extends javax.swing.JApplet
6  {
7      @Override
8      public void init()
9      {
10         /* Set the Nimbus look and feel */
11         // [...]
12
13         /* Create and display the applet */
14         try
15         {
16             java.awt.EventQueue.invokeLater(new Runnable()
17             {
18                 @Override
19                 public void run()
20                 {
21                     initComponents();
22                 }
23             });
24         }
25         catch (Exception ex)
26         {
27             ex.printStackTrace();
28         }
29     }
30
31     @SuppressWarnings("unchecked")
32     private void initComponents()
33     {
34         // [...]
35     }
36
37     private void btnAbschickenMouseClicked(java.awt.event.MouseEvent evt)
38     {
39         if (JOptionPane.showConfirmDialog(null, "Sind Ihre Eingaben korrekt?", "Eingabe prüfen!", JOptionPane.YES_NO_OPTION)
== JOptionPane.YES_OPTION)
40         {
41             rdbgAnrede.setSelected(rdbHerr.getModel(), true);
42             txtName.setText("");
43             txtEmail.setText("");
44             cmbSprache.setSelectedIndex(0);
45             taNachricht.setText("");
46             cbNews.setSelected(false);
47         }
48     }
49
50     // Variables declaration - do not modify
51     private javax.swing.JButton btnAbschicken;
52     private javax.swing.JCheckBox cbNews;
53     private javax.swing.JComboBox<String> cmbSprache;
54     private javax.swing.JPanel formPanel;
55     private javax.swing.JLabel lblAnrede;
56     private javax.swing.JLabel lblEmail;
57     private javax.swing.JLabel lblNachricht;
58     private javax.swing.JLabel lblName;
59     private javax.swing.JLabel lblNews;
60     private javax.swing.JLabel lblSprache;
61     private javax.swing.JRadioButton rdbFrau;
62     private javax.swing.JRadioButton rdbHerr;
63     private javax.swing.ButtonGroup rdbgAnrede;
64     private javax.swing.JScrollPane spNachricht;
65     private javax.swing.JTextArea taNachricht;

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java SE](#) » [Applet](#) » [Applets mit Swing](#)

```
66 | private javax.swing.JTextField txtEmail;  
67 | private javax.swing.JTextField txtName;  
68 | // End of variables declaration  
69 | }
```



**Übrigens:** Natürlich stehen in der Klasse `JApplet` weiterhin die Schnittstellenfunktionen `init()`, `start()`, `stop()` und `destroy()` sowie die Funktion `getParameter()` zur Verfügung. Grund dafür ist, dass `JApplet` eine von `Applet` abgeleitete Klasse ist.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung





Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java SE](#) » [Applet](#) » [Abschluss](#)

## Abschluss

Nun ist auch das Tutorial zu Applets zu Ende. Sie fragen sich, warum dieses Tutorial so kurz war? Die Antwort ist ganz einfach: Bei Java SE Applets handelt es sich lediglich um eine **Technologie** (so wie z. B. auch AJAX für JavaScript) und nicht um eine ganze Auszeichnungs-, Stylesheet-, Skript- oder Programmiersprache (so wie z. B. bei JavaScript oder ActionScript).

Nochmals wollen wir Sie an dieser Stelle darauf hinweisen, Applets nur mit Vorsicht zu verwenden und den produktiven öffentlichen Einsatz (außerhalb eines internen Netzwerkes) zu **vermeiden**.

Falls Sie Interesse an **anderen Technologien** wie [JavaScript](#) oder [ActionScript](#) haben, finden Sie auf unserer Website ebenfalls Tutorials. Auch für serverseitige Skript- und Programmiersprachen wie [PHP](#), [Perl](#), [ASP.NET](#) und [Java EE](#) finden Sie hier Tutorials. Über einen Besuch in einer der Kurse würden wir uns natürlich sehr freuen.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislungen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

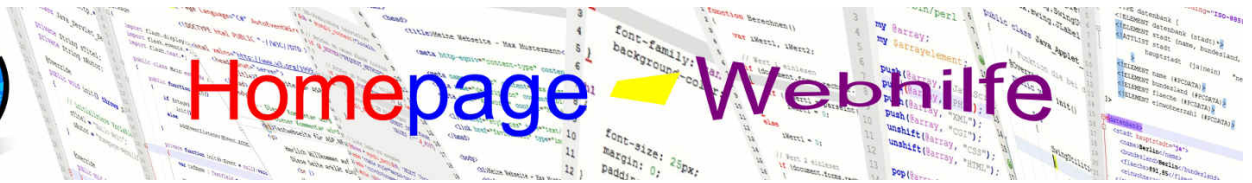
Java EE

XML

# E-Book

## PHP





Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Einführung](#)

## Einführung



Bei PHP handelt es sich wohl um die beliebteste und verbreitetste Skriptsprache für die **dynamische Erstellung von Webseiten**. Die Abkürzung PHP ist ein rekursives Akronym und steht für „**PHP: Hypertext Preprocessor**“.

PHP stellt **keineswegs einen Ersatz für HTML** dar, sondern ermöglicht es den HTML-Code, welcher an den Browser gesendet werden soll, vor dem Versand auf dem Server **mit aktuellen und dynamischen Informationen zu füllen**. Diese Informationen können dabei von **Formularen, URL-Parametern, aus Dateien oder aus einer Datenbank** kommen. Des Weiteren können die mittels Formularen versendeten Daten verarbeitet werden und dann ggf. in Dateien oder einer Datenbank abgelegt oder per E-Mail versendet werden.

PHP ist eine **relativ einfach zu erlernende Sprache**, die jedoch ebenfalls viele Möglichkeiten bietet, um somit auch professionelle und große Webanwendungen erstellen zu können. Der Syntax von PHP lehnt sich dabei an die Sprachen C und Perl an.

PHP-Code wird für gewöhnlich in Dateien mit der Endung `.php` gespeichert. In diesen Dateien ist jedoch zumeist neben PHP-Code auch HTML-Code zu finden. Der **PHP-Code wird von einem Interpreter auf dem Webserver verarbeitet**. Dadurch wird der PHP-Code **nicht an den Browser gesendet**, sondern lediglich der in der Datei ebenfalls enthaltene HTML-Code sowie die Ausgaben, welche in PHP mittels des Befehls `echo` oder `print` (dazu [weiter unten mehr](#)) ausgeführt wurden.

## Geschichte

Die **erste Version** von PHP erschien 1995 und wurde von Rasmus Lerdorf entwickelt. Die Abkürzung stand damals für „**Personal Home Page Tools**“. 1996 wurde PHP/FI veröffentlicht, welche heute als PHP-Version 2 gilt. PHP/FI (FI = Form Interpreter) war eine weitaus umfangreichere Version wie die 1. Version. Die Sprache war damals Perl sehr ähnlich.

1998 erschien eine komplett neue Version von PHP (PHP3). Die Entwicklung erfolgte durch Andi Gutmans und Zeev Suraski, die mit Rasmus Lerdorf kooperierten. Mit PHP3 wurde neben der kompletten **Neuprogrammierung des Sprachkerns** auch die Bedeutung der Abkürzung in das heute bekannte „**PHP: Hypertext Processor**“ geändert, um somit die Verwendung für lediglich private Zwecke aus dem Namen zu entfernen.

1999 gründeten Gutmans und Suraski die Firma **Zend Technologies**. Diese entwickelte unter anderem den **Zend Engine**, bei welchem es sich um den Interpreter und somit um das Herzstück des heutigen PHPs handelt. Weitere bekannte Produkte der Firma sind Zend Server und Zend Studio. Im Jahre 2000 wurde PHP4 veröffentlicht. Als Interpreter wird ab nun der Zend Engine verwendet.

Im Sommer 2004 wurde PHP5 veröffentlicht. PHP5 enthält diverse Verbesserungen und Erweiterungen wie z. B. die neue API MySQLi. Durch die Verwendung des Zend Engine 2 wurde der Sprachkern nochmals erweitert. Der Schwerpunkt liegt seither vor allem auf der **objektorientierten Programmierung**.

Seit Ende 2015 ist PHP7 verfügbar. PHP6 sollte ein früher Nachfolger von PHP5 werden. PHP6 sollte dabei eine **Unterstützung verschiedener Unicode-Standards** mit sich bringen. Die Entwicklung von PHP6 wurde auf Grund von Problemen bei der Implementierung 2010 eingestellt. PHP7 enthält einige Fehlerbehebungen, die **Unterstützung von 64-Bit-Systemen** und eine um bis zu **über 30 Prozent geringere Ausführungszeit**. Es wurden einige Änderungen durchgeführt, wodurch die Sprache zwar klarer definiert ist und keine unnötigen alten Bestandteile von Vorgängerversionen mit sich bringt, jedoch eine volle Abwärtskompatibilität zu PHP5 nicht mehr besteht.

**Wichtig:** Alle in unserem Tutorial vorgestellten Funktionen sowie alle Beispiele sind sowohl unter PHP5 als auch unter PHP7 ausführbar.

**Übrigens:** Der Zend Engine sowie die Betreuung und Verwaltung von PHP erfolgt heute durch die Entwicklergruppe „**The PHP Group**“, welcher unter anderem Rasmus Lerdorf, Andi Gutmans und Zeev Suraski angehören. Der Zend Engine wird weiterhin durch die Firma Zend Technologies gesponsert.

## Webserver

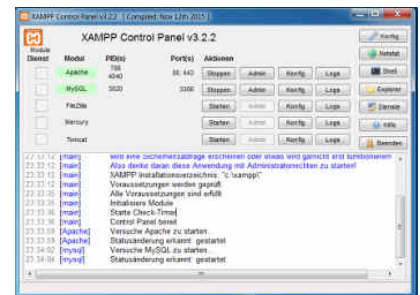
Für die Ausführung von PHP-Skripten ist ein PHP-Interpreter, normalerweise der Zend Engine, vonnöten. Der Interpreter analysiert das Skript und führt die dort angegebenen Anweisungen aus. Dieser Interpretierungsvorgang muss bei jedem Skriptaufwurf erneut durchgeführt werden. Ab PHP 5.5 enthält die PHP-Installation den **Zend Optimizer+**, welcher den vom Interpreter erstellten Zwischencode puffert, um somit während eines zweiten Interpretierungsvorgangs Zeit zu sparen. Dadurch muss die Datei nämlich nicht mehr erneut analysiert werden, sondern lediglich der erstellte Zwischencode ausgeführt werden.

Zur Ausführung von PHP-Skripten wird üblicherweise ein Webserver verwendet. Die Webserver-Software (z. B. der freie Apache HTTP Server oder der kostenpflichtige Microsoft Internet Information Services (IIS)) kommuniziert dann über eine Schnittstelle mit dem PHP-Interpreter. Die **Anfrage an den Webserver für eine PHP-Datei** läuft wie folgt ab:

- Besucher sendet Anfrage an den Webserver.
- Webserver gibt die Datei an den PHP-Interpreter weiter. Diese Datei enthält dabei zumeist neben HTML-Code auch einen PHP-Code.
- PHP-Interpreter erzeugt eine temporäre Ausgabedatei. Dabei werden die Befehle des PHP-Codes ausgeführt. Alle Ausgaben die in PHP ausgeführt wurden, werden in den HTML-Code eingefügt.
- Die temporäre Ausgabedatei wird an den Besucher zurückgesendet. Die Datei enthält keinerlei PHP-Code mehr.

Um Webseiten (mit PHP) zu entwickeln, benötigen wir also einen Webserver. Hierbei ist es hilfreich, wenn man einen **lokalen Webserver** hat, denn zum einen geht dies wesentlich schneller, als wenn man seine Änderungen immer direkt auf einen Server hochladen muss und zum anderen sollten „unfertige“ Änderungen nicht veröffentlicht werden und somit für andere zugänglich sein.

Für die PHP-Entwicklung ist das **Software-Paket XAMPP** (X = Windows, Linux, Mac OS und Solaris, A = Apache, M = MySQL / MariaDB, P = Perl, P = PHP) zu empfehlen. XAMPP ist ein Programm von Apache Friends und kann von [deren Website](#) heruntergeladen werden. Neben den bisher genannten Komponenten enthält XAMPP FileZilla bzw. ProFTPD (FTP-Server), Mercury (E-Mail-Server), phpMyAdmin (Administration von Datenbanken), Apache Tomcat (nur bei Windows) und OpenSSL. XAMPP dient lediglich zu Entwicklerzwecken und sollte nicht auf öffentlichen Webservern eingesetzt werden. Dies kommt vor allem auf Grund von **vernachlässigten Sicherheitsmaßnahmen**. Nach dem Download und der Installation von XAMPP können wir das **Control Panel** öffnen. Dort befinden sich nun Buttons zum Starten der verschiedenen Dienste. Vorerst müssen wir lediglich den Webserver Apache starten. Die Kommunikation zwischen Apache und dem PHP-Interpreter funktioniert sofort und muss nicht konfiguriert werden. Später, wenn wir mit Datenbanken arbeiten, müssen wir dort noch den Datenbank-Server MySQL bzw. MariaDB starten. Unsere PHP-Dateien (und andere Dateien die zur Website gehören) legen wir dabei im Ordner `htdocs` im Installationsverzeichnis von XAMPP (z. B. `C:\xampp\`) ab. Um auf die Website zuzugreifen, rufen wir im Browser die Adresse `localhost` (ggf. gefolgt



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Einführung](#)

von dem Dateinamen) auf. Alternativ kann auch die IP-Adresse des Rechners oder `127.0.0.1` verwendet werden. Ein Aufruf der Website von einem anderen Computer des Netzwerks ist ebenfalls möglich.

Um ein paar Informationen über den Webserver bzw. den PHP-Interpreter und deren Komponenten zu bekommen, gibt es die sogenannte **PHP-Info**. Um diese anzuzeigen, kopieren wir folgenden Code in eine Datei (z. B. `info.php`) und legen diese im `htdocs`-Ordner ab:

```
1 <?php
2 phpinfo();
3 ?>
```

Rufen wir diese Datei in unserem Browser auf, so erscheinen mehrere lange Tabellen, in welcher alle möglichen Informationen zum Zend Engine, zur PHP-Konfiguration, zu den PHP-Variablen (dazu [später mehr](#)) und den installierten Komponenten zu finden sind. Auf Grund der **sensiblen Informationen**, die in dieser Ausgabe enthalten sind, sollten Sie diese Datei niemals auf einem öffentlichen Webserver ablegen bzw. liegen lassen.

## Platzierung

PHP-Code befindet sich üblicherweise in Dateien mit der Endung `.php`. php-Dateien enthalten entweder nur PHP-Code (z. B. für ausgelagerte Funktionen und Objekte) und werden dann von anderen PHP-Dateien inkludiert (also „eingebunden“) oder enthalten PHP-Code zusammen mit HTML-Code. PHP-Code muss sich immer innerhalb des Tags `<?php` und `?>` befinden. Je nach Einstellung von PHP ist auch die verkürzte Schreibweise `<?>` am Anfang möglich. Die PHP-Tags können innerhalb eines HTML-Codes bzw. innerhalb der Datei mehrmals verwendet werden.

Auf Grund des **MVC-Konzepts** (Model, View, Controller) sollte darauf geachtet werden, **HTML und PHP nicht zu stark zu mischen**. Genauso wie man eben HTML mit CSS und HTML mit JavaScript nicht zu stark mischen sollte. Aber was heißt das konkret? Das Modell der Seite sollte lediglich in HTML angegeben werden. Überall dort, wo nun dynamische Inhalte dargestellt werden sollen, wird nun der PHP-Code notiert. Dieser muss dafür natürlich innerhalb der PHP-Tags stehen. Ein kurzes Beispiel hierzu:

```
1 <!doctype html>
2
3 <html>
4   <head>
5     <title>Aktuelle Artikel</title>
6   </head>
7
8   <body>
9     <header>
10      
11    </header>
12
13    <nav>
14      <ul>
15        <li><a href="/">Startseite</a></li>
16        <li><a href="/Aktuelle-Artikel/"><b>Aktuelle Artikel</b></a></li>
17        <li><a href="/Beliebte-Artikel/">Beliebte Artikel</a></li>
18        <li><a href="/Kontakt/">Kontakt</a></li>
19        <li><a href="/Impressum/">Impressum</a></li>
20      </ul>
21    </nav>
22
23    <main>
24      <div id="zeit">
25        <?php // Ausgabe der aktuellen Serverzeit ?>
26      </div>
27      <p>...</p>
28      <?php
29        // Laden der aktuellen Artikel aus einer Datenbank und anzeigen
30        ?>
31    </main>
32
33    <footer>
34      Copyright 2016 by Homepage-Webhilfe - All rights reserved!
35    </footer>
36  </body>
37 </html>
```

## Variablen und Datentypen

PHP kennt 6 grundlegende Datentypen: `boolean`, `integer`, `float`, `String`, `Array` und `Object`.

Der Datentyp `boolean` stellt einen **Wahrheitswert** dar und kennt nur zwei verschiedene Werte: `true` (wahr) und `false` (unwahr). Die Groß- und Kleinschreibung spielt dabei keine Rolle.

Der Datentyp `integer` stellt eine **Ganzzahl mit Vorzeichen** dar. Der Minimal- und Maximalwert ist plattformabhängig, ein Wertebereich von  $-2^{31}$  bis  $+(2^{31})-1$  sollte jedoch zumeist gegeben sein. Auf 64-Bit Systemen ist hingegen zumeist ein Wertebereich von  $-2^{63}$  bis  $+(2^{63})-1$  verfügbar.

Mit dem Datentyp `float` ist es möglich, **Gleitkommazahlen** (Zahlen mit Nachkommastellen) darzustellen. Der verfügbare Wertebereich ist ebenfalls plattformabhängig. Zumeist stehen Gleitkommazahlen mit doppelter Genauigkeit (64-Bit) nach IEEE 754 zur Verfügung. Bei der Angabe von Gleitkommazahlen im Programmcode muss beachtet werden, dass der Punkt als Trennung zwischen Ganzzahl und Nachkommastellen verwendet wird und nicht, wie in Deutschland üblich, das Komma.

**Zeichenketten** (Datentyp `String`) stellen eine Aneinanderreihung (auch als Kette oder Reihe bezeichnet) von Zeichen dar. Die Kette kann dabei aus keinem,

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Einführung](#)

einem oder mehreren Zeichen bestehen. Angegeben werden Zeichenketten in einfachen oder doppelten Anführungszeichen. Im Normalfall werden Zeichenketten in einfachen Anführungszeichen notiert. Möchten Sie innerhalb einer solchen Zeichenkette ein einfaches Anführungszeichen in die Zeichenkette integrieren, so müssen Sie dieses Zeichen mit einem Backslash \ maskieren. Eine Zeichenkette, welche in doppelte Anführungszeichen eingeschlossen ist, erlaubt es, sogenannte **Escape-Sequenzen** auszugeben. Hierzu zählen z. B. \r für einen Wagenrücklauf (CR), \n für einen Zeilenvorschub (LN) und \t für einen Tabulator. Variablenamen können direkt innerhalb einer Zeichenkette, welche in doppelten Anführungszeichen angegeben wurden, notiert werden, um diese mit dem Rest der Zeichenkette zu verketten. Als bessere Praxis gilt jedoch die Verkettung von Zeichenketten mit Variablen.

Der Datentyp `Array` stellt eine Liste und „geordnete Map“ dar. Mit diesem „speziellen“ Datentyp wollen wir uns jedoch erst [später genauer beschäftigen](#).

Zuletzt gibt es noch, wie bereits oben erwähnt, den Datentyp `Object`. Wie man eigene Klassen definiert und daraus Objekte erzeugt, [lernen Sie ebenfalls später](#).

Mittels des Werts `null` ist es möglich, eine Variable als **Variable ohne Datentyp und Wert** zu kennzeichnen. Eingesetzt wird `null` an verschiedenen Stellen, welche Sie ebenfalls innerhalb des Tutorials und der nächsten Seiten kennenlernen werden.

PHP besitzt eine **dynamische Typisierung**, weshalb der **Datentyp einer Variablen nicht explizit angegeben** werden muss. Trotzdem existieren die jeweiligen Datentypen. Der Datentyp einer Variablen wird also automatisch, je nach zugewiesenem Wert, bestimmt. Variablen existieren ab dem Zeitpunkt, wo sie zum ersten Mal zugewiesen werden. Eine Variable kann in PHP nicht direkt deklariert werden. Die **Deklaration** (sozusagen das Erstellen der Variable) erfolgt automatisch mit der ersten Zuweisung (auch als **Variableninitialisierung** bezeichnet).

**Variablen** kennzeichnen sich durch das vorangestellte Dollarzeichen \$. Der Name einer Variablen ist frei wählbar, jedoch muss dieser innerhalb des sogenannten **Scopes** (Geltungsbereich) eindeutig sein. Wird ein Variablenname innerhalb des gleichen Scopes mehrmals verwendet, so löst dies in PHP keinen Fehler aus, sondern überschreibt lediglich den Wert der Variablen. Der Geltungsbereich einer Variablen erstreckt sich über den aktuellen Block und deren untergeordneten Blöcke (dazu [gleich mehr](#)). Eine „Ausnahme“ gilt bei globalen Variablen, auf welche wir im [Thema Funktionen](#) eingehen werden. Um einer Variablen einen Wert zuzuweisen, wird der **Zuweisungsoperator** verwendet, welcher sich durch das einfache Gleichheitszeichen = kennzeichnet. Bei Variablenamen wird zwischen Groß- und Kleinschreibung unterschieden.

```
1 <?php
2 $wahrheit = true;
3 $ganzzahl = 123;
4 $gleitkomma = 123.45;
5 $zeichenkette1 = 'Hallo';
6 $zeichenkette2 = "Hallo";
7 ?>
```

## Syntax

Bevor wir uns im nächsten Abschnitt und in den nächsten Themen der Programmierung in PHP widmen, wollen wir uns zuerst mit dem grundlegenden Syntax von PHP beschäftigen. Der Syntax stellt eine Art **Regelwerk für die Zusammenstellung von Zeichen** dar.

**Blöcke** werden dazu genutzt, einen bestimmten Programmcode zu gruppieren. Benutzt werden solche Blöcke selten alleine, sondern zumeist in Kombination mit einer Abfrage, Schleife, Funktion oder einem Objekt. Ein Block wird in PHP mittels eines geschweiften Klammerspaars angegeben.

```
1 <?php
2 {
3     // Dies ist ein Block
4 }
5 ?>
```

Neben den geschweiften Klammern kommen in PHP auch runde Klammern und eckige Klammern zum Einsatz. **Runde Klammern** werden bei Funktionen, Schleifen und Abfragen benötigt. Zudem können runde Klammern zum Gruppieren verwendet werden (z. B. bei Bedingungen oder bei Berechnungen). **Eckige Klammern** kommen bei der sogenannten Indizierung von Arrays zum Einsatz. Für alle Klammern gilt: Es gibt immer eine öffnende Klammer und eine schließende Klammer. Alle Klammern müssen in umgekehrter Reihenfolge geschlossen werden, wie diese zuvor geöffnet wurden.

Bei Variablenzuweisungen, Ausgaben, Funktionsaufrufen oder anderen Schlüsselwörtern handelt es sich um sogenannte **Anweisungen** (engl. *Statements*). Anweisungen müssen in PHP, so wie in vielen anderen Skript- oder Programmiersprachen auch, mit einem Semikolon ; abgeschlossen werden.

```
1 <?php
2 $meineVariable = 12;
3 meineFunktion();
4 ?>
```

Eine **Funktion** muss zuallererst deklariert werden und kann anschließend an verschiedenen Stellen und bei Bedarf auch mehrmals aufgerufen werden. Die Funktion enthält dabei immer einen bestimmten Programmcode, welcher nur beim Aufruf der Funktion ausgeführt wird. Einige Funktionen werden als sogenannte **native Funktionen** bezeichnet. Diese Funktionen sind bereits im PHP-Interpreter implementiert und können ohne vorherige Deklaration aufgerufen werden. Wie man eigene Funktionen deklariert, werden Sie im [Thema Funktionen](#) lernen. Doch zunächst wollen wir uns anschauen, wie man Funktionen aufruft. Hierfür notieren wir den Namen der Funktion gefolgt von einem runden Klammerspaar. Innerhalb des Klammerspaars können nun sogenannte **Parameter** (auch als Argumente oder Übergabeparameter bezeichnet) übergeben werden. Dabei kann die Funktion keine, einen oder auch mehrere Parameter haben. Auch optionale Parameter sind möglich. Getrennt werden die einzelnen Parameter mittels des Kommas. Einige Funktionen geben einen Wert zurück. Dieser Wert wird als **Rückgabewert** bezeichnet. Zur Erinnerung: Der Aufruf einer Funktion ist eine Anweisung, weshalb dieser mit einem Semikolon abgeschlossen werden muss. Den Rückgabewert der Funktion (sofern vorhanden) kann einer Variablen zugewiesen, bei einer Abfrage verwendet oder einer anderen Funktion als Parameter übergeben werden.

```
1 <?php
2 meineFunktion();
3 meineFunktionMitParameter(7, 'ABC');
4 $rueckgabewert = meineFunktionMitRueckgabe();
5 ?>
```

In PHP gibt es verschiedene und teilweise widersprüchliche Regelungen und Empfehlungen was die **Namenskonventionen**, also die Namen von Variablen, Funktionen und Klassen, angeht. Die wichtigste Regel dabei ist jedoch immer die Folgende: Wenn man sich für einen Stil entschieden hat, muss dieser im kompletten Projekt (ausgenommen Libraries von Drittanbietern) gleich bleiben. Eine gängige Variante ist die **Camel-Case-Schreibweise** (mit einem Kleinbuchstaben am Anfang). Diese Konvention gilt bei Variablen und Funktionen. Bei Klassennamen wird ebenfalls die Camel-Case-Schreibweise verwendet, jedoch mit einem

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Einführung](#)

Großbuchstaben als erstes Zeichen.

```
1 <?php
2 $absenderMail = 'info@example.org';
3 $empfaengerMail = 'info@example.com';
4 ?>
```

Kommentare können auf 3 verschiedene Arten erstellt werden, wovon grundsätzlich zwischen **einzeiligen** und **mehrzeiligen Kommentaren** unterschieden wird. Einen einzeiligen Kommentar können Sie mit zwei Schrägstrichen // oder einem Hash-Zeichen # erstellen. Dabei gilt der Kommentar von der angegebenen Position bis zum Ende der Zeile. Ein mehrzeiliger Kommentar beginnt mit der Zeichensequenz /\* und endet mit \*/. Genutzt werden Kommentare zu Dokumentations- und Testzwecken. Anders wie HTML-, CSS- oder JavaScript-Kommentare werden PHP-Kommentare nicht an den Browser gesendet.

```
1 <?php
2 // Dies ist ein einzeiliger Kommentar!
3 # Dies ist auch ein einzeiliger Kommentar!
4
5 /* Dies ist ein
6    mehrzeiliger Kommentar! */
7 ?>
```

## Ausgabe

Mit dem Befehl `echo` ist es möglich, eine Ausgabe (in das HTML-Dokument) durchzuführen. Dafür wird nach dem Befehl, gefolgt von einem Leerzeichen, der auszugebende Wert (Konstante, Variable oder Rückgabewert einer Funktion) angegeben. Dies sieht dann z. B. wie folgt aus:

```
1 <?php
2 echo 'Hallo Welt!';
3 ?>
```

Der Befehl `echo` kann auch wie eine Funktion genutzt werden. Hierbei werden dann wie oben bereits erwähnt Klammern benötigt und der auszugebende Wert als Parameter übergeben.

```
1 <?php
2 echo('Hallo Welt!');
3 ?>
```

Der Befehl `echo` kann auch mehrere Werte auf einmal ausgeben. Dabei müssen die einzelnen Werte, wie bei anderen Funktionen auch, mit Komma getrennt werden. Jedoch kann hier nur der Syntax ohne Klammern (also als Befehl) verwendet werden und nicht mit den runden Klammern, wie bei einer Funktion.

```
1 <?php
2 echo 'Hallo ', 'Welt', '!';
3 ?>
```

Der Befehl `print` ist mit dem `echo` Befehl zu vergleichen. Auch bei `print` kann eine Notation ohne Klammern oder mit Klammern (wie bei einer Funktion) erfolgen. Im Unterschied zu `echo` ist jedoch nur ein Parameter möglich. Möchten Sie mehrere Werte auf einmal ausgeben, so müssen Sie `echo` verwenden.

```
1 <?php
2 print 'Hallo Welt!';
3 print('Hallo Welt!');
4 ?>
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

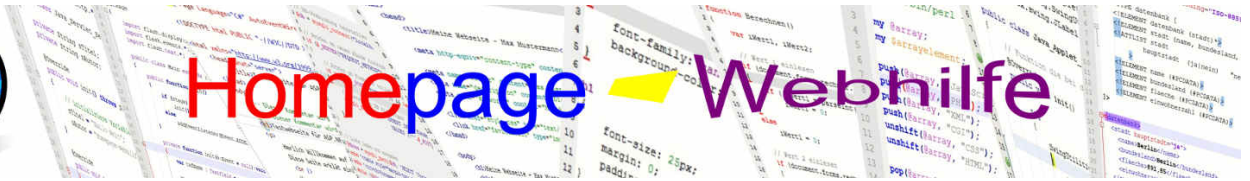
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Zahlen und Mathematik](#)

## Zahlen und Mathematik

Zahlen können in PHP direkt angegeben werden. Die Angabe kann **dezimal**, **hexadezimal**, **oktal** oder **binär** erfolgen. Um die unterschiedlichen Angaben voneinander unterscheiden zu können, muss einer hexadezimalen Zahl `0x` vorangestellt werden, einer oktalen Zahl eine `0` und einer binären Zahl `0b`. Hierzu ein kurzes Beispiel:

```
1 <?php
2 $dez = 13579;
3 $hex = 0x12AB;
4 $okt = 0135;
5 $bin = 0b1011001;
6 ?>
```

**Inhalt dieser Seite:**

- Operatoren
- Konvertierung
- Zufallszahlen
- Mathematische Funktionen

## Operatoren

Wie auch in der Mathematik, gibt es in Programmiersprachen sogenannte Operatoren. Bisher haben wir ja bereits den Zuweisungsoperator `=` kennengelernt. Nun wollen wir uns den mathematischen Operatoren widmen. Hier stehen uns `+` für die **Addition**, `-` für die **Subtraktion**, `*` für die **Multiplikation** und `/` für die **Division** zur Verfügung. Des Weiteren gibt es den in der Mathematik eher selten verwendeten Operator `%`, welcher den sogenannten **Modulo** darstellt. Modulo berechnet den Rest aus einer Division. So ergibt z. B. `10 % 3` den Wert `1`. Das folgende Beispiel zeigt die Berechnung des Umfangs und des Flächeninhalts von einem Rechteck.

```
1 <?php
2 $a = 8;
3 $b = 3;
4
5 $umfang = (2 * $a) + (2 * $b);
6 echo 'Umfang: ';
7 echo $umfang;
8 echo 'cm <br />';
9
10 $flaeche = $a * $b;
11 echo 'Fläche: ';
12 echo $flaeche;
13 echo 'cm²';
14 ?>
```

**Übrigens:** Natürlich könnten wir im obigen Beispiel (und auch in den nächsten Beispielen) unsere Ausgaben mit einem `echo`-Befehl und der Komma-Notation durchführen. Wir haben uns für diese Notation entschieden, da diese für Sie am Anfang vermutlich etwas leichter zu lesen ist.

Die Operatoren bzw. eine Variablenzuweisung mit mathematischer Berechnung können **verkürzt** werden. Dies ist jedoch nur möglich, wenn der Wert der Variablen zugewiesen wird, welcher auch als 1. Wert in der Berechnung verwendet wird. Hier ein Beispiel: `a = a + b` kann mit `a += b` verkürzt werden und `a = a * b` kann mit `a *= b` verkürzt werden. Für das Addieren oder Subtrahieren der Zahl 1 zu bzw. von einer Variablen (z. B. zum Zählen) gibt es eine noch kürzere Schreibweise: `a++` und `a--`. Bei diesen Operatoren spricht man auch von der **Inkrementierung** (Addition um 1) und **Dekrementierung** (Subtraktion um 1). Des Weiteren gibt es die Schreibweise `++a` und `--a`. Der Unterschied zwischen `a++` und `++a` bzw. zwischen `a--` und `--a` ist, dass wenn der Operation einer anderen Variablen zugewiesen oder einer anderen Funktion übergeben wird, bei `a++` und `a--` der „alte Wert“ (vor der Operation) zurückgegeben wird und bei `++a` und `--a` hingegen der „neue Wert“ (nach der Operation). Die folgende Tabelle zeigt eine Übersicht über verkürzte Schreibweisen und deren Bedeutung:

Name	Formel	Formel (kurz)	Formel (schrittweise)
Addition	$x = y + z$	$x += y$ $\triangleq$ $x = x + y$	$x++$ $\triangleq$ $x += 1$ $\triangleq$ $x = x + 1$
Subtraktion	$x = y - z$	$x -= y$ $\triangleq$ $x = x - y$	$x--$ $\triangleq$ $x -= 1$ $\triangleq$ $x = x - 1$
Multiplikation	$x = y * z$	$x *= y$ $\triangleq$ $x = x * y$	-
Division	$x = y / z$	$x /= y$ $\triangleq$ $x = x / y$	-
Modulo	$x = y \% z$	$x \% = y$ $\triangleq$ $x = x \% y$	-

## Konvertierung

Um eine Zeichenkette oder eine andere Variable „sicher“ in eine Zahl umzuwandeln, gibt es die Funktionen `intval()` und `floatval()`. `intval()` konvertiert dabei den im Parameter übergebenen Wert in den Datentyp `integer`, wo hingegen `floatval()` eine Konvertierung in den Datentyp `float` durchführt. Der Funktion `intval()` kann optional ein zweiter Parameter übergeben werden, welcher die Basis darstellt. Der Standardwert ist `10` (für eine Dezimalzahl).

```
1 <?php
2 echo intval('23s');
3 echo '<br />';
4 echo intval('abc');
5 echo '<br />';
6 echo intval('74');
7
8 echo '<br /><br />';
9
10 echo floatval('3.1416');
11 echo '<br />';
12 echo floatval('Hallo');
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Zahlen und Mathematik](#)

```
13 echo '<br />';
14 echo floatval('50,1');
15 ?>
```



## Zufallszahlen

Mit der Funktion `mt_rand()` ist es möglich, **positive Zufallszahlen** (nur Ganzzahlen) zu generieren. Die Funktion kann entweder ohne Parameter aufgerufen werden oder mit einem **Minimal- und Maximalwert**. Beide Werte geben dabei die Inklusivgrenze an. Der mögliche Maximalwert, welcher auch beim Aufruf von `mt_rand()` ohne Parameter verwendet wird, kann mittels der Funktion `mt_getrandmax()` ermittelt werden. Die Funktion `mt_srand()` setzt den **Startwert** (engl. *seed*) für den Zufallsgenerator. Der Aufruf dieser Funktion ist jedoch nicht erforderlich.

```
1 <?php
2 echo mt_getrandmax();
3
4 echo '<br /><br />';
5
6 echo mt_rand();
7 echo '<br />';
8 echo mt_rand(10, 20);
9 ?>
```



**Übrigens:** Der „Zusatz“ `mt_` im Funktionsnamen steht für **Mersenne-Twister**, bei welchem es sich um den „neueren“ und „besseren“ Zufallsgenerator von PHP handelt. Der alte `libc`-Zufallsgenerator ist weiterhin über die Funktionen `rand()`, `getrandmax()` und `srand()` verfügbar. Der alte Zufallsgenerator kann laut PHP-Dokumentation seltsame und unerwartete Verhaltensweisen aufzeigen, zudem generiert der MT-Zufallsgenerator die Zahlen um ein 4-faches schneller.

## Mathematische Funktionen

PHP bietet auch einige vordefinierte mathematische Funktionen an. Hier sind zum einen die Funktion `floor()`, `ceil()` und `round()` zu erwähnen, mit welchen **Gleitkommazahlen ab- und aufgerundet** werden können. `floor()` rundet eine Zahl auf die nächstkleinere Ganzzahl ab, `ceil()` hingegen auf die nächstgrößere Ganzzahl auf. Die Funktion `round()` führt eine **kaufmännische Rundung** durch. Demnach ergibt `round(3.4)` den Wert 3 und `round(3.5)` den Wert 4. Die Funktionen `min()` und `max()` können dazu verwendet werden, den **Minimal- oder Maximalwert aus mehreren Zahlen** zu bestimmen. Dabei kann der Funktion entweder ein Array (dazu im [Thema Arrays](#) mehr) oder beliebig viele Parameter übergeben werden. Als Rückgabe geben die Funktionen den kleinsten bzw. größten gefundenen Wert zurück. Die Funktion `sqrt()` kann dazu genutzt werden, die **Quadratwurzel** (engl. *square root*) zu berechnen. Mit der Funktion `pow()` kann eine **Potenz** berechnet werden. Hierfür wird der Funktion die Basis (1. Parameter) und der Exponent (2. Parameter) übergeben.

```
1 <?php
2 echo floor(3.4);
3 echo '<br />';
4 echo ceil(3.4);
5 echo '<br />';
6 echo round(3.4);
7
8 echo '<br /><br />';
9
10 echo min(3, 6, 2, 7, 5, 9);
11 echo '<br />';
12 echo max(3, 6, 2, 7, 5, 9);
13
14 echo '<br /><br />';
15
16 echo pow(9, 2);
17 echo '<br />';
18 echo sqrt(9);
19 ?>
```



**Übrigens:** An Stelle der Funktion `pow()` kann ab der PHP Version 5.6 auch der Operator `**` benutzt werden.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Zeichenketten](#)

## Zeichenketten

### Inhalt dieser Seite:

1. Länge ermitteln
2. Suche in einer Zeichenkette
3. Teilzeichenkette extrahieren
4. Umwandlung
5. Suchen und Ersetzen

Zeichenketten sind eine **Aneinanderreihung von Zeichen**. Bei einem Zeichen kann es sich um einen Buchstaben, eine Zahl oder ein anderes Zeichen handeln. Eine Zeichenkette (engl. *string*) ist vom Datentyp `String` und kann kein, ein oder mehrere Zeichen enthalten. Notiert werden Zeichenketten innerhalb von einfachen `'` oder doppelten `"` Anführungszeichen. Für gewöhnlich sollten Sie Zeichenketten in einfachen Anführungszeichen notieren. In doppelten Anführungszeichen ist die Angabe von **Escape-Sequenzen** (wie z. B. `\r`, `\n` oder `\t`) und Variablen möglich. Vereinfacht kann man sagen, dass Werte in doppelten Anführungszeichen geparkt werden, d. h. diese werden interpretiert und verarbeitet. Anders ist dies bei einfachen Anführungszeichen. Zeichenketten in diesen stellen lediglich eine einfache Kette von Zeichen dar. Zur **Verkettung** oder einfacher gesagt zur Verbindung von zwei oder mehreren Zeichenketten wird der Punkt-Operator `.` verwendet. Dies ist sowohl mit Zeichenketten in einfachen als auch mit Zeichenketten in doppelten Anführungszeichen möglich. Des Weiteren ist auch die Verbindung einer Zeichenkette, welche in einfachen Anführungszeichen notiert wurde, mit einer Zeichenkette, welche in doppelten Anführungszeichen notiert wurde, möglich. Einige Zeichen müssen in PHP-Strings maskiert (engl. *escape*) werden. Dies gilt für das Zeichen `'`, welches zu `\'` wird, und für `"`, welches zu `\"` wird. Natürlich ist `\` nur bei Zeichenketten, welche in einfachen Anführungszeichen notiert wurden, notwendig und das Gleiche für `\n` bei Zeichenketten in doppelten Anführungszeichen. Zudem muss der Backslash `\` mit `\\` maskiert werden.

```
1 <?php
2 $name = 'Peter';
3
4 echo 'Hallo '.$name.', wie geht es dir?';
5 ?>
```

**Übrigens:** Festwerte, dabei kann es sich um eine Zeichenkette oder auch um eine Zahl handeln, welche „direkt“ angegeben wurden, werden auch als **Literale** bezeichnet.

**Übrigens:** Möchten Sie an eine Variable einen Wert anhängen, so können Sie den Operator `.=` verwenden (z. B. `$meinName .= 'Max';` und anschließend `$meinName .= ' Meyer';`).

## Länge ermitteln

Um die Länge einer Zeichenkette zu ermitteln, benötigen wir die Funktion `strlen()`. Dieser wird als Parameter die Zeichenkette übergeben. Als Rückgabe erhalten wir einen **Zahlenwert**, welcher die Länge der Zeichenkette repräsentiert.

```
1 <?php
2 echo strlen('Hallo Welt!');
3 ?>
```

## Suche in einer Zeichenkette

Um eine **Zeichenkette innerhalb einer Zeichenkette zu suchen**, gibt es die Funktionen `strpos()`, `stripos()`, `strrpos()` und `strripos()`. Alle Funktionen haben das gleiche Schema: Der erste Parameter gibt den sogenannten *haystack* an, bei welchem es sich um die Zeichenkette handelt, in welcher gesucht werden soll. Der zweite Parameter stellt das sogenannte *needle* dar, wobei es um die Zeichenkette handelt, welche innerhalb des *haystacks* gesucht werden soll. Als dritter und optionaler Parameter kann ein nullbasierter Offset für die Suche angegeben werden. Im Gegensatz zu `strpos()` führt `stripos()` die **Suche ohne Beachtung der Groß- und Kleinschreibung** durch. Der Unterschied zwischen `strrpos()` und `strrpos()` und `stripos()` und `strripos()` ist, dass bei `strrpos()` und `strripos()` die **Suche vom Ende bis zum Anfang** durchgeführt wurde. Vereinfacht gesagt ist es mit diesen zwei Funktionen möglich, das letzte Vorkommen in einer Zeichenkette zu finden. Alle vier Funktionen geben als Rückgabe die Zeichenposition zurück. Dabei entspricht das erste Zeichen der Position 0, das zweite Zeichen der Position 1 usw.. Wird die Zeichenkette nicht gefunden, so gibt die Funktion `false` zurück.

```
1 <?php
2 echo strpos('Diese Zeichenkette enthält ZEICHEN und dient zur Suche.', 'ZEICHEN');
3 echo '<br />';
4 echo stripos('Diese Zeichenkette enthält ZEICHEN und dient zur Suche.', 'ZEICHEN');
5 ?>
```

**Wichtig:** Wird der Wert `false` mit `echo` ausgegeben (z. B. über eine der Suchfunktionen), so werden Sie bei der Ausgabe nichts sehen. Dies kommt daher, dass Werte vom Typ `boolean` nicht zur Ausgabe gedacht sind. Solche Werte werden wir später bei Bedingungen und Schleifen benutzen.

## Teilzeichenkette extrahieren

Um aus einer Zeichenkette einen Teil (eine sogenannte Teilzeichenkette) zu extrahieren, gibt es die Funktion `substr()`. Als Parameter wird der Funktion die Zeichenkette, die Startposition (ebenfalls nullbasierend) und optional die Länge der Teilzeichenkette übergeben.

```
1 <?php
2 echo substr('Diese Zeichenkette enthält ZEICHEN und dient zur Suche.', 6, 29);
3 ?>
```

## Umwandlung

Die Funktion `strtolower()` wandelt alle in der Zeichenkette enthaltene **Buchstaben in Kleinbuchstaben** um. Mit Hilfe der Funktion `strtoupper()` ist es hingegen möglich, alle **Buchstaben in Großbuchstaben** umzuwandeln. Die Funktion `trim()` entfernt sogenannte *white spaces am Anfang und am Ende* der

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Zeichenketten](#)

Zeichenkette. Standardmäßig werden folgende Zeichen entfernt: (Leer) (Leerzeichen, ASCII 0x20), `\t` (Tabulator, ASCII 0x09), `\x0B` (vertikaler Tabulator, 0x0B), `\r` (Wagenrücklauf, ASCII 0x0D), `\n` (Zeilenvorschub, ASCII 0x0A) und `\0` (NULL-Byte, 0x00). Als zweiter Parameter kann eine Zeichenkette übergeben werden, welche die Maske der zu entfernenden Zeichen enthält. Dadurch ist es auch möglich, mit Hilfe dieser Funktion **andere Zeichen** am Anfang und Ende zu entfernen.

```

1 <?php
2 echo strtolower('Diese Zeichenkette enthält ZEICHEN und dient zur Suche.');
```

## Suchen und Ersetzen

Die Funktion `str_replace()` ersetzt alle Vorkommen einer Zeichenkette in einer Zeichenkette. Vereinfacht gesagt, handelt es sich um eine Suchen- und Ersetzen-Funktion, die Sie vielleicht von Textverarbeitungsprogrammen und Editoren kennen. Der Funktion werden drei Parameter übergeben: `search` (Zeichenkette, welche gesucht werden soll), `replace` (Zeichenkette, welche ersetzt werden soll) und `subject` (Zeichenkette, in welcher `search` durch `replace` entfernt werden soll). `str_replace()` ändert nicht den `String`, welcher im Parameter `subject` übergeben wurde, sondern gibt eine neue Zeichenkette zurück. Um das Suchen **unabhängig von der Groß- und Kleinschreibung** durchzuführen, können wir an Stelle der Funktion `str_replace()` die Funktion `str_ireplace()` verwenden. Für das Suchen und Ersetzen mit Hilfe von **regulären Ausdrücken** können wir die Funktion `preg_replace()` verwenden. Auf die Verwendung von regulären Ausdrücken gehen wir aber an diesem Punkt nicht weiter ein.

```

1 <?php
2 echo str_replace('ZEICHEN', 'Buchstaben', 'Diese Zeichenkette enthält ZEICHEN und dient zur Suche.');
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Arrays](#)

## Arrays

Ein Array in PHP stellt eine sogenannte „geordnete Map“ dar. Arrays stellen dabei einen speziellen Datentyp bzw. eine Datenstruktur dar, in welcher **mehrere Werte** abgelegt werden können. Zu jedem Wert (engl. *value*) gibt es einen sogenannten Schlüssel (engl. *key*). Mit diesem **Schlüssel** ist es möglich, auf die einzelnen Elemente zuzugreifen. Man spricht hier auch von einem assoziativen Array. Standardmäßig handelt es sich bei Arrays um **indizierte Arrays**. Hierbei werden als Schlüssel numerische Werte verwendet. Das erste sogenannte Element eines Arrays besitzt dabei den Schlüssel 0, das zweite Element den Schlüssel 1, das dritte Element den Schlüssel 2 usw.. PHP unterscheidet bei der Verwendung von Arrays nicht zwischen assoziativen und indizierten Arrays. Wird jedoch kein Schlüssel durch den Programmierer explizit angegeben, so werden die Schlüssel für die einzelnen Elemente automatisch vergeben. Diese Vergabe erfolgt so wie eben bei einem indiziertem Array.

Die Definierung eines indizierten Arrays erfolgt mittels des Schlüsselworts `array()`. Innerhalb der runden Klammern können nun die Werte für das Array angegeben werden. Dabei kann es sich um eine Zeichenkette, eine Nummer oder einen anderen Datentyp handeln. Mehrere Werte werden wie bei Funktionsparametern durch Komma getrennt. Natürlich ist es auch möglich, innerhalb eines Arrays Elemente von unterschiedlichen Datentypen zu speichern.

```
1 <?php
2 $meinArray = array('A', 'B', 'C');
3 ?>
```

Um auf Elemente des Arrays zuzugreifen, benötigen wir die **eckigen Klammern**. Wollen wir z. B. auf den Wert des zweiten Elements eines Arrays zugreifen, so notieren wir den Array-Namen gefolgt von der öffnenden eckigen Klammer, der Zahl 1 und der schließenden eckigen Klammer. Die Zahl 1 stellt hierbei den Schlüssel (bei indizierten Arrays) des Elements dar. Bei assoziativen Arrays wird zwischen den eckigen Klammern der vergebene Schlüssel angegeben.

```
1 <?php
2 echo $meinArray[1]; # Gibt 'B' aus (siehe oben)
3 ?>
```

Um die **Länge** eines Arrays zu bestimmen, können wir die Funktion `count()` verwenden, welcher als Parameter das Array übergeben wird.

Während der Entwicklungszeit ist es oft praktisch, wenn man sich die Elemente eines Arrays (sowohl Schlüssel als auch Werte) anschauen kann. Hierfür können wir die Funktion `print_r()` nutzen. Dieser wird als Parameter das Array übergeben. Da die Funktion einen präformierten Text ausgibt, sollten wir die Ausgabe (sofern diese auf einer HTML-Seite verwendet wird) innerhalb der `pre`-Tags von HTML platzieren.

Hier nun das erste Beispiel zum Thema Arrays:

```
1 <?php
2 $liste = array('HTML', 'CSS', 'JavaScript', 'ActionScript', 'PHP');
3
4 echo 'Erstes Element: '.$liste[0].<br />';
5 echo 'Letztes Element: '.$liste[count($liste) - 1].<br />';
6
7 echo <pre>;
8 print_r($liste);
9 echo </pre>;
10 ?>
```

**Übrigens:** Seit der PHP-Version 5.4 ist auch eine verkürzte Schreibweise für die Array-Deklaration möglich. Dies sieht dann z. B. wie folgt aus:

```
1 <?php
2 $meinArray = ['A', 'B', 'C'];
3 ?>
```

**Übrigens:** Natürlich können Sie auch ein leeres Array deklarieren und dieses später mit Werten füllen:

```
1 <?php
2 $meinArray = array();
3 ?>
```

## Array verändern

Um ein Array zur Laufzeit zu verändern, stehen uns die Funktionen `array_push()`, `array_unshift()`, `array_pop()` und `array_shift()` zur Verfügung.

Die Funktionen `array_push()` und `array_unshift()` werden dazu verwendet, dem Array ein **Element anzufügen**. Dabei fügt die Funktion `array_push()` das Element am Ende an, die Funktion `array_unshift()` hingegen am Anfang. Beiden Funktionen wird als erster Parameter das Array übergeben und als zweiter Parameter das hinzuzufügende Element. Optional können den Funktionen auch **mehrere Werte** (durch Kommas getrennt) übergeben werden. Als Rückgabewert geben beide Funktionen die **neue Länge des Arrays** zurück.

Die Funktion `array_pop()` und `array_shift()` werden dazu verwendet, **Elemente aus einem Array zu entfernen**. Dabei entfernt die Funktion `array_pop()` das letzte Element und die Funktion `array_shift()` das erste. Als Parameter wird lediglich das Array übergeben. Beide Funktionen geben den **Wert des entfernten Elements** zurück.

```
1 <?php
2 $liste = array('CSS', 'JavaScript', 'ActionScript');
3
4 array_unshift($liste, 'HTML');
5 array_push($liste, 'PHP');
6
7 echo <pre>;
8 print_r($liste);
9 echo </pre>;
```

### Inhalt dieser Seite:

1. Array verändern
2. Assoziatives Array
3. Mehrdimensionales Array
4. Suche
5. Sortieren
6. Umwandlung von Arrays und Zeichenketten

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: Homepage-Webhilfe » PHP » Arrays

10 | >>

**Wichtig:** Bei Verwendung der Funktionen `array_unshift()` und `array_shift()` werden alle **numerischen Schlüssel automatisch angepasst**, sodass die Zählung wieder bei 0 beginnt. Array-Elemente, bei welchem der Schlüssel vom Datentyp `String` ist, werden bei der automatischen Schlüsselanpassung übersprungen.

### Assoziatives Array

Bei einem assoziativen Array, wie bereits weiter oben angesprochen, können die einzelnen Elemente einen **individuellen Schlüssel** besitzen. Dieser Schlüssel kann vom Datentyp `String` oder `integer` (wie bei indizierten Arrays) sein. Bei der Deklaration eines Arrays kann nun zusätzlich zum Wert des Elements ein Schlüssel angegeben werden. Das Schema sieht hierbei wie folgt aus: Schlüssel => Wert.

```
1 <?php
2 $liste = array(
3     'a' => 12,
4     'b' => 34,
5     'c' => 56,
6     'd' => 78
7 );
8
9 echo '<pre>';
10 print_r($liste);
11 echo '</pre>';
12 ?>
```

**Übrigens:** Beim Zugriff auf die einzelnen Elemente wird ebenfalls das eckige Klammernpaar verwendet. Zwischen den Klammern wird hier nun nicht der sogenannte Index, sondern der Schlüssel notiert.

```
1 <?php
2 echo $liste['a']; # Gibt 12 aus (siehe oben)
3 ?>
```

**Wichtig:** Für assoziative Arrays sollten Sie die Funktionen `array_push()`, `array_unshift()`, `array_pop()` und `array_shift()` vermeiden. Um Elemente hinzuzufügen, verwenden Sie den Syntax `$meinArray[$meinNeuerKey] = $meinNeuerWert;`. Das Entfernen eines bestimmten Elements mittels des Schlüssels ist mit der Funktion `unset()` möglich. `unset()` ist eine allgemeine Funktion von PHP, um **Variablen zur Laufzeit zu löschen**. Um zu **prüfen, ob eine Variable existiert**, können wir die Funktion `isset()` verwenden. Beiden Funktionen wird als einziger Parameter die zu löschende oder überprüfende Variable (oder eben das zu löschende oder zu überprüfende Element eines Arrays) übergeben. Die Funktion `isset()` werden wir noch im Laufe dieses PHP-Kurses benötigen. Hier noch ein kurzes Beispiel zur Veränderung von assoziativen Arrays:

```
1 <?php
2 $liste['e'] = 90; # Fügt ein Element mit dem Schlüssel e und dem Wert 90 hinzu (siehe oben)
3 unset($liste['b']); # Löscht das Element mit dem Schlüssel b und dem Wert 34 (siehe oben)
4 ?>
```

### Mehrdimensionales Array



Ein mehrdimensionales Array besitzt, wie der Name bereits vermuten lässt, **mehrere Dimensionen**. Bisher hatten wir nur **Arrays mit einer Dimension**. Vorstellen kann man sich ein solches eindimensionales Array als **Schrank mit mehreren Schubladen**. Ein **zweidimensionales Array**, also ein Array mit zwei Dimensionen, kann man sich mit an Hand einer **Tabelle** relativ einfach vorstellen: Es gibt Zeilen (erste Dimension) und es gibt Spalten (zweite Dimension). Die Vorstellung eines **Arrays mit drei Dimensionen** kann man sich z. B. mit einem **Zauberwürfel** (Rubik's Cube) veranschaulichen. Dort gibt es (beim klassischen Zauberwürfel) drei Felder in der Breite (X-Achse), drei in der Höhe (Y-Achse) und drei in der Tiefe (Z-Achse). Programmiertechnisch sind auch **Arrays mit vier oder mehr Dimensionen** möglich. Die Vorstellung solcher Arrays ist jedoch wesentlich komplizierter und wird auch kaum gebraucht. Mehrdimensionale Arrays müssen nicht besonders deklariert werden. Der Unterschied zu „normalen“

Arrays ist lediglich, dass ein Array als Wert ein weiteres Array usw. enthält. Auch mehrdimensionale assoziative Arrays sind möglich. Das folgende Beispiel zeigt ein indiziertes Array mit drei Dimensionen.

```
1 <?php
2 $liste = array(
3     array(
4         array('Q', 'W', 'E', 'R', 'T'),
5         array('Z', 'U', 'I', 'O', 'P'),
6         array('Ü')
7     ),
8     array(
9         array('A', 'S', 'D', 'F', 'G'),
10        array('H', 'J', 'K', 'L', 'Ö'),
11        array('Ä')
12    ),
13    array(
14        array('Y', 'X', 'C', 'V', 'B'),
15        array('N', 'M')
16    )
17 )
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » PHP » Arrays

```

17 );
18
19 echo '<pre>';
20 print_r($liste);
21 echo '</pre>';
22 ?>

```



## Suche

Mit der Funktion `array_search()` ist es möglich, ein **Array nach einem Wert zu durchsuchen**. Hierfür wird der Funktion der zu suchende Wert (*needle*) und das zu durchsuchende Array (*haystack*) angegeben. Als Rückgabewert gibt `array_search()` den Schlüssel des gefundenen Elements zurück. Bleibt die Suche erfolglos, so wird `false` zurückgegeben.

```

1 <?php
2 $liste = array('HTML', 'CSS', 'JavaScript', 'ActionScript', 'PHP');
3
4 echo 'Key von "JavaScript": ' . array_search('JavaScript', $liste);
5 ?>

```



## Sortieren

Zur Sortierung von Arrays gibt es insgesamt 6 verschiedene Funktionen: `sort()`, `rsort()`, `ksort()`, `krsort()`, `asort()` und `arsort()`. Die Funktionen `sort()`, `ksort()` und `asort()` führen eine **aufsteigende Sortierung** durch, wohingegen die Funktionen `rsort()`, `krsort()` und `arsort()` eine **absteigende Sortierung** durchführen. Alle Sortierungsfunktionen verändern dabei das übergebene Array.

Die Funktionen `sort()` und `rsort()` sortieren das Array **nach den Werten**. Dabei gehen die evtl. manuell **zugewiesenen Schlüssel verloren**. Für ein einfaches indiziertes Array sind diese beide Funktionen die geläufigsten.

Die Funktionen `ksort()` und `krsort()` sortieren das Array **nach den Schlüsseln**. Benötigt werden diese Sortierungsfunktionen fast ausschließlich bei assoziativen Arrays.

Die Funktionen `asort()` und `arsort()` sind mit `sort()` und `rsort()` zu vergleichen. Der Unterschied liegt jedoch darin, dass die **Schlüsselzuordnung nicht verloren** geht. Verwendet werden diese Funktionen ebenfalls zumeist nur bei assoziativen Arrays.

Das folgende Beispiel zeigt die Verwendung und Auswirkung der Funktionen `sort()` und `rsort()`.

```

1 <?php
2 $liste = array('HTML', 'CSS', 'JavaScript', 'ActionScript', 'PHP');
3
4 sort($liste);
5
6 echo '<pre>';
7 print_r($liste);
8 echo '</pre>';
9
10 rsort($liste);
11
12 echo '<pre>';
13 print_r($liste);
14 echo '</pre>';
15 ?>

```



## Umwandlung von Arrays und Zeichenketten

Um eine Zeichenkette an einem bestimmten Zeichen oder einer bestimmten **Zeichenfolge zu trennen**, gibt es die Funktion `explode()`. Der Funktion wird als erster Parameter das **Trennzeichen** (bzw. die Trennzeichen-Zeichenfolge) und als zweiter Parameter die **zu zerteilende Zeichenkette** übergeben. Als Rückgabewert erhalten Sie von der Funktion `explode()` ein indiziertes Array, welches die Teile der Zeichenkette repräsentiert. Optional kann der Funktion `explode()` ein dritter Parameter übergeben werden, welcher die maximale Anzahl an Array-Elementen (*limit*) angibt. Wird diese Grenze überschritten, enthält das letzte Array-Element den Rest der Zeichenkette.

Für die andere Richtung, also um **ein Array in eine Zeichenkette umzuwandeln**, gibt es die Funktion `implode()`. Dieser werden als Parameter das Zeichen mit welchem die Elemente verbunden werden sollen (*glue*) und das Array übergeben. Als Rückgabewert erhalten Sie von dieser Funktion eine Zeichenkette (Datentyp `String`).

```

1 <?php
2 $liste = array('HTML', 'CSS', 'JavaScript', 'ActionScript', 'PHP');
3
4 echo '<pre>';
5 print_r($liste);
6 echo '</pre>';
7
8 $string = implode(' ', $liste);
9
10 echo $string . '<br />';
11

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Arrays](#)

```
12 | $neueListe = explode(',', $string);
13 |
14 | echo '<pre>';
15 | print_r($neueListe);
16 | echo '</pre>';
17 | ?>
```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Bedingungen](#)

## Bedingungen

### Inhalt dieser Seite:

1. Einfache Verzweigung
2. Mehrfache Verzweigung

Bedingungen und Verzweigungen werden für **Fallunterscheidungen** verwendet und dienen zur **Steuerung des Programmablaufs**. Mit Hilfe der Fallunterscheidung ist es z. B. möglich, eine andere Aktion auszuführen, wenn Variable a größer als Variable b ist, wie wenn Variable b größer oder gleich Variable a ist. Einen Programmcode für eine solche Fallunterscheidung mittels einer einfachen Verzweigung sehen Sie [weiter unten](#).

Bedingungen benötigen immer einen Wert vom Typ `boolean`. Um einen solchen Wert zu erzeugen, stehen uns verschiedene sogenannte **Vergleichsoperatoren** zur Verfügung. Eine Liste von Vergleichsoperatoren und deren Bedeutung sehen Sie in der untenstehenden Tabelle:

<code>a == b</code>	Wert a ist gleich b
<code>a === b</code>	Wert und Typ a ist gleich b
<code>a != b</code>	Wert a ist ungleich b
<code>a !== b</code>	Wert und Typ a ist ungleich b
<code>a &gt; b</code>	Wert a ist größer als b
<code>a &gt;= b</code>	Wert a ist größer als oder gleich b
<code>a &lt; b</code>	Wert a ist kleiner als b
<code>a &lt;= b</code>	Wert a ist kleiner als oder gleich b

Natürlich können Sie Vergleichsoperatoren nicht nur bei Verzweigungen nutzen, sondern auch bei einer Schleifen-Bedingung (dazu später mehr). Das Zuweisen einer Bedingung zu einer Variablen speichert den Wert vom Typ `boolean` in der Variable ab. Als Bedingung kann neben einem mit den Vergleichsoperatoren erzeugten Wert auch einfach eine Variable vom Typ `boolean` abgefragt werden. Auch das Abfragen eines Rückgabewerts einer Funktion in Bedingungen ist möglich.

Bei der Verwendung der Operatoren `>==` und `===` sowie `!=` und `!==` ist Vorsicht geboten. Wie bereits in der Tabelle erwähnt, vergleicht `==` und `!=` lediglich **den Wert** und `===` und `!==` hingegen **den Wert und den Datentyp**. So ergibt z. B. `'12' == 12` den Wert `true`. Die Bedingung `'12' === 12` ergibt hingegen `false`. Besondere Vorsicht ist z. B. hier bei der Funktion `strpos()` geboten. Stellen Sie sich vor, Sie suchen nach einem Zeichen. Dieses befindet sich nun am Anfang der Zeichenkette (Index 0). Eine Bedingung zur Prüfung, ob die Suche erfolglos geblieben ist, würde wie folgt aussehen: `strpos('abc', 'a') === false`. Würden Sie hier statt dem `===`-Operator den `==`-Operator verwenden, so würde Ihre Bedingung nicht nur zutreffen, wenn die Suche erfolglos bleibt (und der Wert `false` ist), sondern auch, wenn sich der gesuchte Wert am Anfang der Zeichenkette befindet (Rückgabewert 0).

Um **komplexere Bedingungen** zu erstellen, ist es notwendig mehrere Bedingungen zu **verknüpfen**. Hierfür dient das **logische Und** `&&` und das **logische Oder** `||`. Bei der Kombination von Und und Oder müssen (runde) Klammern zur Gruppierung genutzt werden. Alternativ zu den Zeichen `&&` und `||` können auch die Schlüsselwörter `and` und `or` verwendet werden. Für eine Bedingung mit dem logischen Operator **Exklusiv-Oder** gibt es das Schlüsselwort `xor`. Jeder Wert vom Typ `boolean` kann mit Hilfe eines Ausrufezeichens `!` (**Nicht-Operator**) „negiert“ (umgekehrt) werden. Dabei entspricht dann `!true` dem Wert `false`.

## Einfache Verzweigung

Verzweigungen dienen zur Steuerung des Programmablaufs. So ist es Ihnen möglich, in einem **bestimmten Fall** eine oder mehrere **Aktion(en) auszuführen**. Bei einer einfachen Verzweigung wird die Fallunterscheidung mit Hilfe des Schlüsselworts `if` durchgeführt. Wir notieren bei einer einfachen Verzweigung also das Schlüsselwort `if` gefolgt von einem runden Klammernpaar, in welchem wir die Bedingung angeben. Das Ganze wird nun von einem Block mit geschweiften Klammern gefolgt. Alle Anweisungen, die sich innerhalb des Blocks befinden, werden nur dann ausgeführt, wenn die angegebene Bedingung zutrifft.

```
1 <?php
2 if ($a == $b)
3 {
4     // Anweisungen wenn Bedingung zutrifft
5 }
6 ?>
```

Möchten wir zusätzlich zu dem Fall, dass die Bedingung zutrifft auch den Fall, dass die **Bedingung nicht zutrifft** behandeln, so notieren wir nach dem `if`-Block das Schlüsselwort `else` gefolgt von einem weiteren Block. Alle Anweisungen des `else`-Blocks werden dann nur ausgeführt, wenn die Bedingung nicht zutrifft.

```
1 <?php
2 if ($a == $b)
3 {
4     // Anweisungen wenn Bedingung zutrifft
5 }
6 else
7 {
8     // Anweisungen wenn Bedingung nicht zutrifft
9 }
10 ?>
```

Innerhalb des `if`- und `else`-Blocks können weitere Verzweigungen folgen. Eine solche Verschachtelung könnte z. B. so aussehen:

```
1 <?php
2 if ($a == $b)
3 {
4     // Anweisungen wenn a gleich b ist
5 }
6 else
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Bedingungen](#)

```

7  {
8      if ($a > $b)
9      {
10         // Anweisungen wenn a größer als b ist
11     }
12     else
13     {
14         // Anweisungen wenn a kleiner als b ist
15     }
16 }
17 ?>

```

Eine Verschachtelung im `else`-Block (so wie oben) lässt sich mit Hilfe des Schlüsselworts `elseif` eleganter lösen. Der obige Code könnte somit auch wie folgt dargestellt werden:

```

1  <?php
2  if ($a == $b)
3  {
4      // Anweisungen wenn a gleich b ist
5  }
6  elseif ($a > $b)
7  {
8      // Anweisungen wenn a größer als b ist
9  }
10 else
11 {
12     // Anweisungen wenn a kleiner als b ist
13 }
14 ?>

```

Befindet sich innerhalb einer der Blöcke nur eine Anweisung, so können die geschweiften Klammern einfach weggelassen werden. Dies zeigt das folgende Beispiel:

```

1  <?php
2  $a = 43;
3  $b = 29;
4
5  if ($a == $b)
6      echo '$a ist gleich $b!';
7  elseif ($a > $b)
8      echo '$a ist größer als $b!';
9  else
10     echo '$a ist kleiner als $b!';
11 ?>

```

Befindet sich sowohl im `if`- als auch im `else`-Block eine Anweisung, um einer Variablen einen Wert zuzuweisen, so erhalten Sie mit Hilfe des **ternären Operators** `?:` eine verkürzte Schreibweise. Schauen wir uns zunächst einen Konstrukt aus `if`- und `else`-Block an:

```

1  <?php
2  if ($a == $b)
3  {
4      $c = 'gleich';
5  }
6  else
7  {
8      $c = 'ungleich'
9  }
10 ?>

```

Die verkürzte Schreibweise mit dem ternären Operator sieht wie folgt aus:

```

1  <?php
2  $c = ($a == $b) ? 'gleich' : 'ungleich';
3  ?>

```

Der Syntax ist also wie folgt: `Bedingung ? WertWennWahr : WertWennUnwahr`. Bezeichnet wird ein solches Konstrukt auch als In-Line-Abfrage.

## Mehrfache Verzweigung

Möchten Sie einen Wert auf viele verschiedene Werte abprüfen, so wirken viele `if`-, `elseif`- und `else`-Blöcke schnell unübersichtlich. Oft kann hier eine mehrfache Verzweigung mittels `switch-case` Abhilfe schaffen.

Bei einer solchen mehrfachen Verzweigung notieren wir das Schlüsselwort `switch` gefolgt von einem runden Klammernpaar, in welchem der zu überprüfende Wert (z. B. eine Variable) angegeben wird. Das Ganze wird nun von einem Block aus geschweiften Klammern gefolgt:

```

1  <?php
2  switch ($a)
3  {
4      // case-Blöcke
5  }
6  ?>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

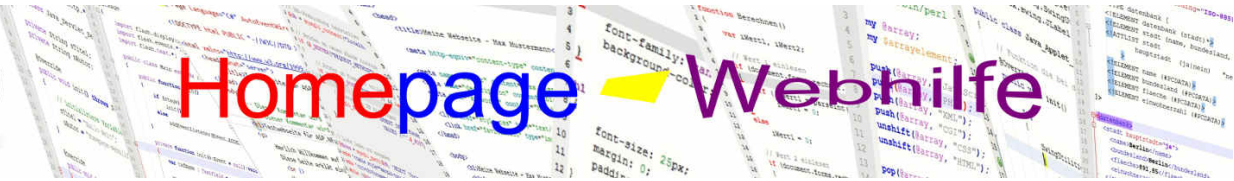
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Bedingungen](#)

Innerhalb des Blocks werden nun die **einzelnen Fälle** mit Hilfe von `case`-Blöcken geprüft. Hierfür notieren wir das Schlüsselwort `case` gefolgt von einem Wert, der mit dem oben angegebenen Wert verglichen werden soll. Zum Schluss notieren wir einen Doppelpunkt `:`. Alle nun gefolgt Anweisungen werden nur ausgeführt, wenn die Bedingung zutrifft. Um einen `case`-Block zu „beenden“, notieren wir das Schlüsselwort `break` gefolgt von einem Semikolon (es handelt sich hierbei um eine Anweisung).

```

1 <?php
2 case 1:
3     // Anweisungen wenn Wert 1 entspricht
4     break;
5 ?>
    
```

Ein `switch`-Block enthält für gewöhnlich mehrere `case`-Blöcke. Optional kann ein `switch`-Block noch einen `default`-Block enthalten. Der `default`-Block ist vom Aufbau mit dem `case`-Block fast identisch. Der Unterschied liegt darin, dass beim `default`-Block kein Wert angegeben wird. Die Anweisungen des `default`-Blocks werden nur ausgeführt, wenn **keine der anderen Bedingungen zutrifft**.

Möchten Sie z. B., wenn die zu überprüfende Variable den Wert 1 oder den Wert 2 hat, die gleichen Anweisungen ausführen, so ist es möglich, `case`-Blöcke zu stapeln. Dies sieht dann z. B. so aus:

```

1 <?php
2 case 1:
3 case 2:
4     // Anweisungen wenn Wert 1 oder 2 entspricht
5     break;
6 ?>
    
```

Hier nun ein vollständiges Beispiel zur Verwendung von mehrfachen Verzweigungen:

```

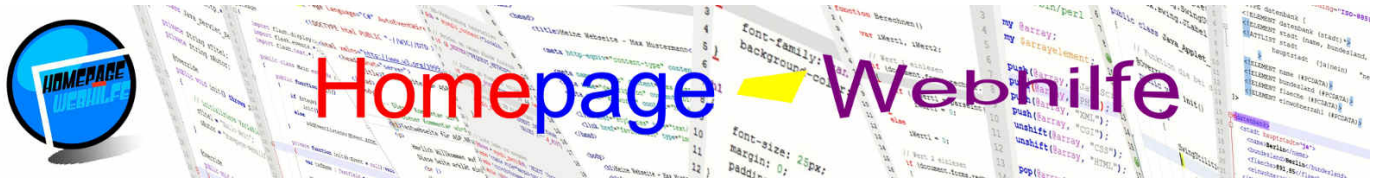
1 <?php
2 $monat = 7;
3
4 switch ($monat)
5 {
6     case 1:
7         echo 'Januar';
8         break;
9     case 2:
10        echo 'Februar';
11        break;
12    case 3:
13        echo 'März';
14        break;
15    case 4:
16        echo 'April';
17        break;
18    case 5:
19        echo 'Mai';
20        break;
21    case 6:
22        echo 'Juni';
23        break;
24    case 7:
25        echo 'Juli';
26        break;
27    case 8:
28        echo 'August';
29        break;
30    case 9:
31        echo 'September';
32        break;
33    case 10:
34        echo 'Oktober';
35        break;
36    case 11:
37        echo 'November';
38        break;
39    case 12:
40        echo 'Dezember';
41        break;
42 }
43 ?>
    
```

**Wichtig:** Mit Hilfe von `switch-switch` können keine Wertebereiche, sondern lediglich Einzelwerte überprüft werden. Für die Überprüfung von Wertebereichen müssen Sie `if-else` verwenden.

**Übrigens:** Auch wenn eine `switch-case`-Verzweigung mehr Programmzeilen als eine `if-else`-Verzweigung erzeugt, gilt die `switch-case`-Verzweigung in solchen Situationen wie das Beispiel oben als übersichtlicher.

**Übrigens:** Die in unseren Code-Ausschnitten enthaltene(n) Zeilenumbrüche, Leerzeichen und Einrückungen sind oft nicht zwingend notwendig. Wir empfehlen Ihnen

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Bedingungen](#)

jedoch, diese ebenfalls zu verwenden, da dadurch Ihr Programmcode für Sie selbst und für Andere **viel leichter zu lesen** ist. Als Einrückung wird zumeist 1 Tab bzw. 4 Leerzeichen verwendet. Die meisten Quellcode-Editoren führen die Einrückungen automatisch hinzu.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Schleifen](#)

## Schleifen

Schleifen dienen dazu, einen bestimmten **Programmcode mehrmals auszuführen**. PHP stellt uns hier insgesamt 4 verschiedene Schleifen zur Verfügung: `for`, `while`, `do-while` und `foreach`. Bei der `for`- und `while`-Schleife handelt es sich um eine **kopfgesteuerte Schleife**. Dabei wird die Bedingung geprüft bevor der Code in der Schleife ausgeführt wird. Die `do-while`-Schleife ist eine **fußgesteuerte Schleife**. Hierbei wird die Bedingung erst geprüft nachdem der Code innerhalb der Schleife ausgeführt wurde und somit bevor der Code u. U. ein weiteres Mal ausgeführt wird. Die Schleife `foreach` wird für Arrays verwendet.

Eine Schleife besteht immer aus einem **Schleifenkopf oder Schleifenfuß** und einem **Schleifenrumpf**. Der Schleifenrumpf stellt den Code dar, welcher u. U. mehrmals ausgeführt wird. Der Code wird mit Hilfe von geschweiften Klammern gruppiert. Man spricht auch hier von einem Block. Die geschweiften Klammern können, so wie bei Verzweigungen auch, weggelassen werden, wenn im Schleifenrumpf nur eine Anweisung enthalten ist. Bei der `do-while`-Schleife ist das Weglassen der Klammern grundsätzlich nicht möglich. Der Schleifenkopf bzw. Schleifenfuß enthält die **Bedingung der Schleife**, mit welcher geprüft wird, ob der Schleifenrumpf erneut ausgeführt werden soll. Eine genaue Erklärung zu den einzelnen Schleifen sowie einen Beispielcode sehen Sie weiter unten.

### Inhalt dieser Seite:

1. Zählschleife
2. Kopfgesteuerte Schleife
3. Fußgesteuerte Schleife
4. Array-Schleife

## Zählschleife

Die `for`-Schleife ist eine klassische Schleife und dient zumeist zur **n malen Ausführung** des Schleifenrumpfs. Deshalb wird diese Schleife auch gerne als Zählschleife bezeichnet. Bei der `for`-Schleife handelt es sich um eine kopfgesteuerte Schleife. Der Schleifenkopf ist hier etwas komplexer als bei anderen Schleifen, da dieser aus drei Teilen besteht. Die drei Teile werden durch Semikolon getrennt und befinden sich innerhalb eines runden Klammerspaars hinter dem Schlüsselwort `for`. Der erste Teil dient zur Variablendeklaration und -Initialisierung bzw. Wertzuweisung. Der zweite Teil stellt die eigentliche Schleifenbedingung dar. Da es sich bei der `for`-Schleife um eine kopfgesteuerte Schleife handelt, wird die Bedingung vor jeder Ausführung des Schleifenrumpfs überprüft. Der dritte Teil des Schleifenkopfs ermöglicht das Ändern einer beliebigen Variablen oder das Ausführen eines anderen Programmcodes. Dieser wird nach jedem Durchlauf des Schleifenrumpfs ausgeführt. Bei der klassischen Zählschleife wird hier eine **Zählvariable inkrementiert**.

```
1 <?php
2 for ($i = 0; $i < 10; $i++)
3     echo $i.'  
';
4 ?>
```



## Kopfgesteuerte Schleife

Die `while`-Schleife ist eine einfache kopfgesteuerte Schleife. Die `while`-Schleife kennzeichnet sich durch das Schlüsselwort `while` und durch ein rundes Klammerspaar, in welchem die Bedingung angegeben wird.

```
1 <?php
2 $i = 0;
3
4 while ($i < 10)
5 {
6     echo $i.'  
';
7
8     $i++;
9 }
10 ?>
```



**Wichtig:** Das obige Beispiel ist äquivalent zum Beispiel zur `for`-Schleife. Für „Zählvorgänge“ sollten Sie jedoch lieber die `for`-Schleife verwenden, da diese kompakter ist.

## Fußgesteuerte Schleife

Die `do-while`-Schleife ist eine einfache fußgesteuerte Schleife. Um diese anzugeben, notieren wir zuerst das Schlüsselwort `do` gefolgt von einem Block. Dahinter folgt das Schlüsselwort `while` und die Bedingung in einem runden Klammerspaar. Am Ende muss noch zusätzlich ein Semikolon angegeben werden.

```
1 <?php
2 $i = 0;
3
4 do
5 {
6     echo $i.'  
';
7
8     $i++;
9 } while ($i < 10);
10 ?>
```



**Wichtig:** Auch wenn das Beispiel zur `do-while`-Schleife die gleiche Ausgabe erzeugt, wie die der `while`-Schleife, kann die `do-while`-Schleife nicht als Ersatz zu einer `while`-Schleife angesehen werden, denn im Gegensatz zur `while`-Schleife wird der Schleifenrumpf einer `do-while`-Schleife immer ein Mal ausgeführt.

## Array-Schleife

Die Array-Schleife in PHP dient dazu, durch ein Array bzw. durch dessen Elemente durchzugehen. Um eine Array-Schleife anzugeben, notieren wir das Schlüsselwort `foreach` und anschließend ein rundes Klammerspaar. Innerhalb des Klammerspaars wird nun der Name des Arrays angegeben, das Schlüsselwort

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Schleifen](#)

`as` und ein frei definierbarer Variablenname, welcher innerhalb der Schleife dazu genutzt werden kann, um auf den **Wert des aktuellen Array-Elements** zuzugreifen.

```
1 <?php
2 $liste = array('HTML', 'CSS', 'JavaScript', 'ActionScript', 'PHP');
3
4 foreach ($liste as $listenEintrag)
5     echo $listenEintrag.'  
';
6 ?>
```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Funktionen](#)

## Funktionen

### Inhalt dieser Seite:

1. Funktion mit Parametern
2. Funktion mit Rückgabewert

Funktionen sind dazu gedacht, **Programmcode auszulagern** und um diesen später **mehrmals verwenden** zu können. Wie Funktionen aufgerufen werden, haben Sie bereits am Anfang dieses Tutorials kennengelernt. Zur Erinnerung: Sie notieren den Namen der Funktion gefolgt von einem runden Klammerspaar. Innerhalb des Klammerspaars können nun mehrere Parameter durch Komma getrennt an die Funktion übergeben werden. Gibt die Funktion einen Wert zurück, so können wir den Rückgabewert der Funktion z. B. in einer Variablen speichern oder diesen einer anderen Funktion übergeben.

In diesem Thema wollen wir uns damit beschäftigen, wie Sie **eigene Funktionen definieren** können. Der Aufruf von selbst definierten Funktionen unterscheidet sich nicht vom Aufruf der im PHP-Interpreter integrierten Funktionen. Um eine Funktion zu definieren, notieren wir das Schlüsselwort `function` gefolgt von dem **Funktionsnamen** (dieser muss innerhalb des aktuellen Objekts eindeutig sein, dazu später mehr) und einem runden Klammerspaar. Nun folgt ein Block aus geschweiften Klammern. Innerhalb des Blocks können Sie nun den Programmcode angeben, welcher ausgeführt werden soll, wenn die Funktion aufgerufen wird.

```
1 <?php
2 function gebeHalloAus()
3 {
4     echo 'Hallo Welt!';
5 }
6
7 gebeHalloAus();
8 ?>
```

**Wichtig:** Wenn Sie innerhalb der Funktion Variablen deklarieren, sind diese nur innerhalb der Funktion zugänglich. Ein Zugriff von außerhalb der Funktion ist nicht möglich, da die Variable sich in einem anderen **Scope** (Geltungsbereich) befindet. Wenn Sie außerhalb der Funktion eine Variable (auch als **globale Variable** bezeichnet) definiert haben, können Sie auf diese von der Funktion auch nur über einen kleinen „Umweg“ zugreifen. Hierfür existieren zwei Möglichkeiten: Wenn Sie z. B. auf die globale Variable `$meinName` zugreifen möchten, geben Sie innerhalb Ihrer Funktion den Variablennamen `$GLOBALS` an. Bei `$GLOBALS` handelt es sich um ein assoziatives Array, welches alle globale Variablen enthält. Ein Zugriff auf die globale Variable `$meinName` wäre also über `$GLOBALS['meinName']` innerhalb der Funktion möglich. Als zweite Möglichkeit können Sie innerhalb Ihrer Funktion (am besten am Anfang der Funktion) die Anweisung `global` gefolgt von dem Variablennamen angeben (z. B. `global $meinName;`). Globale Variablen existieren nur außerhalb von Objekten. Ist Ihre Variable Teil eines Objekts (dazu später mehr), sieht der Zugriff auf die Objektvariable nochmals anders aus. Damit beschäftigen wir uns jedoch erst im [Thema Objektorientierung](#).

## Funktion mit Parametern

Natürlich können Sie auch Funktionen mit Parametern definieren. Hierfür geben Sie innerhalb der Klammern die Namen der Parameter an. Die Parameternamen dienen zum Zugriff auf die Parameterwerte innerhalb der Funktion. Die Parameternamen müssen so wie Variablennamen auch, eindeutig sein. Des Weiteren dürfen sich die Parameternamen nicht mit den Variablennamen innerhalb der Funktion (den sogenannten **lokalen Variablen**) überschneiden. In PHP können Sie Parameter auch als optional markieren. Hierfür müssen Sie dem Parameter, welcher optional sein soll, einen **Standardwert** zuweisen. Dies können Sie mit Hilfe des Zuweisungsoperators innerhalb der Funktionsdeklaration durchführen. Das folgende Beispiel zeigt eine Funktion mit einem Parameter, welcher optional ist.

```
1 <?php
2 function gebeNamenAus($name = 'Peter')
3 {
4     echo 'Hallo '.$name.'!<br />';
5 }
6
7 gebeNamenAus('Michael');
8 gebeNamenAus();
9 gebeNamenAus('Anna');
10 ?>
```

**Wichtig:** Die optionalen Parameter müssen sich immer am Ende befinden. Stellen Sie sich vor, Sie haben eine Funktion mit den Parametern a, b, c, d und e. Die Parameter d und e sind optional. Sie können diese Funktion nun auf drei verschiedene Arten aufrufen: mit den Parametern a, b und c, mit den Parametern a, b, c und d oder mit den Parametern a, b, c, d und e. Das Auslassen eines optionalen Parameters in der Mitte ist nicht möglich. Ein Aufruf mit den Parametern a, b, c und e ist also nicht möglich.

## Funktion mit Rückgabewert

Um von einer Funktion aus einen Wert zurückzugeben, verwenden Sie das Schlüsselwort `return`, gefolgt von dem zurückzugebenden Wert und einem Semikolon am Ende. Durch das Ausführen der `return`-Anweisung wird die **Funktions-Ausführung beendet**. Bei einer Funktion, welche keinen Wert zurückgibt, können Sie die Anweisung `return;` also auch einfach nur dazu verwenden, um **aus der Funktion herauszuspringen**.

```
1 <?php
2 function addiere($a, $b)
3 {
4     return $a + $b;
5 }
6
7 echo addiere(3, 5);
8 ?>
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisllingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Objektorientierung](#)

## Objektorientierung

### Inhalt dieser Seite:

1. Konstruktor
2. Vererbung

PHP ist, was viele nicht wissen, eine objektorientierte Sprache. Dies kommt vor allem daher, da sich viele Grundlagen-Tutorials nicht mit dem Thema der Objektorientierung beschäftigen. In Anbetracht dessen, dass Sie PHP lernen wollen, um damit später vielleicht auch ein **größeres Projekt** zu erstellen, ist es nützlich, die **Möglichkeiten der Objektorientierung** zu kennen, da es damit möglich ist, Daten (Werte, Variablen und Funktionen) einfach zu **kapseln**. In PHP ist es möglich, Programmcode außerhalb von Objekten zu platzieren. Dies ist bei anderen Sprachen oftmals, wie z. B. in C++, nicht möglich.

Bevor wir unser erstes Objekt erzeugen bzw. unsere erste Klasse definieren, müssen wir einige Begriffe erklären. Unter der objektorientierten Programmierung (oder kurz **OOP**) versteht man eine bestimmte Technik, Daten (dazu zählen Konstanten sowie die sogenannten Eigenschaften und Methoden) zu kapseln, also „auszulagern“ und zu „gruppieren“.

Um ein Objekt zu erzeugen, man spricht hier auch von instanzieren, benötigen wir zuerst eine Art Bauplan. Ein **Bauplan** wird in der OOP durch eine **Klasse** repräsentiert. Um eine Klasse zu definieren, notieren wir das Schlüsselwort `class` gefolgt von einem frei definierbaren Klassennamen. Der Klassename muss, wie Variablennamen auch, eindeutig sein. Der Name kann dabei Buchstaben, Zahlen oder Unterstriche enthalten. Als gängige Schreibweise wird auch hier die sogenannte **Camel-Case-Schreibweise** verwendet, bei welcher jeder Anfangsbuchstabe eines Worts groß geschrieben wird. Im Gegensatz zu Variablen- oder Funktionsnamen, beginnen Klassennamen für gewöhnlich mit einem Großbuchstaben. Die Schreibweise ist jedoch nur eine Empfehlung und keineswegs eine Pflicht. Nach dem Klassennamen notieren wir einen Block.

```
1 <?php
2 class MeineKlasse
3 {
4
5 }
6 ?>
```

Innerhalb des Blocks können nun **Konstanten**, **Variablen** und **Funktionen** angegeben werden. Diese gehören damit zur Klasse und können nur innerhalb der Klasse oder über ein instanziiertes Objekt (dazu später mehr) aufgerufen und verwendet werden (abgesehen von zwei Ausnahmen, dazu gleich mehr). Für die Variablen (auch als **Eigenschaften** bezeichnet) und Funktionen (auch als **Methoden** bezeichnet) innerhalb einer Klasse kann bzw. sollte ein sogenannter **Zugriffsmodifizierer** angegeben werden. Hierfür stehen die Schlüsselwörter `public`, `protected` und `private` zur Verfügung. Mit Hilfe des Zugriffsmodifizierers kann die sogenannte **Sichtbarkeit** von Eigenschaften und Methoden gesteuert werden. Durch das Schlüsselwort `public` ist ein Zugriff auf die Variable oder Funktion von innerhalb der Klasse als auch von „außerhalb“ möglich. Bei `private` ist lediglich ein Zugriff von innerhalb der Klasse möglich. `protected` ist ähnlich wie `private`, jedoch mit dem Unterschied, dass auf Eigenschaften oder Methoden, die von einer anderen Klasse geerbt wurden und als `private` markiert sind, von einer anderen Klasse, der sogenannten Kindklasse, zugegriffen werden kann. Wenn Sie innerhalb von einer Klasse auf eine Eigenschaft oder Methode der Klasse zugreifen wollen, so benötigen Sie die Variable `$this`. `$this` enthält immer das aktuelle Objekt, also die Instanz der Klasse. Wollen Sie nun z. B. auf die Eigenschaft `meineVariable` zugreifen, so notieren Sie `$this->meineVariable`. Das gleiche Prinzip gilt bei Methoden. Die Zeichenfolge `->` wird als Pfeiloperator bezeichnet.

```
1 <?php
2 class MeineKlasse
3 {
4     public $MeineEigenschaft;
5
6     public meineFunktion()
7     {
8         $this->MeineEigenschaft = 'Mein Wert';
9     }
10 }
11 ?>
```

**Objekte** werden mit Hilfe des Bauplans (also der Klasse) erzeugt. Dieser Vorgang wird als **Instanziierung** bezeichnet. Das instanziierte Objekt wird auch als **Instanz der Klasse** bezeichnet. Um eine Instanz zu erstellen, benötigen wir das Schlüsselwort `new`, den Klassennamen und ein rundes Klammersymbol. Durch das runde Klammersymbol wird der sogenannte Konstruktor aufgerufen. Standardmäßig existiert nur ein parameterloser Konstruktor. Mehr dazu jedoch [später](#). Die erzeugte Instanz kann jetzt z. B. einer Variablen zugewiesen werden. Wollen Sie nun (von „außerhalb“) auf eine der Eigenschaften oder Methoden zugreifen, so verwenden Sie auch hier den Pfeiloperator `->`.

```
1 <?php
2 $meinObjekt = new MeineKlasse();
3 ?>
```

**Konstanten** in Klassen besitzen keinen Zugriffsmodifizierer. Ein Zugriff auf Konstanten ist somit immer möglich. Üblicherweise werden Konstanten in Großbuchstaben geschrieben. Zu beachten ist, dass Konstanten zwar Teil der Klasse, aber **nicht Teil des Objekts** sind. Bei der Notation einer Konstante geben Sie das Schlüsselwort `const` gefolgt von dem Konstantennamen an. Anschließend wird der Konstante mit einem Gleichheitszeichen ein Wert zugewiesen.

```
1 <?php
2 class MeineKlasse
3 {
4     const MEINEKONSTATE = 'Mein Wert';
5 }
6 ?>
```

Um auf Konstanten einer Klasse zuzugreifen, wird der sogenannte **Gültigkeitsbereichsoperator** `::` verwendet. Dieser Operator wird neben dem Zugriff auf Klassenkonstanten auch für den Zugriff auf **statische Eigenschaften oder Methoden** verwendet. Statische Klassenbestandteile kennzeichnen sich durch das Schlüsselwort `static`, welches nach dem Zugriffsmodifizierer angegeben wird, und sind **nicht an eine Objektinstanz gebunden**.

```
1 <?php
2 echo MeineKlasse::MEINEKONSTATE; // Gibt 'Mein Wert' aus
3 ?>
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Objektorientierung](#)

In der Objektorientierung haben sich im Laufe der Jahre einige **ungeschriebene Regeln** durchgesetzt. So sollte sich jedes Objekt in einer **separaten Datei** befinden. Wie Sie Dateien einbinden, lernen Sie im [Thema Dateizugriffe](#). Private Eigenschaften und Methoden beginnen üblicherweise mit einem Unterstrich. Des Weiteren gilt es als „schlechte Praxis“, Eigenschaften als `public` zu kennzeichnen, da Sie so **keine Kontrolle** über den von „außen“ zugewiesenen Wert haben. Um dieses Problem zu lösen, werden üblicherweise sogenannte **Getter- und Setter-Methoden** verwendet. Wenn Sie z. B. eine Eigenschaft mit dem Namen `Betriebssystem` haben, sollten Sie die Methoden `getBetriebssystem()` und `setBetriebssystem()` implementieren.

Das folgende Beispiel zeigt die Klasse `Computer`. Die Eigenschaft `Betriebssystem` wurde im Beispiel absichtlich auf `public` gesetzt, um das Beispiel zu vereinfachen. Im unteren Teil des Codes sehen Sie, wie ein Objekt dieser Klasse instanziiert wird:

```

1  <?php
2  class Computer
3  {
4      const OS_WINDOWS_XP = 'Windows XP';
5      const OS_WINDOWS_7 = 'Windows 7';
6      const OS_WINDOWS_8 = 'Windows 8';
7      const OS_WINDOWS_10 = 'Windows 10';
8
9      private $_gestartet = false;
10     public $Betriebssystem = 'Windows 7';
11
12     public function starteComputer()
13     {
14         if (!$this->_gestartet)
15         {
16             echo $this->Betriebssystem.' wird gestartet ...<br />';
17             $this->_gestartet = true;
18         }
19         else
20             echo 'Computer bereits gestartet!<br />';
21     }
22
23     public function stoppeComputer()
24     {
25         if ($this->_gestartet)
26         {
27             echo $this->Betriebssystem.' wird heruntergefahren ...<br />';
28             $this->_gestartet = false;
29         }
30         else
31             echo 'Computer läuft nicht!<br />';
32     }
33 }
34
35 $meinComputer = new Computer();
36 $meinComputer->starteComputer();
37 $meinComputer->starteComputer(); // Computer läuft bereits
38 $meinComputer->stoppeComputer();
39 $meinComputer->stoppeComputer(); // Computer ist bereits heruntergefahren
40 $meinComputer->Betriebssystem = Computer::OS_WINDOWS_10;
41 $meinComputer->starteComputer();
42 ?>

```



## Konstruktor

Bei dem Konstruktor handelt es sich um eine spezielle Methode, welcher **bei der Instanziierung des Objekts** aufgerufen wird. Standardmäßig existiert bereits ein sogenannter **Standard-Konstruktor**, welcher keine Parameter erwartet. Möchten Sie in Ihrer Klasse einen Konstruktor implementieren, so können Sie auch die Konstruktor-Parameter bestimmen. Natürlich können ein oder mehrere dieser Parameter auch optional sein. Aus Sicht der Programmierung wird ein Konstruktor wie eine Funktion implementiert. Der Funktionsname des Konstruktors ist auf den Namen `__construct` festgelegt.

```

1  <?php
2  class Computer
3  {
4      const OS_WINDOWS_XP = 'Windows XP';
5      const OS_WINDOWS_7 = 'Windows 7';
6      const OS_WINDOWS_8 = 'Windows 8';
7      const OS_WINDOWS_10 = 'Windows 10';
8
9      private $_gestartet = false;
10     public $Betriebssystem = 'Windows 7';
11
12     public function __construct($betriebssystem)
13     {
14         $this->Betriebssystem = $betriebssystem;
15     }
16
17     public function starteComputer()
18     {

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislinsen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Objektorientierung](#)

```

19     if (!$this->_gestartet)
20     {
21         echo $this->Betriebssystem.' wird gestartet ...<br />';
22         $this->_gestartet = true;
23     }
24     else
25         echo 'Computer bereits gestartet!<br />';
26 }
27
28 public function stoppeComputer()
29 {
30     if ($this->_gestartet)
31     {
32         echo $this->Betriebssystem.' wird heruntergefahren ...<br />';
33         $this->_gestartet = false;
34     }
35     else
36         echo 'Computer läuft nicht!<br />';
37 }
38 }
39
40 $meinComputer = new Computer(Computer::OS_WINDOWS_7);
41 $meinComputer->starteComputer();
42 $meinComputer->stoppeComputer();
43 ?>
```



## Vererbung

Bei der Vererbung werden, wie der Name schon vermuten lässt, Daten (also Konstante, Eigenschaften und Methoden) von einer Klasse an eine andere vererbt, also „weitergegeben“. Die Klasse, von welcher Daten geerbt werden, wird als **Elternklasse** oder auch als Basisklasse bezeichnet. Die Klasse, welche Daten von einer anderen Klasse erbt, wird als **Kindklasse** bezeichnet. Um die Vererbung anzugeben, wird nach dem Namen der Kindklasse das Schlüsselwort `extends` (zu Deutsch: erweitern) und anschließend der Name der Elternklasse angegeben. Um von einer Kindklasse auf eine Elternklasse zuzugreifen, nutzen wir das Schlüsselwort `parent`. Als Operator zum Zugriff wird der Gültigkeitsbereichsoperator `::` verwendet. Funktionen, welche innerhalb der Kindklasse angegeben wurden und den gleichen Namen wie die Funktionen der Elternklasse haben, werden **überschrieben**. Wird nun eine solche überschriebene Funktion der Kindklasse aufgerufen, dann wird lediglich der Programmcode der Funktion der Kindklasse ausgeführt, nicht aber der der Elternklasse. Innerhalb der Kindklasse kann aber hier mit Hilfe des Schlüsselworts `parent` auf die überschriebene Funktion wieder zugegriffen werden.

Das untere Beispiel zeigt die Klasse `Computer` und `Notebook`. `Notebook` ist eine Kindklasse, dessen Elternklasse die Klasse `Computer` ist. In der `Notebook`-Klasse werden der Konstruktor sowie die Funktion `starteComputer()` überschrieben.

```

1 <?php
2 class Computer
3 {
4     const OS_WINDOWS_XP = 'Windows XP';
5     const OS_WINDOWS_7 = 'Windows 7';
6     const OS_WINDOWS_8 = 'Windows 8';
7     const OS_WINDOWS_10 = 'Windows 10';
8
9     private $_gestartet = false;
10    public $Betriebssystem = 'Windows 7';
11
12    public function __construct($betriebssystem)
13    {
14        $this->Betriebssystem = $betriebssystem;
15    }
16
17    public function starteComputer()
18    {
19        if (!$this->_gestartet)
20        {
21            echo $this->Betriebssystem.' wird gestartet ...<br />';
22            $this->_gestartet = true;
23        }
24        else
25            echo 'Computer bereits gestartet!<br />';
26    }
27
28    public function stoppeComputer()
29    {
30        if ($this->_gestartet)
31        {
32            echo $this->Betriebssystem.' wird heruntergefahren ...<br />';
33            $this->_gestartet = false;
34        }
35        else
36            echo 'Computer läuft nicht!<br />';

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Objektorientierung](#)

```

37     }
38 }
39
40 class Notebook extends Computer
41 {
42     public $DisplayGroesse;
43
44     public function __construct($betriebssystem, $displayGroesse)
45     {
46         parent::__construct($betriebssystem);
47
48         $this->DisplayGroesse = $displayGroesse;
49     }
50
51     public function starteComputer()
52     {
53         parent::starteComputer();
54
55         echo 'Ihr Bildschirm mit '.$this->DisplayGroesse.' Zoll wird initialisiert ...<br />';
56     }
57 }
58
59 $meinNotebook = new Notebook(Computer::OS_WINDOWS_7, 15.6);
60 $meinNotebook->starteComputer();
61 $meinNotebook->stoppeComputer();
62 ?>

```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » Fehlerbehandlung

## Fehlerbehandlung

In diesem Thema wollen wir uns mit der **Behandlung und Behebung von Fehlern** beschäftigen. Grundsätzlich muss strikt zwischen PHP-Fehlern und Exceptions unterschieden werden, weshalb wir diese in diesem Thema auch getrennt haben.

### Inhalt dieser Seite:

1. PHP-Fehlermeldung
2. Exception-Handling

## PHP-Fehlermeldungen

PHP löst in einigen Fällen Fehler aus. Dabei handelt es sich immer um **Laufzeitfehler** (da PHP-Code nicht kompiliert wird). Bei Laufzeitfehlern kann es sich um **Syntaxfehler** (da PHP zur Laufzeit interpretiert wird) handeln, aber auch um andere Fehler wie Division durch 0, ungültige Zugriffe (vor allem bei Objekten), Fehler beim Öffnen, Lesen oder Schreiben einer Datei, Fehler beim Herstellen der Datenbankverbindung uvm.. Dieses Meldungssystem von PHP wird auch **Error-Reporting-System** bezeichnet. Die folgende Tabelle zeigt die wichtigsten Typen von Fehlern:

Wert	Konstante	Beschreibung
1	E_ERROR	fataler Fehler, dies hat einen Skriptabbruch zur Folge
2	E_WARNING	Warnung
4	E_PARSE	Fehler beim Interpretieren, dies hat einen Skriptabbruch zur Folge
8	E_NOTICE	Benachrichtigungen
2048	E_STRICT	Benachrichtigungen über Codeverbesserungen
8192	E_DEPRECATED	Benachrichtigungen über veraltete Codebestandteile
32767	E_ALL	alle Fehler, Warnungen und Benachrichtigungen

Mit Hilfe der Funktion `error_reporting()` können Sie steuern, **welche Fehlermeldungen ausgegeben bzw. gemeldet werden** sollen. Als Parameter wird der sogenannte Fehler-Level übergeben. Davon kann als Wert ein numerischer Wert oder eine Konstante aus der obigen Tabelle verwendet werden. Die Werte können bei Bedarf bitweise verundet und verodert werden. Für Entwicklungszwecke eignet sich die Verwendung des Fehler-Levels `E_ALL`.

```

1 <?php
2 error_reporting(E_ALL);
3 >

```

Die Funktion `error_log()` erlaubt es, **Fehlermeldungen auszulösen**. Der Funktion muss eine Nachricht (Datentyp `String`) übergeben werden. Optional kann ein Typ (`message_typ`) und ein Ziel (`destination`) übergeben werden. Mit dem Parameter `message_typ` wird festgelegt, wo der Fehler hingesendet wird: PHP-Log-System (0), E-Mail versenden (1), anhängen an Datei (3). Der `destination`-Parameter wird nur für den E-Mail-Versand (als E-Mail-Adresse) und beim Anhängen an eine Datei (als Dateiname) benötigt.

```

1 <?php
2 error_log('Fehler bei der Datenbankverbindung!', 1, 'info@example.com');
3 >

```

Wenn Sie Ihre Webseite veröffentlichen, sollten Sie vor allem danach schauen, Fehler zu behandeln, indem Sie z. B. eine Variable auf `null` oder vor einer Division auf 0 prüfen. Trotzdem kann es natürlich immer wieder zu Fehlern kommen (die entweder nicht abgefangen wurden oder nicht abgefangen werden können), die dem Besucher natürlich **nicht angezeigt** werden sollten. Der „einfachste Weg“, der auch in vielen Foren als Lösung vorgeschlagen wird, ist die **Deaktivierung der Fehler-Benachrichtigung** mittels der Funktion `error_reporting()`. Das Problem an dieser Lösung ist, dass Fehler zwar nicht mehr angezeigt aber auch nicht mehr gemeldet werden. Dies hat zur Folge, dass Fehler im **Serverlogfile** nicht mehr geloggt werden. So treten auf Ihrer Webseite u. U. Fehler auf, die Sie nie bemerken würden. Um dieses Problem zu lösen, stellen Sie die Fehler-Benachrichtigung mittels der Funktion `error_reporting()` so ein wie gewünscht (z. B. nur Fehler und Warnungen). Nun sollte die PHP-Einstellung `display_errors` auf 0 und die Einstellung `log_errors` auf 1 gesetzt werden. Dadurch werden Fehler nicht mehr angezeigt jedoch geloggt. Beide Einstellungen können mittels der Funktion `ini_set()` oder innerhalb der Konfigurationsdatei `php.ini` geändert werden. Wird die Funktion `ini_set()` verwendet, so gilt die Einstellung nur für die Ausführung des aktuellen Skripts.

```

1 <?php
2 ini_set('display_errors', '0');
3 ini_set('log_errors', '1');
4 >

```

## Exception-Handling

Nun wollen wir uns noch mit dem sogenannten Exception-Handling (zu Deutsch **Ausnahmebehandlung**) beschäftigen. Viele PHP-Funktionen lösen keine Ausnahmen (engl. *exceptions*) aus, sondern nutzen das Error-Reporting-System, welches wir weiter oben bereits beschrieben haben, um Fehler „an den Programmierer“ weiterzugeben. Einige neue „moderne“ PHP-Komponenten, wie z. B. die `PDO`-Klasse für Datenbankzugriffe, nutzen jedoch das `Exception`-System. Des Weiteren ist es Ihnen möglich, Ausnahmen selber auszulösen bzw., wie es in der Fachsprache heißt, zu werfen.

Exceptions können von Funktionen, welche im PHP-Interpreter integriert sind oder von selbst geschriebenen Funktionen „geworfen“ (also ausgelöst) werden. Exceptions sollten unbedingt abgefangen werden, da **andernfalls ein fataler PHP-Fehler** (`E_ERROR`) gemeldet wird, was ein Skriptabbruch zur Folge hat. Um eine Exception abzufangen, muss der Code, welcher eine Exception auslösen kann, innerhalb eines `try`-Blocks notiert werden. Hinter dem `try`-Block folgt ein `catch`-Block. Der Code im `catch`-Block enthält einen **Programmcode zur Fehlerbehandlung**. Dieser wird nur ausgeführt, wenn eine Exception geworfen wurde. Um auf die Informationen der Exceptions zuzugreifen, werfen alle Exceptions ein **Objekt mit Fehlerinformationen**. Darauf kann vom `catch`-Block zugegriffen werden. Alle Ausnahmeobjekte sind dabei eine Instanz von der Klasse `Exception` oder von einer der Unterklassen (z. B. `PDOException`). Um z. B. auf die **Fehlermeldung** zuzugreifen, können wir die Funktion `getMessage()` aufrufen. Mit Hilfe der Funktionen `getFile()` und `getLine()` kann der Dateiname und die dazugehörige Zeile abgerufen werden, in welcher die Ausnahme geworfen wurde.

```

1 <?php
2 try
3 {

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » Fehlerbehandlung

```

4 | // Code der Exceptions wirft
5 | }
6 | catch (Exception $ex)
7 | {
8 |     // Code zur Behandlung der Fehler
9 | }
10| ?>

```

Nach dem `try`- und `catch`-Block kann noch zusätzlich ein `finally`-Block folgen. Der Code im `finally`-Block wird nach der vollständigen Ausführung des `try`-Blocks oder des `catch`-Blocks ausgeführt. Der `finally`-Block dient grundsätzlich zur **Freigabe von Ressourcen**.

```

1 | <?php
2 | try
3 | {
4 |     // Code der Exceptions wirft
5 | }
6 | catch (Exception $ex)
7 | {
8 |     // Code zur Behandlung der Fehler
9 | }
10| finally
11| {
12|     // Code zur Freigabe von Ressourcen
13| }
14| ?>

```

Um eine **Ausnahme zu werfen**, notieren wir das Schlüsselwort `throw`, gefolgt von einem Objekt der Klasse `Exception` oder einer der Unterklassen. Üblicherweise wird die Objektinstanziierung direkt mit dem `throw`-Statement ausgeführt. Dem Konstruktor der `Exception`-Klasse kann optional eine Nachricht übergeben werden, welche mittels der Funktion `getMessage()` abgerufen werden kann.

```

1 | <?php
2 | throw new Exception('Fehler bei der Datenbankverbindung!');
3 | ?>

```

Das folgende Beispiel zeigt eine einfache und doch reelle Variante, wie Ausnahmen verwendet werden:

```

1 | <?php
2 | $a = 4;
3 | $b = 0;
4 |
5 | try
6 | {
7 |     if ($b == 0)
8 |         throw new Exception('Division durch 0 nicht erlaubt!');
9 |     echo $a / $b;
10| }
11| catch (Exception $ex)
12| {
13|     echo 'Meldung: '.$ex->getMessage();
14| }
15| ?>

```

**Wichtig:** Auch wenn es im ersten Moment vielleicht etwas verwirrend klingen mag, Exceptions zu werfen nur um diese später wieder abfangen zu können, kann es gerade bei größeren Projekten Sinn machen.

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Datumsangaben](#)

## Datumsangaben

Um mit Datumsangaben zu arbeiten, gibt es in PHP einige Funktionen, die jedoch größtenteils einen sehr speziellen Verwendungszweck haben. Es gibt jedoch zwei einfache, aber auch grundlegende Funktionen, welche wir Ihnen hier vorstellen möchten: `time()` und `date()`. Die Funktion `time()` gibt den aktuellen sogenannten **UNIX-Zeitstempel** zurück. Ein UNIX-Zeitstempel enthält die Anzahl an Sekunden seit dem 01.01.1970 00:00:00. Mit Hilfe der Funktion `date()` ist es möglich, einen UNIX-Zeitstempel **formatiert**, also in Form eines Datums und / oder einer Uhrzeit, darzustellen. Der `date()`-Funktion werden ein Format und ein Zeitstempel übergeben. Wird der Zeitstempel weggelassen, so wird als Wert der Rückgabewert von `time()` verwendet, was zur Folge hat, dass das aktuelle Datum mit Uhrzeit verwendet wird. Das Format wird als Zeichenkette übergeben. Für das Format werden **Kürzel**, auch als Format-Zeichen bezeichnet, verwendet. Die wichtigsten Kürzel sind in der untenstehenden Tabelle zu sehen.

<b>l</b>	Wochentag (Monday bis Sunday)
<b>D</b>	Wochentag (Mon bis Sun)
<b>d</b>	Tag (01 bis 31)
<b>j</b>	Tag (1 bis 31)
<b>W</b>	Kalender-Woche (1 bis 53)
<b>F</b>	Monat (January bis December)
<b>M</b>	Monat (Jan bis Dec)
<b>m</b>	Monat (01 bis 12)
<b>n</b>	Monat (1 bis 12)
<b>Y</b>	Jahr (z. B. 2016)
<b>y</b>	Jahr (z. B. 16)
<b>H</b>	Stunde (00 bis 23)
<b>h</b>	Stunde (01 bis 12)
<b>G</b>	Stunde (0 bis 23)
<b>g</b>	Stunde (1 bis 12)
<b>i</b>	Minute (00 bis 59)
<b>s</b>	Sekunde (00 bis 59)

Das folgende Beispiel zeigt 3 Datumsausgaben (vor 1 Stunde, aktuell und in 1 Stunde) mit Verwendung der `time()` - und `date()` -Funktion:

```

1 <?php
2 $zeit = time();
3
4 echo 'Zeit vor 1h: '.date('d.m.Y H:i:s', $zeit - (60 * 60)).'<br />';
5 echo 'Aktuelle Zeit: '.date('d.m.Y H:i:s').' ('.$zeit.')<br />';
6 echo 'Zeit in 1h: '.date('d.m.Y H:i:s', $zeit + (60 * 60));
7 ?>

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislngen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Dateizugriffe](#)

## Dateizugriffe

In diesem Thema wollen wir uns damit beschäftigen, wie Sie mit Dateien arbeiten können. Hierzu zählt das Lesen, Schreiben und Anhängen von / an Dateien sowie das Einbinden von Dateien, um Quellcode auszulagern. In den Beispielen der Unterthemen [Dateiinhalte einlesen](#) und [Arbeiten mit Datei-Handles](#) wird die folgende Text-Datei verwendet (Zeilenbruch mittels `\r\n`):

```

1 | Wichtig: Die Datei dient nur zum Test.
2 |
3 | Diese Datei enthält ein paar Zeilen
4 | mit "reinem" Text.
5 | Verwendet wird die Datei mit
6 | verschiedenen PHP-Funktionen.
7 |
8 | Zur Erinnerung: Die Datei dient nur zum Test.

```

Möchten Sie prüfen, ob eine **Datei existiert**, so können Sie die Funktion `file_exists()` aufrufen, welcher als Parameter der Dateiname mit Pfad übergeben wird und als Rückgabe ein Wert vom Typ `Boolean` zurückgibt. Die Funktion `filesize()` erlaubt es, die **Dateigröße einer Datei zu ermitteln**. Als Parameter wird der Dateiname übergeben und als Rückgabewert erhalten Sie die Dateigröße oder `false` im Fehlerfall.

## Dateiinhalte einlesen

Möchten Sie den **kompletten Inhalt** einer Datei einlesen, so können Sie die Funktion `file_get_contents()` verwenden. Als Parameter übergeben Sie dieser den Dateinamen mit Pfad. Die Pfadangabe kann relativ oder absolut sein. Des Weiteren ist es möglich, eine URL anzugeben, um z. B. den Inhalt einer anderen Seite zu laden. Die Funktion `file_get_contents()` gibt als Rückgabewert eine Zeichenkette zurück. Schlägt das Auslesen der Datei fehl, so wird `false` zurückgegeben. Im folgenden Beispiel wird eine Datei eingelesen, an den Zeilenbrüchen getrennt und anschließend alle Zeilen nacheinander ausgegeben:

```

1 | <?php
2 | $inhalt = file_get_contents('datei-beispiel.txt');
3 |
4 | $zeilen = explode("\r\n", $inhalt);
5 |
6 | foreach ($zeilen as $zeile)
7 |     echo $zeile.'<br />';
8 | ?>

```



## Arbeiten mit Datei-Handles

Wollen Sie nur einen **Teil einer Datei auslesen** oder in eine **Datei schreiben**, so müssen Sie Datei-Handles verwenden. Ein **Handle** (zu Deutsch Griff oder Henkel) ist eine Referenz auf eine Ressource (in diesem Fall eine Datei) des Systems. Um ein Datei-Handle zu erhalten, müssen wir die Funktion `fopen()` aufrufen, welche eine Datei öffnet. Der Funktion `fopen()` werden der Dateiname (ggf. mit Pfad) und ein Modus übergeben. Für den Modus sind einige Kürzel verfügbar, welche Sie bitte der folgenden Tabelle entnehmen:

Kürzel	Lesen	Schreiben	Dateizeiger	Dateiinhalte	Datei
r	ja	nein	Anfang	-	-
r+	ja	ja	Anfang	-	-
w	nein	ja	Anfang	alter Inhalt wird überschrieben	wird ggf. erstellt
w+	ja	ja	Anfang	alter Inhalt wird überschrieben	wird ggf. erstellt
a	nein	ja	Ende	neuer Inhalt wird angehängt	wird ggf. erstellt
a+	ja	ja	Ende	neuer Inhalt wird angehängt	wird ggf. erstellt

Die Funktion `fopen()` gibt ein Datei-Handle oder `false` zurück, wenn ein Fehler auftritt. Hier ein Beispiel für den Aufruf der Funktion `fopen()`:

```

1 | <?php
2 | $handle = fopen('datei-beispiel.txt', 'r')
3 | ?>

```

Um aus einer Datei (mittels Datei-Handle) zu **lesen**, gibt es die Funktion `fread()`. Der Funktion `fread()` werden das Datei-Handle und die Anzahl der zu lesenden Bytes übergeben. Die Funktion gibt eine Zeichenkette oder im Fehlerfall `false` zurück.

```

1 | <?php
2 | $daten = fread($handle, 20);
3 | ?>

```

Das **Schreiben** in eine Datei erfolgt mittels der Funktion `fwrite()`. Der Funktion werden das Datei-Handle, eine Zeichenkette und optional die Anzahl der zu schreibenden Bytes übergeben. Wird die Anzahl der zu schreibenden Bytes nicht mit übergeben, so wird die Länge der Zeichenkette bestimmt und diese als Längenangabe verwendet. `fwrite()` gibt die Anzahl der geschriebenen Bytes oder im Fehlerfall `false` zurück.

```

1 | <?php
2 | $len = fwrite($handle, $daten);
3 | ?>

```

Die Funktion `ftell()` ermöglicht es, die aktuelle Position in dem Datei-Handle, den sogenannten **Dateizeiger**, zu ermitteln. Als Rückgabe erhalten Sie die

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Dateizugriffe](#)

**Position in der Datei** (Dateianfang entspricht der Position 0). Schlägt die Funktion fehl, so wird `false` zurückgegeben.

```
1 <?php
2 $pos = ftell($handle);
3 ?>
```

So wie der Dateizeiger ausgelesen werden kann, kann der Dateizeiger auch gesetzt werden. Hierfür wird die Funktion `fseek()` verwendet, welcher das Datei-Handle und die Dateiposition übergeben werden muss.

```
1 <?php
2 fseek($handle, 40);
3 ?>
```

Möchten Sie prüfen, ob sich der Dateizeiger am Ende befindet, so können Sie die Funktion `feof()` aufrufen.

```
1 <?php
2 $ende = feof($handle);
3 ?>
```

Sobald Sie mit einem Datei-Handle fertig sind und es nicht mehr benötigen, sollten Sie es wieder **schließen**. Hierfür verwenden Sie die Funktion `fclose()`.

```
1 <?php
2 fclose($handle);
3 ?>
```

Im folgenden Beispiel wird ein Teil der Datei ausgelesen und ausgegeben. Zusätzlich wird, bevor die Datei geschlossen wird, die Position des Dateizeigers ausgegeben:

```
1 <?php
2 $handle = fopen('datei-beispiel.txt', 'r');
3
4 fseek($handle, 42);
5 echo str_replace("\r\n", '<br />', fread($handle, 118));
6
7 echo '<br /><br />Position: '.ftell($handle);
8
9 fclose($handle);
10 ?>
```



## Datei einbinden

Wenn Sie Funktionen oder Objekte in mehrere Dateien aufteilen wollen, so müssen Sie die Möglichkeit haben, Dateien einzubinden. Eingebundene Dateien müssen nicht zwingend PHP-Code enthalten, sondern können z. B. auch nur HTML-Code enthalten. Befindet sich in der Datei PHP-Code, so wird dieser vom PHP-Interpreter ausgeführt. Insgesamt gibt es für das Einbinden vier verschiedene sogenannte Sprachkonstrukte: `include`, `include_once`, `require` und `require_once`. Bei allen Sprachkonstrukten wird nach dem jeweiligen Schlüsselwort der Dateiname mit Pfad (Zeichenkette) und einem abschließenden Semikolon angegeben. Bei allen vier Sprachkonstrukten ist auch eine alternative Schreibweise wie bei Funktionen möglich.

```
1 <?php
2 include 'datei2.php';
3 include('datei2.php'); // Entspricht der obigen Zeile
4 ?>
```

Der Sprachkonstrukt `require` verhält sich fast identisch wie `include`, nur mit dem Unterschied, dass wenn die angegebene Datei nicht gefunden wurde oder ein Zugriff auf die Datei nicht möglich ist, ein **fataler PHP-Fehler** ausgelöst wird. Das Gleiche gilt für `require_once` im Verhältnis zu `include_once`. Die Befehle `include_once` und `require_once` entsprechen `include` und `require` mit dem Unterschied, dass geprüft wird, ob die **angegebene Datei bereits eingebunden** wurde. Ist dies der Fall, so wird der Befehl ignoriert.

```
1 <?php
2 require 'datei-einbinden-2.php';
3
4 echo $a * $b;
5 ?>
```

**datei-einbinden-2.php:**

```
1 <?php
2 $a = 7;
3 $b = 4;
4 ?>
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Umgebungsinformationen und Formulardaten](#)

## Umgebungsinformationen und Formulardaten

In PHP gibt es drei **wichtige assoziative Arrays**, mit welchen Umgebungsinformationen, URL-Parameter sowie Formulardaten abgerufen werden können. Des Weiteren kann man von PHP aus, einige Seiteninformationen ändern, um somit z. B. eine Weiterleitung durchzuführen.

- Inhalt dieser Seite:**

  1. Umgebungsinformationen
  2. Weiterleitung
  3. URL-Parameter
  4. Formulardaten

### Umgebungsinformationen

**PHP Variables**

Name	Typ
<code>\$_SERVER</code>	Array
<code>\$_POST</code>	Array
<code>\$_GET</code>	Array
<code>\$_FILES</code>	Array
<code>\$_COOKIE</code>	Array
<code>\$_ENV</code>	Array
<code>\$argc</code>	int
<code>\$argv</code>	Array
<code>\$HTTP_RAW_POST_DATA</code>	string
<code>\$HTTP_USER_AGENT</code>	string
<code>\$SERVER_ADDR</code>	string
<code>\$REMOTE_ADDR</code>	string
<code>\$SERVER_PROTOCOL</code>	string
<code>\$REQUEST_METHOD</code>	string
<code>\$REQUEST_URI</code>	string
<code>\$SCRIPT_FILENAME</code>	string
<code>\$PHP_SELF</code>	string
<code>\$DOCUMENT_ROOT</code>	string
<code>\$SCRIPT_NAME</code>	string
<code>\$PHP_VERSION</code>	string
<code>\$PHP_OS</code>	string
<code>\$PHP_VERSION_NUM</code>	float
<code>\$PHP_EXTRA_VERSION</code>	string
<code>\$PHP_ZIP</code>	string
<code>\$PHP_UPLOAD_PROGRESS_PREFIX</code>	string
<code>\$PHP_CRYPT_LIBRARY</code>	string
<code>\$PHP_CRYPT_VERSION</code>	string
<code>\$PHP_CRYPT_TIMESTAMP</code>	string
<code>\$PHP_CRYPT_TIMESTAMP_FORMAT</code>	string
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS</code>	int
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_FORMAT</code>	string
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP</code>	int
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_FORMAT</code>	string
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS</code>	int
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_FORMAT</code>	string
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP</code>	int
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_FORMAT</code>	string
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS</code>	int
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_FORMAT</code>	string
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP</code>	int
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_FORMAT</code>	string
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS</code>	int
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_FORMAT</code>	string
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP</code>	int
<code>\$PHP_CRYPT_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_SECONDS_TIMESTAMP_FORMAT</code>	string

PHP bietet den Zugriff auf verschiedene Umgebungsinformationen an. Hierfür wird das assoziative Array `$_SERVER` genutzt. Bei `$_SERVER` handelt es sich um eine sogenannte **Superglobale**. Superglobale sind in jedem Gültigkeitsbereich (Scope) verfügbar. Um alle verfügbaren Umgebungsinformationen zu sehen, können Sie entweder das Array mittels `print_r` ausgeben oder sich den Bereich „PHP Variables“ in der Ausgabe der Funktion `phpinfo()` anschauen (siehe Bild).

Das untenstehende Beispiel sollte selbsterklärend sein. Ein paar Informationen noch, was Sie mit den Informationen anfangen können: Der **User-Agent** kann z. B. für **einfache serverseitige Statistiken** genutzt werden, da dieser, sofern er vom Benutzer nicht verändert wurde, den Browser und das Betriebssystem enthält. Die **Remote-Adresse** enthält die IP-Adresse des Besuchers und kann z. B. beim **Dokumentieren von Anmeldungen** auf der Webseite verwendet werden (bitte lokal geltendes Gesetz beachten!). Des Weiteren finden sich in den Umgebungsinformationen einige **Pfadangaben**, um den Dateinamen des aktuellen Skripts / der aktuellen Anfrage bzw. dessen Pfad zu ermitteln.

```

1 <<?php
2 echo 'Host: ' . $_SERVER['HTTP_HOST'] . '<br />';
3 echo 'User-Agent: ' . $_SERVER['HTTP_USER_AGENT'] . '<br />';
4 echo 'Server-Adresse: ' . $_SERVER['SERVER_ADDR'] . '<br />';
5 echo 'Remote-Adresse: ' . $_SERVER['REMOTE_ADDR'] . '<br />';
6 echo 'Protokoll: ' . $_SERVER['SERVER_PROTOCOL'] . '<br />';
7 >>
    
```

### Weiterleitung

Eine Weiterleitung oder **andere Änderungen am HTTP-Header** können Sie mittels der Funktion `header()` durchführen. Um mit der Funktion `header()` eine Weiterleitung durchzuführen, übergeben Sie der Funktion eine Zeichenkette mit dem Inhalt `'Location: pfad/zur/datei.php'`. Alle weiteren Header haben einen identischen Aufbau: Name des Header-Felds, Doppelpunkt, Leerzeichen und der Wert für das Header-Feld. Alle verfügbaren Header-Felder entnehmen Sie [RFC 2616](#). Der Aufruf der Funktion `header()` muss zwingend vor allen anderen Ausgaben und jeglichem HTML-Code erfolgen, andernfalls kann die Weiterleitung nicht mehr erfolgen, da der HTTP-Header bereits vollständig versendet wurde. Nach dem Aufruf der `header()`-Funktion muss die Funktion `exit()` aufgerufen werden, um somit die **Skriptausführung zu stoppen**. `exit` ist ein Sprachkonstrukt, weshalb die Klammern weggelassen werden können.

```

1 <<?php
2 header('Location: umgebung.php');
3 exit();
4 >>
    
```

### URL-Parameter

URL-Parameter werden, wie der Name schon sagt, **in der URL mitgegeben**. Dabei wird hinter dem Ordner- und Dateiname ein Fragezeichen `?` angehängt. Anschließend folgt eine Angabe in der Form `name=wert`. Werden mehrere Werte übergeben, so wird dazwischen ein Und-Zeichen `&` angegeben. Dies sieht dann z. B. so aus: `?name1=wert1&name2=wert2&name3=wert3`. Diese URL-Parameter können Sie z. B. in HTML-Links mitgeben. Des Weiteren werden solche Parameter zur Weitergabe von Formulardaten verwendet, sofern die Methode im `form`-Element auf `GET` eingestellt ist (bzw. das Attribut weggelassen wird). Der Zugriff auf die URL-Parameter von PHP aus, erfolgt mittels des assoziativen Arrays `$_GET`, wobei es sich um eine Superglobale handelt. Möchten Sie **prüfen, ob ein URL-Parameter mitgesendet wurde**, so können Sie die Funktion `isset()` verwenden. Der Funktion `isset()` übergeben Sie als Parameter die zu prüfende Variable (siehe Beispiel). Als Rückgabewert gibt die Funktion einen Wert vom Typ `Boolean` zurück.

```

1 <<?php
2 echo 'Wert a: ' . (isset($_GET['a']) ? $_GET['a'] : '-') . '<br />';
3 echo 'Wert b: ' . (isset($_GET['b']) ? $_GET['b'] : '-') . '<br />';
4 >>
5 <br />
6 <b>Links:</b><br />
7 <a href="url-parameter.php">A: -, B: -</a><br />
8 <a href="url-parameter.php?a=7">A: 7, B: -</a><br />
9 <a href="url-parameter.php?b=4">A: -, B: 4</a><br />
    
```



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Umgebungsinformationen und Formulardaten](#)

10 | [A: 8, B: 3](url-parameter.php?a=8&b=3)



## Formulardaten

Bei den meisten **HTML-Formularen** wird als **Methode POST** verwendet. Der Vorteil von POST ist, dass die Daten „unsichtbar“ versendet werden. Des Weiteren ist es dem Besucher dadurch z. B. nicht möglich, einen Link der Seite zu speichern, in welcher die Formulardaten mit enthalten sind (was in den meisten Fällen unerwünscht wäre). Die URL-Parameter und somit die Methode GET wird hauptsächlich bei der **Filterung von Daten** (z. B. aus einer Datenbank) oder auch für die in Shops, Foren und Blogs verbreiteten **Blätter-Funktionen** verwendet. Auf die (POST-)Formulardaten können Sie in PHP mittels dem superglobalen assoziativen Array `$_POST` zugreifen. Die Funktion `isset()` kann auch hier zur Prüfung, ob der Wert vorhanden ist, verwendet werden.

```

1  <?php
2  echo 'Wert "anrede": ' . (isset($_POST['anrede']) ? $_POST['anrede'] : '-') . '<br />';
3  echo 'Wert "vorname": ' . (isset($_POST['vorname']) ? $_POST['vorname'] : '-') . '<br />';
4  echo 'Wert "nachname": ' . (isset($_POST['nachname']) ? $_POST['nachname'] : '-') . '<br />';
5  ?>
6  <br />
7  <form action="formulardaten.php" method="post">
8      <table>
9          <tr>
10             <td style="width: 100px;">Anrede:</td>
11             <td>
12                 <select name="anrede">
13                     <option value="H">Herr</option>
14                     <option value="F">Frau</option>
15                 </select>
16             </td>
17         </tr>
18         <tr>
19             <td style="width: 100px;">Vorname:</td>
20             <td><input type="text" name="vorname" /></td>
21         </tr>
22         <tr>
23             <td style="width: 100px;">Nachname:</td>
24             <td><input type="text" name="nachname" /></td>
25         </tr>
26         <tr>
27             <td></td>
28             <td><input type="submit" value="Absenden" /></td>
29         </tr>
30     </table>
31 </form>

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Sessions und Cookies](#)

## Sessions und Cookies

Sessions und Cookies erlauben es, **Variablen anfrageübergreifend zu speichern**.

### Inhalt dieser Seite:

1. Session
2. Cookie

### Session

Sessions erlauben es, Variablen **innerhalb einer Sitzung zu speichern**. Die Sitzung erstreckt sich vom expliziten Start der Session bis zur expliziten Zerstörung der Session oder dem Schließen des Browsers. Um eine **Session zu starten**, muss die Funktion `session_start()` aufgerufen werden. In jeder Datei, in welcher auf die Session-Variablen lesend oder schreibend zugegriffen werden muss, muss die Funktion `session_start()` aufgerufen werden. Einzelne Session-Variablen existieren natürlich erst ab dem Zeitpunkt, ab dem diese das erste Mal zugewiesen (initialisiert) wurden. Wenn Sie prüfen möchten, ob eine Session-Variablen existiert, können Sie die Funktion `isset()` verwenden. Möchten Sie einzelne **Session-Variablen wieder löschen**, so können Sie die Funktion `unset()` nutzen, welcher Sie die zu löschende Variable übergeben müssen. Sollen alle Session-Variablen und somit die komplette **Session gelöscht** werden, so können Sie die Funktion `session_destroy()` aufrufen.

Beim ersten Aufruf der Funktion `session_start()` wird vom PHP-Interpreter eine sogenannte **Session-ID** erzeugt. Diese wird dann verwendet, um den **Besucher zu identifizieren**. Die ID kann entweder in der URL mitgegeben werden oder beim Besucher mittels eines Cookies gespeichert werden. In den meisten Fällen wird die Cookie-Variante verwendet, weil die andere einige Probleme mit sich bringt, weshalb wir auf diese auch nicht näher eingehen. Bei der Cookie-Variante wird beim Setzen des Cookies dem Browser mitgeteilt, dass das Cookie **nur während dieser Browsersitzung gültig** ist, weshalb der Browser das Cookie nicht zwingend als Datei abspeichern muss. Das Cookie für die Session-ID trägt standardmäßig den Namen `PHPSESSID`. Die Werte der Session-Variablen können vom Benutzer nicht betrachtet und nicht manipuliert werden, da diese **auf dem Server gespeichert** sind. Der Zugriff auf die Session-Variablen erfolgt mittels des superglobalen assoziativen Arrays `$_SESSION`. Im folgenden Beispiel werden die Formulareingaben (Vorname und Nachname) in einer Session gespeichert. Rufen Sie die Seite auf, füllen Sie das Formular aus, schließen Sie die Seite, rufen Sie diese erneut auf und Sie werden feststellen, dass Ihre Formulareingaben immer noch da sind. Wenn Sie den Browser schließen und die Seite erneut aufrufen, so werden Sie feststellen, dass die Formularfelder wieder leer sind.

```

1 <?php
2 session_start();
3
4 if (isset($_POST['vorname']) && isset($_POST['nachname']))
5 {
6     $_SESSION['vorname'] = $_POST['vorname'];
7     $_SESSION['nachname'] = $_POST['nachname'];
8 }
9 ?>
10
11 <form action="session.php" method="post">
12 <table>
13 <tr>
14 <td style="width: 100px;">Vorname:</td>
15 <td><input type="text" name="vorname" value="<?php echo (isset($_SESSION['vorname']) ? $_SESSION['vorname'] :
16 '); ?>" /></td>
17 </tr>
18 <tr>
19 <td style="width: 100px;">Nachname:</td>
20 <td><input type="text" name="nachname" value="<?php echo (isset($_SESSION['nachname']) ? $_SESSION['nachname'] :
21 '); ?>" /></td>
22 </tr>
23 <tr>
24 <td></td>
25 <td><input type="submit" value="Session schreiben" /></td>
26 </tr>
27 </table>
28 </form>

```



### Cookie

Bei einem Cookie (zu Deutsch Keks) handelt es sich vereinfacht dargestellt um eine **Textdatei**, welche **auf dem Computer des Besuchers gespeichert** wird. Ein Cookie ist so lange gültig wie angegeben, dies kann nur die aktuelle Browsersitzung sein (wie z. B. bei dem Session-ID-Cookie) oder bis zu einem bestimmten angegebenen Zeitpunkt. Cookies werden vom Webserver gesetzt und können in PHP mittels der Funktion `setcookie()` gesetzt werden. Der Funktion wird der Cookie-Name, der Cookie-Wert, die Gültigkeit, der Pfad, die Domain, ein Wert zur Angabe, ob das Cookie nur bei HTTPS gesendet werden soll, und ein Wert zur Angabe, ob das Cookie nur per HTTP erreichbar ist, übergeben. Alle Parameter, abgesehen vom ersten Parameter für den Cookie-Namen, sind optional. Für gewöhnlich werden jedoch auch noch die Parameter für den Cookie-Wert und die Gültigkeit mit übergeben. Die anderen Parameter werden seltener benötigt. Dem **Gültigkeits-Parameter** wird ein UNIX-Zeitstempel übergeben, um somit die **Ablaufzeit** des Cookies festzulegen. Wird hier 0 übergeben, so gilt das Cookie nur bis zum Ende der Browsersitzung. Um ein **Cookie zu löschen**, übergeben Sie als Ablaufzeitpunkt einen abgelaufenen Zeitstempel (z. B. `time() - 3600`). Mit dem Pfad- und Domain-Parameter ist es möglich, den **Gültigkeitsort** des Cookies zu verändern. Standardmäßig gilt ein Cookie nur für das aktuelle Verzeichnis und deren Unterverzeichnisse. Der Aufruf der Funktion `setcookie()` muss, wie der Aufruf der Funktion `header()`, erfolgen, bevor irgendeine Art von Ausgabe gesendet wird. Nachdem ein Cookie gesetzt wurde, wird dessen Wert bei einer Anfrage **vom Browser an den Server geschickt**. In PHP kann auf den Cookie-Wert mittels des superglobalen assoziativen Arrays `$_COOKIE` zugegriffen werden. Das folgende Beispiel ist mit dem obigen Beispiel für Sessions vergleichbar. Zur Speicherung der Formular Daten werden jedoch die Cookies `vorname` und `nachname` verwendet, welche für 1 Stunde gültig sind.



```

1 <?php
2 if (isset($_POST['vorname']) && isset($_POST['nachname']))
3 {
4     // Aktuelle Werte lokal merken

```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Sessions und Cookies](#)

```

5     $vorname = $_POST['vorname'];
6     $nachname = $_POST['nachname'];
7
8     // Cookies schreiben
9     setcookie('vorname', $vorname, time() + 3600);
10    setcookie('nachname', $nachname, time() + 3600);
11  }
12  else
13  {
14    // Aktuelle Werte wenn verfügbar aus Cookie laden
15    $vorname = isset($_COOKIE['vorname']) ? $_COOKIE['vorname'] : '';
16    $nachname = isset($_COOKIE['nachname']) ? $_COOKIE['nachname'] : '';
17  }
18  ?>
1
2  <form action="cookie.php" method="post">
3    <table>
4      <tr>
5        <td style="width: 100px;">Vorname:</td>
6        <td><input type="text" name="vorname" value="<?php echo $vorname; ?>" /></td>
7      </tr>
8      <tr>
9        <td style="width: 100px;">Nachname:</td>
10       <td><input type="text" name="nachname" value="<?php echo $nachname; ?>" /></td>
11     </tr>
12     <tr>
13       <td></td>
14       <td><input type="submit" value="Cookie schreiben" /></td>
15     </tr>
16   </table>
</form>

```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [E-Mail-Versand](#)

## E-Mail-Versand

### Inhalt dieser Seite:

1. Text E-Mail
2. HTML E-Mail

In PHP wird die Funktion `mail()` dazu verwendet, E-Mails zu versenden. Dabei kann es sich um eine Text-Nachricht, eine HTML-Nachricht oder sogar um eine „Multipart Message“ (damit ist zumeist eine E-Mail mit Anhang gemeint) handeln. Beim Versenden von E-Mails mit Anhängen wird jedoch gerne auf die Library `PHPMailer` zurückgegriffen, da das Zusammenbauen der E-Mail-Header für solche komplexe E-Mails komplizierter ist. Der Funktion `mail()` müssen der Empfänger, der Betreff und der Inhalt übergeben werden. Zusätzlich können E-Mail-Header und Parameter (für das E-Mail-Programm) übergeben werden.

Die Angabe des **Empfängers** enthält die E-Mail-Adresse und ggf. einen Anzeigenamen (welcher z. B. von E-Mail-Clients verwendet wird). Gibt es mehrere Empfänger, so werden diese durch Komma getrennt. Eine detaillierte Spezifikation finden Sie in [RFC 2822](#). Hier sind einige Beispiele für gültige Empfängerangaben:

```
1 <?php
2 $toEinfach = 'info@example.com';
3 $toEinfachMitName = 'Example GmbH <info@example.com>';
4 $toMehrfach = 'info@example.com, vertrieb@example.com';
5 $toMehrfachMitName = 'Example GmbH <info@example.com>, Example GmbH - Vertrieb <vertrieb@example.com>;?>
```

Der **Betreff** kann „direkt“ als Zeichenkette angegeben werden. Enthält die Zeichenkette Sonderzeichen (z. B. Ä, Ö oder Ü), so muss diese speziell formatiert werden, da andernfalls der Betreff falsch dargestellt wird. Grund dafür ist, dass der SMTP-Header (in welchem sich der Betreff verbindet) nur ASCII-Zeichen enthalten darf. Eine Möglichkeit, wie der Betreff formatiert werden kann, ist die Konvertierung in eine BASE64-Zeichenkette (siehe 2. Beispiel). Eine genaue Beschreibung zu den Möglichkeiten der Betreff-Formatierung bzw. -Konvertierung finden Sie in [RFC 2047](#).

```
1 <?php
2 $betreff = 'Frage zum Hosting-Angebot';
3 $betreffMitSonderzeichen = '?=UTF-8?B?'.base64_encode('Frage zu Homepage-Baukästen').'=?';?>
```

Der **Nachrichteninhalt** selbst stellt keine Besonderheit dar. Die Nachricht kann nur aus Zeichen oder, wenn es sich um eine HTML-E-Mail handelt, HTML-Code enthalten.

Der Parameter für den **E-Mail-Header** (auch Kopfzeilen genannt) kann dazu genutzt werden, einige erweiterte Informationen (z. B. CC- oder BCC-Adressen) anzugeben. Dabei muss darauf geachtet werden, dass jede Header-Angabe mit `\r\n` abgeschlossen sein muss. Eine Header-Angabe setzt sich immer aus dem Namen der Header-Eigenschaft, einem Doppelpunkt, einem Leerzeichen, dem Wert, der Eigenschaft und einem Zeilenumbruch mittels `\r\n` zusammen. Grundsätzlich ist es zu empfehlen, immer einen E-Mail-Header mit anzugeben, in welchem die Eigenschaften `From` (Absender), `To` (Empfänger), `Reply-To` (Antwort-Adresse) und `Content-Type` sowie, wenn benötigt, `Cc` und `Bcc` angegeben werden.

```
1 <?php
2 $kopfzeilen = "From: Max Mustermann <max-mustermann@example.com>\r\n";
3 $kopfzeilen .= "To: Katrin Meyer <katrin-meyer@example.com>\r\n";
4 $kopfzeilen .= "Cc: Tom Dreher <tom-dreher@example.com>\r\n";
5 $kopfzeilen .= "Bcc: Daniel Busch <daniel-busch@example.com>\r\n";
6 $kopfzeilen .= "Reply-To: Max Mustermann <max-mustermann@example.com>\r\n";
7 $kopfzeilen .= "Content-Type: text/plain; charset=utf-8\r\n";
8 ?>
```

**Übrigens:** Die Eigenschaften `From`, `To` und `Reply-To` können so angegeben werden, wie bereits weiter oben für den Empfänger-Parameter beschrieben. Üblicherweise wird im Empfänger-Parameter lediglich die E-Mail-Adresse mitgegeben, wohingegen im E-Mail-Header eine formatierte Angabe (mit Name und E-Mail-Adresse) verwendet wird. Des Weiteren muss darauf geachtet werden, dass die Namen im E-Mail-Header ggf. speziell formatiert werden müssen (z. B. mit BASE64), sofern diese Sonderzeichen enthalten.

Der letzte Parameter, welcher ebenfalls optional ist und dazu genutzt wird, dem **E-Mail-Programm einen Parameter mitzugeben**, wird z. B. bei Verwendung des Programms `sendmail` dazu genutzt mittels der Option `-f`, dem Programm die Absenderadresse zu übergeben. Dadurch weiß das Programm, mit welchem Absender die E-Mail verschickt werden soll. Je nachdem, welches Programm zum Versand der E-Mails auf Ihrem Webserver verwendet wird, können sich die notwendigen Optionen unterscheiden. Zumeist ist es jedoch nicht notwendig, diesen Parameter mit anzugeben.

**Wichtig:** Um den E-Mail-Versand mittels XAMPP zu testen, kann das Programm `mailtodisk` verwendet werden. Dieses ist bereits standardmäßig konfiguriert und führt dazu, dass die E-Mails in den Ordner `mailoutput` des XAMPP-Installationsverzeichnis in Textdateien geschrieben werden. Wer den E-Mail-Versand zusammen mit einem E-Mail-Client testen möchte, kann den in XAMPP integrierten E-Mail-Server Mercury nutzen, welcher zuvor eingerichtet werden muss.

## Text E-Mail

Der Inhalt einer Text E-Mail ist nichts Besonderes, sie enthält eben, wie der Name schon sagt, Text. Einen Zeilenumbruch in einer Textnachricht können Sie mittels `\r\n` erzeugen. Im folgenden Beispiel haben Sie die Möglichkeit, eine E-Mail-Adresse anzugeben und sich mit dem Klick auf den Button, eine Test-Nachricht zuschicken zu lassen:

```
1 <?php
2 if (isset($_POST['email']))
3 {
4     $kopfzeilen = "From: Homepage-Webhilfe Test <test-webseite@homepage-webhilfe.de>\r\n";
5     $kopfzeilen .= "To: ".$_POST['email']."\r\n";
6     $kopfzeilen .= "Reply-To: Homepage-Webhilfe Test <test-webseite@homepage-webhilfe.de>\r\n";
7     $kopfzeilen .= "Content-Type: text/plain; charset=utf-8\r\n";
8
9     $nachricht = "Hallo,\r\n\r\n";
10    $nachricht .= "wenn Sie diese E-Mail in Ihrem Postfach gefunden haben, dann hat der Test-Versand funktioniert.\r\n\r\n";
11    $nachricht .= "Mit freundlichen Grüßen\r\n";
12    $nachricht .= "Homepage-Webhilfe\r\n\r\n";
13    $nachricht .= "Diese E-Mail wurde von ".$_SERVER['REMOTE_ADDR']." versendet!";
14
15    if (mail($_POST['email'], 'Test-E-Mail von Homepage-Webhilfe', $nachricht, $kopfzeilen))
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [E-Mail-Versand](#)

```

16     echo 'E-Mail-Versand erfolgreich!';
17     else
18     echo 'E-Mail-Versand fehlgeschlagen!';
19 }
20 else
21 echo '<form method="post">E-Mail-Adresse: <input type="text" name="email" /> <input type="submit" value="Test E-Mail
senden" /></form>';
22 ?>

```



## HTML E-Mail

Eine HTML E-Mail unterscheidet sich von einer Text E-Mail im Großen und Ganzen nur dadurch, dass es sich beim Nachrichteninhalt um einen HTML-Code handelt. Zusätzlich muss der Wert der `Content-Type`-Eigenschaft angepasst und die Eigenschaft `MIME-Version` mit dem Wert `1.0` angegeben werden. Beim HTML-Code einer E-Mail muss darauf geachtet werden, dass **keine „neuen“ HTML-Elemente sowie CSS-Eigenschaften** verwendet werden sollten. Die meisten E-Mail-Clients haben einen sehr eingeschränkten Rendering-Engine und **sperrn auf Grund von Sicherheitseinstellungen u. U. externe Inhalte** (wie z. B. Bilder). Das Verwenden von aktiven Inhalten (wie JavaScript) sollte vollständig unterlassen werden. Des Weiteren ist das Laden von Stylesheets aus dem Internet (z. B. von der eigenen Webseite) ebenfalls nicht zu empfehlen. Einige Clients ignorieren sogar den Inhalt des `head`-Elements. Die „beste“ Lösung für HTML E-Mails scheint daher das **Layouten und Designen mittels Tabellen** zu sein, da dies von den meisten E-Mail-Clients korrekt dargestellt wird. Bevor Sie also HTML E-Mails produktiv verschicken möchten, sollten Sie sich überlegen, ob sich der Aufwand lohnt und falls ersteres der Fall ist, ausführliche Tests mit unterschiedlichen E-Mail-Clients durchführen, um „sicherzustellen“, dass die E-Mail, bei dem der die E-Mail erhält, richtig angezeigt wird. Das folgende Beispiel funktioniert gleich wie das von oben, verschickt jedoch eine HTML E-Mail mit einfacher Text-Auszeichnung:

```

1 <?php
2 if (isset($_POST['email']))
3 {
4     $kopfzeilen = "From: Homepage-Webhilfe Test <test-webseite@homepage-webhilfe.de>\r\n";
5     $kopfzeilen .= "To: ".$_POST['email']."\r\n";
6     $kopfzeilen .= "Reply-To: Homepage-Webhilfe Test <test-webseite@homepage-webhilfe.de>\r\n";
7     $kopfzeilen .= "MIME-Version: 1.0\r\n";
8     $kopfzeilen .= "Content-Type: text/html; charset=utf-8\r\n";
9
10    $nachricht = "<!doctype html><html><head></head><body>";
11    $nachricht .= "Hallo,<br />";
12    $nachricht .= "wenn Sie diese <i>E-Mail in Ihrem Postfach gefunden</i> haben, dann hat der <b>Test-Versand
funktioniert</b>.<br /><br />";
13    $nachricht .= "Mit freundlichen Grüßen<br />";
14    $nachricht .= "Homepage-Webhilfe<br /><br />";
15    $nachricht .= "Diese E-Mail wurde von ".$_SERVER['REMOTE_ADDR']." versendet!";
16    $nachricht .= "</body></html>";
17
18    if (mail($_POST['email'], 'Test-E-Mail von Homepage-Webhilfe', $nachricht, $kopfzeilen))
19        echo 'E-Mail-Versand erfolgreich!';
20    else
21        echo 'E-Mail-Versand fehlgeschlagen!';
22 }
23 else
24 echo '<form method="post">E-Mail-Adresse: <input type="text" name="email" /> <input type="submit" value="Test E-Mail
senden" /></form>';
25 ?>

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

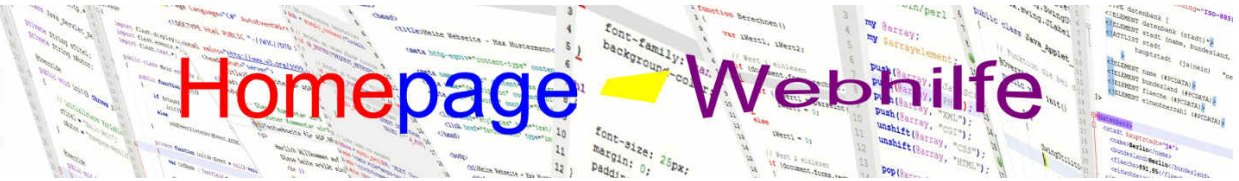
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Datenbankzugriffe](#)

## Datenbankzugriffe

Um mit einer Datenbank zu kommunizieren, bietet PHP grundsätzlich 3 verschiedene Module (APIs): `mysql`, `mysqli` und `PDO`. `mysql` und `mysqli` können lediglich für die **Kommunikation mit MySQL und MariaDB** genutzt werden, nicht jedoch z. B. mit MSSQL. Des Weiteren ist die `mysql`-API seit PHP-Version 5.5 als veraltet gekennzeichnet und wurde in PHP 7 entfernt. Die `PDO`-API kann von verschiedenen Treibern implementiert werden. Dies hat den Vorteil, dass die `PDO`-API für **unterschiedliche Datenbanktypen** genutzt werden kann. Aktuell sind unter anderem Treiber für MySQL, MSSQL, SQLite und PostgreSQL verfügbar. Welche Treiber aktuell verfügbar sind, können Sie in der Ausgabe von `phpinfo()` im Abschnitt „PDO“ sehen. Die `mysqli`- und `PDO`-API haben beide eine **objektorientierte Schnittstelle**. Auf Grund der Vorteile der `PDO`-API in Bezug auf die Unterstützung unterschiedlicher Datenbanktypen werden wir in diesem Thema ausschließlich die `PDO`-API behandeln und empfehlen Ihnen, diese auch in Ihrem Projekt zu verwenden.

**Wichtig:** Für dieses Thema sind Vorkenntnisse in der Datenbanksprache SQL notwendig. Falls Sie über diese nicht verfügen, empfehlen wir Ihnen, zuerst den [SQL-Crashkurs](#) zu lesen.

**Inhalt dieser Seite:**

1. Verbindungsaufbau
2. SQL-Befehl ausführen
3. Datensätze abrufen
4. SQL-Injektion

## Verbindungsaufbau

Um mit einer Datenbank eine Verbindung aufzubauen, müssen Sie ein Objekt der Klasse `PDO` instanzieren. Als Parameter übergeben Sie den sogenannten DSN (Data Source Name), welcher Informationen über die Verbindung enthält, und optional einen Benutzernamen, ein Passwort und ein assoziatives Array mit Optionen. Die **Optionen** können im Nachhinein noch mit der Methode `setAttribute()` gesetzt werden. Dieser wird das Attribut (dafür werden Konstanten der `PDO`-Klasse genutzt) und ein Wert (evtl. auch über Konstanten) übergeben. Ein Aufruf der Methode `setAttribute()` ist natürlich erst nach der Objektinstanziierung möglich.

Jede DSN-Zeichenkette verfügt am Anfang über eine **Kennung** für das jeweilige Datenbanksystem (z. B. `mysql` für MySQL, `pgsql` für PostgreSQL und `sqlsrv` für Microsoft SQL Server). Anschließend folgen ein Doppelpunkt und die einzelnen **Eigenschaften**, welche sich aus einem Namen, einem Gleichheitszeichen und dem Wert zusammensetzen. Mehrere Eigenschaften werden dabei mit Semikolon getrennt. Die Eigenschaften unterscheiden sich je nach Datenbanktyp. In der DSN-Zeichenkette kann neben der **Adresse des Datenbankservers** auch die **zu verwendende Datenbank** selektiert werden. Das folgende Beispiel zeigt eine DSN für MySQL. Hier wird neben dem Host und der Datenbank auch noch ein **Zeichensatz** festgelegt.

```
1 | mysql:host=localhost;dbname=meineDatenbank;charset=utf8
```

Der DSN für PostgreSQL-Datenbanksysteme sieht fast gleich aus. Die Angabe des Zeichensatzes ist hier jedoch nicht möglich. Der Zeichensatz kann daher nur im Nachhinein über das Ausführen des SQL-Statements `SET NAMES 'UTF-8'` gesetzt werden.

```
1 | pgsql:host=localhost;dbname=meineDatenbank
```

Zuletzt wollen wir noch auf den DSN für Microsoft SQL Server eingehen. Hier heißen, wie es bei Microsoft oft der Fall, die Eigenschaften anders. Genauso, wie bei PostgreSQL-Datenbanken auch, ist es hier nicht möglich, den Zeichensatz direkt zu setzen. Daher ist auch hier auf die Verwendung des oben genannten SQL-Statements zurückzugreifen.

```
1 | sqlsrv:Server=localhost;Database=meineDatenbank
```

Um eine **Datenbankverbindung wieder zu trennen**, muss die Variable, in welcher die Instanz der Datenbank gespeichert wurde, lediglich auf `null` gesetzt werden. Die Ressourcen und die Verbindung zur Datenbank werden dann durch PHP automatisch getrennt und freigegeben. Wird die Datenbank-Verbindung nicht explizit getrennt, so geschieht dies **automatisch beim Skriptende**. Das folgende Beispiel zeigt einen vollständigen Code zum Aufbau einer Datenbankverbindung zu einem MySQL-Server.

```
1 | <?php
2 | try
3 | {
4 |     $db = new PDO('mysql:host=localhost;dbname=hwhptest;charset=utf8', 'root', '');
5 |
6 |     echo 'Verbindung aufgebaut';
7 |
8 |     $db = null;
9 | }
10 | catch (PDOException $ex)
11 | {
12 |     echo 'Meldung: '.$ex->getMessage();
13 | }
14 | ?>
```

**Wichtig:** Schlägt die Verbindung zum Datenbank-Server fehl, so wird eine Ausnahme der Klasse `PDOException` geworfen. Diese sollte unbedingt abgefangen werden, da andernfalls ein fataler PHP-Fehler ausgelöst wird, was einen Skriptabbruch auslöst. Zudem werden u. U. sensible Informationen wie Benutzername und Passwort ausgegeben. Besteht die Verbindung erst einmal, so werden von der `PDO`-Klasse standardmäßig keine Exceptions mehr geworfen, sondern nur noch Fehlercodes gesetzt, welche mittels der Funktion `errorCode()` abgerufen werden können. Erweiterte Fehlerinformationen können mittels der Methode `errorInfo()` abgerufen werden. Das Verhalten bei Fehlern kann jedoch mit der Option `ATTR_ERRMODE` geändert werden.

## SQL-Befehl ausführen

Um einen SQL-Befehl (zumeist als SQL-Statement bezeichnet) auszuführen, benötigen wir die Methode `query()`. Als Parameter wird dieser das SQL-Statement übergeben. Wird der Befehl erfolgreich ausgeführt, so wird eine Instanz der Klasse `PDOStatement` zurückgegeben. Auf deren Verwendung gehen wir später ein. Schlägt das Ausführen des Statements fehl, so wird `false` zurückgegeben.

```
1 | <?php
2 | try
3 | {
4 |     $db = new PDO('mysql:host=localhost;dbname=hwhptest;charset=utf8', 'root', '');
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » PHP » Datenbankzugriffe

```

5
6     if ($db->query('UPDATE `php-tutorial-kunden` SET `Nachname`=\'Schulze\' WHERE `Nummer`=6') !== false)
7         echo 'Befehl ausgeführt!';
8     else
9     {
10        echo '<pre>';
11        print_r($db->errorInfo());
12        echo '</pre>';
13    }
14
15    $db = null;
16 }
17 catch (PDOException $ex)
18 {
19     echo 'Meldung: '.$ex->getMessage();
20 }
21 ?>

```



## Datensätze abrufen

Wollen Sie Datensätze aus einer Datenbank abrufen, so nutzen wir ebenfalls die Methode `query()`. Damit wird jedoch nur das Statement ausgeführt und noch keine Daten abgeholt. Deshalb wollen wir uns jetzt mit dem `PDOStatement`-Objekt, welches von der Methode `query()` zurückgegeben wird, beschäftigen. Die Klasse `PDOStatement` bietet 3 wichtige Funktionen, um die Daten abzuholen: `fetch()`, `fetchAll()` und `fetchColumn()`. Die Funktionen `fetch()` und `fetchAll()` holen **alle Spalten** (bzw. alle die angegeben wurden) ab und geben diese zurück. Die Art wie diese zurückgegeben wurden, kann der Funktion als Parameter mitgegeben werden. Hierbei sind `FETCH_NUM` für ein indiziertes Array, `FETCH_ASSOC` für ein assoziatives Array und `FETCH_OBJ` für ein anonymes Objekt zu erwähnen. `fetch()` gibt das Ergebnis eines einzelnen Datensatzes zurück. `fetchAll()` hingegen gibt ein Array mit allen Datensätzen zurück. Im Fehlerfall, oder wenn keine Datensätze (mehr) verfügbar sind, wird `false` zurückgegeben. Die Funktion `fetchColumn()` gibt den Wert einer **einzelnen Spalte eines Datensatzes** zurück. Als Parameter wird der Funktion `fetchColumn()` die Spaltennummer übergeben. Es gilt jedoch zu beachten, dass nach dem Aufruf von `fetchColumn()` der Zeiger auf die nächste Zeile gesetzt wird und somit keine weitere Spalte dieses Datensatzes mehr abgerufen werden kann.

```

1 <?php
2 try
3 {
4     $db = new PDO('mysql:host=localhost;dbname=hwhptest;charset=utf8', 'root', '');
5
6     $result = $db->query('SELECT * FROM `php-tutorial-kunden`');
7     if ($result !== false)
8     {
9         echo '<table>';
10        echo '<tr><th>Nummer</th><th>Vorname</th><th>Nachname</th></tr>';
11        while (($row = $result->fetch(PDO::FETCH_ASSOC)) !== false)
12            echo '<tr><td>'.$row['Nummer'].'</td><td>'.$row['Vorname'].'</td><td>'.$row['Nachname'].'</td></tr>';
13    }
14    else
15    {
16        echo '<pre>';
17        print_r($db->errorInfo());
18        echo '</pre>';
19    }
20
21    $db = null;
22 }
23 catch (PDOException $ex)
24 {
25     echo 'Meldung: '.$ex->getMessage();
26 }
27 ?>

```



## SQL-Injektion

Unter SQL-Injektion (engl. *SQL Injection*) versteht man das **gezielte Manipulieren von SQL-Statements**, um somit Daten zu verändern, zu löschen, auszuspähen oder anderen Unfug auf dem Server anzustellen.

Als erstes wollen wir daher kurz erklären, wie eine solche SQL-Injektion funktioniert. Meistens enthalten die ausgeführten SQL-Statements Daten, welche von URL-Parametern oder Formulardaten stammen. Hierzu ein Beispiel:

**Aufruf:**

```
1 | index.php?seite=12
```

**Resultierendes SQL:**

```
1 | SELECT * FROM seiten WHERE id=12
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Datenbankzugriffe](#)

Der Aufruf kann nun so manipuliert werden, dass wir z. B. einen `DROP TABLE`-Befehl ausführen. Dies sieht dann z. B. so aus:

#### Aufruf:

```
1 | index.php?seite=12;DROP+TABLE+seiten
```

#### Resultierendes SQL:

```
1 | SELECT * FROM seiten WHERE id=12;DROP TABLE seiten
```

Doch was kann man dagegen tun? Grundsätzlich muss darauf geachtet werden, dass keine Daten direkt an die Datenbank übergeben werden. Die `PDO`-Klasse stellt hier die Methode `quote()` dar, um Sonderzeichen zu maskieren und sich somit vor SQL-Injektionen zu schützen. Hierzu ein Beispiel:

```
1 | <?php
2 | $sql = 'SELECT * FROM seiten WHERE id=' . $db->quote($_GET['seite']);
3 | ?>
```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [PHP](#) » [Abschluss](#)

## Abschluss

Nun haben Sie es geschafft und haben dieses PHP-Tutorial vollständig abgearbeitet. Sie haben in diesem Tutorial die **Grundlagen der Sprache** (Syntax, Verzweigungen, Schleifen, Funktionen), die Objektorientierung, die Fehlerbehandlung, den Dateizugriff, den E-Mail-Versand, den Datenbankzugriff sowie **weitere spannende Themen** kennengelernt. Mit Hilfe von PHP ist es Ihnen nun möglich, **einfache aber auch komplexe Websites zu erstellen**, wie z. B. das Laden von Daten aus einer Datenbank.

PHP bietet natürlich viel mehr Funktionen und Objekte, als wir hier in diesem Tutorial beschrieben haben. Als Handbuch und Nachschlagewerk empfehlen wir hier die [offizielle PHP-Dokumentation](#) von der „The PHP Group“.

Neben PHP gibt es noch einige **weitere Technologien**, die dazu verwendet werden können, dynamische Webseiten zu erstellen. Hierzu zählen [Perl](#), [ASP.NET](#) und [Java EE](#), zu welchen wir auf dieser Website ebenfalls Tutorials anbieten.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

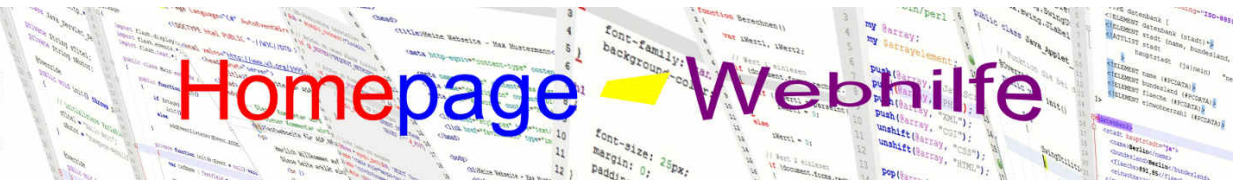
Java EE

XML

# E-Book

## Perl





Sie befinden sich hier: [Homepage-Webhilfe](#) » [Perl](#) » [Einführung](#)

## Einführung



Perl ist eine **freie und plattformunabhängige Skriptsprache**, die dazu genutzt werden kann, **dynamische Webseiten zu erstellen**. Hierfür kommuniziert der Webserver mittels der **CGI-Schnittstelle** mit dem Perl-Interpreter.

Perl wurde 1987 von Larry Wall entworfen. Die Sprache war ursprünglich zur Manipulation von Textdateien sowie zur Steuerung von anderen Programmen gedacht. Mit der Entstehung und Entwicklung des World Wide Webs wurde **Perl verstärkt auf Webservern eingesetzt**, um somit Webseiten dynamisch erzeugen zu können. Dabei war es Perl schon damals möglich, auch mit anderen Ressourcen wie Dateien und Datenbanken zu arbeiten. Auch wenn PHP in der Zwischenzeit als eindeutiger Marktführer für serverseitige Webanwendungen identifizierbar ist, werden auf vielen Webservern noch immer Perl-Skripte eingesetzt.

Vom Syntax ist Perl an die menschliche Sprache angelehnt, weshalb die **Sprache auch als einfach erlernbar gilt**. In diesem Thema wollen wir uns jedoch nicht mit den Grundlagen der Sprache Perl, sondern vielmehr **mit dem CGI-Modul von Perl beschäftigen**, um Ihnen die Erzeugung von dynamischen Webseiten sowie das Verarbeiten von übergebenen URL-Parametern und Formulardaten näher zu bringen. Für einen Schnelleinstieg in Perl können Sie unseren [Crashkurs](#) besuchen.

**Inhalt dieser Seite:**

1. Webserver
2. Erstes Skript

## Webserver

Um Perl-Skripte auszuführen, wird auf dem Webserver in erster Linie ein **Perl-Interpreter benötigt**. Dieser kann auf der [Perl-Website](#) heruntergeladen werden. In den meisten Hosting-Angeboten ist ein Perl-Interpreter bereits vom Provider vorinstalliert (da kein Root-Zugriff auf den Server durch den Kunden vorhanden ist), welcher direkt genutzt werden kann. In dem Software-Paket XAMPP, welches unter Webentwicklern sehr beliebt ist, ist Perl bereits enthalten.

Bevor wir uns nun das erste Perl-Skript anschauen, wollen wir uns mit der **Funktionsweise der CGI-Schnittstelle** beschäftigen: Bei CGI (Common Gateway Interface) handelt es sich um eine Schnittstelle, die zum **Austausch von Daten zwischen Webserver und anderen Programmen** (in diesem Fall der Perl-Interpreter) verwendet wird. Dabei stellt der Webserver eine sogenannte **Laufzeitumgebung** zur Verfügung, in welcher das Programm ausgeführt wird und neben **Umgebungsvariablen und Ein- und Ausgabedatenströme** (`stdin` und `stdout`) zur Verfügung gestellt werden. Die Umgebungsvariablen enthalten Informationen zum Server, zum Skript, zum Anfrager, zur Anfrage und zur HTTP-Verbindung. Die Ein- und Ausgabedatenströme dienen dazu, die URL-Parameter und Formulardaten vom Browser zur CGI-Applikation und die Ausgaben von der CGI-Applikation zurück zum Browser zu transportieren. In dem Bild auf der rechten Seite sehen Sie, wie die Anfrage einer Seite von einem Webserver aussieht.



## Erstes Skript

Nun wollen wir uns dem ersten Perl-Skript widmen. An sich enthält die Datei nichts Besonderes: die Shebang-Zeile, welche den Pfad zum Perl-Interpreter enthält, die `use strict;`-Anweisung und eine Ausgabe im Heredoc-Stil. Bei der Ausgabe wird das HTTP-Headerfeld `Content-Type` und das HTML-Dokument ausgegeben. Es gilt dabei, die Leerzeile zwischen HTTP-Header und dem Inhalt zu beachten, welche nicht fehlen darf. Natürlich kann die Ausgabe auch mit einzelnen `print`-Befehlen erfolgen. Der folgende Code, oder generell ein Perl-Skript, wird zumeist in einer Datei mit der Endung `.pl` oder `.cgi` gespeichert.

```

1  #!/usr/bin/perl -w
2
3  use strict;
4
5  print <<END;
6  Content-Type: text/html;charset=UTF-8
7
8  <!doctype html>
9  <html>
10     <head>
11         <title>Erstes Skript - Perl Code-Beispiel</title>
12
13         <meta charset="utf-8" />
14
15         <meta name="robots" content="noindex,nofollow" />
16         <meta name="publisher" content="Homepage-Webhilfe" />
17     </head>
18
19     <body>
20         <h1>Hallo Welt</h1>
21         <p>Dies ist das erste Perl-Skript! Abgesehen von der Ausgabe enthält dieses Skript keinen Code.</p>
22     </body>
23 </html>
24 END
    
```

**Wichtig:** Auf einigen Webservern bzw. bei einigen Providern können bzw. dürfen CGI-Skripte nur innerhalb des Ordners `/cgi-bin/` ausgeführt werden. Dies kann auf einem Apache-Webserver mittels der Option `ExecCGI` geändert werden. Einige Provider lassen das Setzen dieser Option aber u. U. nicht zu.

Bei Verwendung von XAMPP muss zudem die Shebang-Zeile angepasst werden: `#!/C:\xampp\perl\bin\perl -w` (bei einer Standardinstallation). Die Option `-w` bewirkt, wie Ihnen vermutlich bekannt ist, die Ausgabe von Warnungen.

Im Gegensatz zu PHP können in Perl **Skript-Blöcke nicht im HTML-Code eingebettet** werden. Perl wird also nicht dazu verwendet, einen Teil einer HTML-Seite dynamisch zu erzeugen, sondern **erzeugt immer die komplette Ausgabedatei**. Hierfür kann natürlich statischer HTML-Code einfach mittels eines `print`-Befehls ausgegeben werden. Das obige Beispiel enthält keine dynamischen Bestandteile, sondern lediglich einen statischen HTML-Code, welcher mittels des `print`-Befehls ausgegeben wird. Perl muss jedoch neben dem HTML-Code auch noch einen Teil des HTTP-Headers ausgeben. Dieser kann gegebenenfalls leer sein,

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Perl](#) » [Einführung](#)

jedoch muss zwischen Header und HTML-Code immer eine Leerzeile vorhanden sein. In dem Fall, dass kein Header mitgeschickt werden soll, bedeutet dies also, dass eine leere Zeile vor dem HTML-Code notiert werden muss. Grundsätzlich ist es jedoch zu empfehlen, die `Content-Type`-Eigenschaft mitzusenden (wie auch im Beispiel oben).

In den nächsten Beispielen (ausgenommen die vom [Thema Umgebungsinformationen](#)) werden wir uns mit dem **CGI-Modul** von Perl beschäftigen. Dieses wird mit `use CGI;` eingebunden. Anschließend muss noch ein `CGI`-Objekt instanziiert werden. Aus der daraus entstandenen Instanz werden wir später unsere Funktionen aufrufen.

```
1 use CGI;
2
3 my $cgi = new CGI();
```

Enthält Ihr Perl-Skript einen Fehler, so teilt der Perl-Interpreter dies an den Webserver mit, welcher darauf eine Seite zur Anzeige eines **500 Internal Server Errors** an den Browser schickt bzw. auf ein dazugehöriges `ErrorDocument` weiterleitet. Ein solches Verhalten ist gerade für Entwicklungszwecke nicht sehr hilfreich, deshalb empfiehlt es sich, die Funktion `fatalToBrowser()` aus dem Untermodul `Carp` des `CGI`-Moduls zu importieren. Dadurch werden **detaillierte Fehlermeldungen** (mit Zeilenangaben) an den Browser geschickt. Nachdem Sie die Entwicklung eines Skripts abgeschlossen haben, sollten Sie die Importierungs-Anweisung wieder entfernen, da wenn Ihr Skript doch noch einen Fehler enthalten sollte, u. U. sensible Informationen ausgegeben werden, die einen Besucher nichts angehen. Der Import von `fatalToBrowser()` sieht wie folgt aus:

```
1 use CGI::Carp qw(fatalToBrowser);
```

**Übrigens:** Alle unsere Perl-Beispiele enthalten die Anweisung `use strict;`, um „unsichere Konstrukte“ nicht zuzulassen.

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Perl](#) » [Ausgaben](#)

## Ausgaben

Das `CGI`-Modul stellt einige **Funktionen zur Ausgabe von HTTP-Headern und HTML-Code** zur Verfügung. Diese Funktionen wollen wir uns in diesem Thema etwas genauer anschauen.

### Inhalt dieser Seite:

1. Header
2. HTML-Code

### Header

Mit der Funktion `header()` einer `CGI`-Instanz ist es möglich, einen **HTTP-Header** auszugeben. Der Funktion kann eine Hash-Referenz übergeben werden. Als Namen für die Hash-Elemente stehen u. a. `-status`, `-type` und `-expires` zur Verfügung. Mit `-status` kann die **Statusmeldung** gesetzt werden, welche den Statuscode und den Statustext enthält. Die Statusmeldung befindet sich in der ersten Zeile in einem HTTP-Antwort-Header. Die Standard-Statusmeldung ist `200 OK`, die nicht explizit gesetzt werden muss. Mit `-type` wird das HTTP-Headerfeld `Content-Type` festgelegt. Dort wird der sogenannte **MIME-Typ und ggf. die Zeichenkodierung** (engl. `charset`) festgelegt. Mittels `-expires` kann der **Ablaufzeitpunkt bzw. die Gültigkeitsdauer** des Dokuments festgelegt werden. Dies hat zur Folge, dass der Browser, wenn er das Dokument zum ersten Mal anfragt, dieses für den angegebenen Zeitraum im Cache-Speicher ablegt. Wird das Dokument dann innerhalb dieses Zeitraums erneut geladen, so wird dieses aus dem Cache geladen. Wird es hingegen nach dem Ablaufzeitraum aufgerufen, so wird das Dokument erneut vom Server geholt. Als Wert für `-expires` sind absolute Zeitangaben oder Zahlen in Kombination mit Kürzeln (`s` für Sekunde, `m` für Minute, `h` für Stunde, `F` für Tage, `M` für Monate und `y` für Jahr) möglich. Wird eine Zahl mit einem Kürzel verwendet, so muss der Zahl noch ein Pluszeichen `+` vorangestellt werden.

```

1  #!/usr/bin/perl -w
2
3  use strict;
4  use CGI;
5
6  my $cgi = new CGI();
7
8  # Header-Ausgabe
9  print $cgi->header({
10     -status => '200 OK',
11     -expires => '+1h',
12     -type => 'text/html; charset=UTF-8'
13 });
14
15 # HTML-Ausgabe
16 print <<END;
17 <!doctype html>
18 <html>
19     <head>
20         <title>Ausgabe Header - Perl Code-Beispiel</title>
21
22         <meta charset="utf-8" />
23
24         <meta name="robots" content="noindex,nofollow" />
25         <meta name="publisher" content="Homepage-Webhilfe" />
26     </head>
27
28     <body>
29         <p>Hier steht der Inhalt ...</p>
30     </body>
31 </html>
32 END

```



### HTML-Code

Bisher haben wir HTML-Code mittels `print`-Befehlen ausgegeben. Doch im `CGI`-Modul gibt es auch einige Funktionen, die es ermöglichen, **HTML-Code einfacher zu erzeugen**. Das `CGI`-Modul erzeugt standardmäßig einen XHTML 1.0 Transitional konformen Code. Über Umwege können auch andere Dokumententypen (z. B. HTML 4.01) gewählt werden. Das Erzeugen von HTML5-Code ist jedoch nicht problemlos möglich, da HTML5 über keine DTD verfügt.

Als erstes wollen wir uns damit beschäftigen, die **HTML-Grundstruktur** sowie den Inhalt innerhalb des `head`-Elements zu erzeugen. Hierfür nutzen wir die Funktionen `start_html()` und `end_html()`. Der Funktion `start_html()` kann eine Hash-Referenz übergeben werden. Mittels `-title` kann dort der **Seitentitel** festgelegt werden. Die Elementnamen `-style` und `-script` erlauben das Einbinden von Stylesheet- (`link`-Element mit dem Wert `stylesheet` im `rel`-Attribut) und Skript-Dateien (`script`-Element). Über `-encoding` kann die **Zeichenkodierung** festgelegt werden. Diese wird im ausgegebenen HTML-Code in einem `meta`-Element, welches lediglich bei älteren HTML-Versionen und XHTML verwendet wird, angegeben. `-lang` erlaubt das Festlegen der **Sprache**. Diese wird bei XHTML im `html`-Element über das Attribut `lang` festgelegt. Weitere **Meta-Angaben** können mit `-meta` spezifiziert werden. Hier ein kurzer Beispielfragment:

```

1  #!/usr/bin/perl -w
2
3  use strict;
4  use CGI;
5
6  my $cgi = new CGI();
7
8  print $cgi->start_html({
9     -title => 'Meine Perl-Webseite',
10    -encoding => 'UTF-8',

```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » Perl » Ausgaben

```

11     -lang => 'de-DE',
12     -meta => {
13         'publisher' => 'Max Mustermann'
14     }
15 });
16 print $cgi->end_html();

```

Der obige Code erzeugt folgenden HTML-Code:

```

1 <!DOCTYPE html
2     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml" lang="de-DE" xml:lang="de-DE">
5 <head>
6 <title>Meine Perl-Webseite</title>
7 <meta name="publisher" content="Max Mustermann" />
8 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
9 </head>
10 <body>
11
12 </body>
13 </html>

```

Perl ist mit Hilfe des CGI-Moduls in der Lage, so ziemlich jedes HTML-Element auszugeben. Dafür stehen die Funktionen `x()`, `start_x()` und `end_x()` zur Verfügung, wovon `x` durch den HTML-Elementnamen ersetzt wird (z. B. `div()`, `start_div()` und `end_div()`). Den Funktionen `x()` und `start_x()` können eine Hash-Referenz übergeben werden. Der Funktion `x()` kann des Weiteren noch der Inhalt des Elements übergeben werden. Die Hash-Referenz wird dazu verwendet, die **Attribute des Elements** festzulegen. Dafür wird als Elementname eines Hash-Eintrags der Name des Attributs mit einem vorangestellten Bindestrich notiert (z. B. `-style` für das `style`-Attribut). Die Funktion `x()` gibt das komplette Element (Start-Tag ggf. mit Attributen, Inhalt und End-Tag) aus, wohingegen die Funktion `start_x()` lediglich den Start-Tag (ggf. mit Attributen) ausgibt. `end_x()` gibt lediglich den End-Tag aus. Bei kleineren bzw. „einfacheren“ Ausgaben wird zumeist die Funktion `x()` verwendet. Die Funktionen `start_x()` und `end_x()` werden immer in Kombination verwendet und dienen zur Ausgabe größerer und komplexerer Inhalte, da es hier möglich ist, zwischen den Funktionsaufrufen anderen Perl-Code wie z. B. Verzweigungen oder Schleifen zu notieren. Mit Hilfe der Funktion `comment()` ist es möglich, einen HTML-Kommentar zu erzeugen. Als Parameter wird hier der Text für den Kommentar übergeben. Das folgende Beispiel zeigt die Verwendung verschiedener Funktionen zur Ausgabe von HTML-Code mittels des CGI-Moduls:

```

1 #!/usr/bin/perl -w
2
3 use strict;
4 use CGI;
5
6 my $cgi = new CGI();
7
8 # Header-Ausgabe
9 print $cgi->header({
10     -type => 'text/html; charset=UTF-8'
11 });
12
13 # HTML-Ausgabe
14 print $cgi->start_html({
15     -title => 'Ausgabe HTML-Code - Perl Code-Beispiel',
16     -encoding => 'UTF-8',
17     -lang => 'de-DE',
18     -meta => {
19         'robots' => 'noindex,nofollow',
20         'publisher' => 'Homepage-Webhilfe'
21     }
22 });
23 print $cgi->comment('Ab hier kommt der Inhalt ...');
24 print $cgi->h1('Ausgabe HTML-Code');
25 print $cgi->img({
26     -style => 'float: left;',
27     -src => '/Bilder/Logo/Logo.jpg',
28     -width => 100,
29     -height => 100,
30     -alt => 'Homepage-Webhilfe Logo',
31     -title => 'Homepage-Webhilfe'
32 });
33 print $cgi->p({
34     -style => "width: 500px; text-align: justify;"
35 }, '...');
36 print $cgi->br();
37 print $cgi->i('Copyright 2016 by ' . $cgi->a({
38     -href => 'https://www.homepage-webhilfe.de',
39     -target => '_blank'
40 }, 'Homepage-Webhilfe')
41 );
42 print $cgi->end_html();

```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

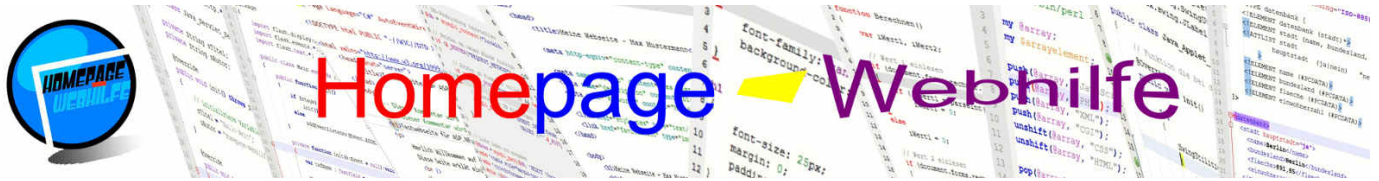
## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Perl](#) » [Ausgaben](#)

**Wichtig:** Auf Grund von Überschneidungen mit dem Wortschatz von Perl müssen bei der Erzeugung der HTML-Elemente `link`, `select`, `sub` und `tr` die Funktionsnamen mit einem Großbuchstaben geschrieben werden, also z. B. `Link()` an Stelle von `link()`. Dies gilt auch für die Funktionen `start_x()` und `end_x()` bei Verwendung einer der genannten Elemente.

**Übrigens:** Es ist natürlich auch möglich, Ausgaben mittels der Funktionen des `CGI`-Moduls mit einfachen direkten Ausgaben (wie im Einleitungsbeispiel) zu mischen.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Perl](#) » [Umgebungsinformationen](#)

## Umgebungsinformationen

Wie bereits im [Einführungs-Thema](#) erklärt, werden vom Webserver über die CGI-Schnittstelle dem Perl-Interpreter Umgebungsinformationen bereitgestellt. Dazu zählen Server-, Skript-, Remote-, Anfrage- und HTTP-Informationen, welche wir innerhalb dieses Themas noch genauer erklären werden. Alle Umgebungsvariablen können über den Hash `%ENV` abgerufen werden (bzw. in der Form von `$ENV{'VARIABLEN_NAME'}` für den Wert einer einzelnen Umgebungsvariable).

### Inhalt dieser Seite:

1. Server-Informationen
2. Skript-Informationen
3. Remote-Informationen
4. Anfrage-Informationen
5. HTTP-Informationen

## Server-Informationen

Server-Informationen beginnen mit dem Variablennamen `SERVER_` und enthalten Netzwerk-, Software- und Administrations-Informationen in Bezug auf den Server. Die folgende Tabelle zeigt eine Übersicht über Server-Informationen:

<b>SERVER_NAME</b>	Hostname des Servers.
<b>SERVER_ADDR</b>	IP-Adresse des Servers.
<b>SERVER_PORT</b>	Port des Webserver (z. B. 80 für HTTP oder 443 für HTTPS).
<b>SERVER_PROTOCOL</b>	Verwendetes Protokoll und Protokoll-Version (z. B. HTTP/1.1).
<b>SERVER_ADMIN</b>	Name und / oder E-Mail-Adresse des eingetragenen Server-Administrators.
<b>SERVER_SOFTWARE</b>	Verwendete Webserver-Software und dessen Version.

Das folgende Beispiel gibt einige der in der Tabelle enthaltenen Variablen aus:

```

1  #!/usr/bin/perl -w
2
3  use strict;
4
5  print <<END;
6  Content-Type: text/html;charset=UTF-8
7
8  <!doctype html>
9  <html>
10     <head>
11         <title>Server-Informationen - Perl Code-Beispiel</title>
12
13         <meta charset="utf-8" />
14
15         <meta name="robots" content="noindex,nofollow" />
16         <meta name="publisher" content="Homepage-Webhilfe" />
17     </head>
18
19     <body>
20         <dl>
21             <dt>SERVER_NAME</dt>
22             <dd>$ENV{'SERVER_NAME'}</dd>
23             <dt>SERVER_ADDR</dt>
24             <dd>$ENV{'SERVER_ADDR'}</dd>
25             <dt>SERVER_PORT</dt>
26             <dd>$ENV{'SERVER_PORT'}</dd>
27             <dt>SERVER_PROTOCOL</dt>
28             <dd>$ENV{'SERVER_PROTOCOL'}</dd>
29         </dl>
30     </body>
31 </html>
32 END

```



## Skript-Informationen

Mit den Skript-Informationen ist es möglich, den absoluten realen Pfad (also den Pfad auf dem Server-Dateisystem, `SCRIPT_FILENAME`) und den Pfad auf der Website (auch als HTTP-Pfad bezeichnet, `SCRIPT_NAME`) abzurufen. Aus Sicherheitsgründen zeigen wir im folgenden Beispiel nur den HTTP-Pfad an:

```

1  #!/usr/bin/perl -w
2
3  use strict;
4
5  print <<END;
6  Content-Type: text/html;charset=UTF-8
7
8  <!doctype html>
9  <html>
10     <head>
11         <title>Skript-Informationen - Perl Code-Beispiel</title>
12

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Perl](#) » [Umgebungsinformationen](#)

```

13     <meta charset="utf-8" />
14
15     <meta name="robots" content="noindex,nofollow" />
16     <meta name="publisher" content="Homepage-Webhilfe" />
17 </head>
18
19 <body>
20     <dl>
21         <dt>SCRIPT_NAME</dt>
22         <dd>${ENV{'SCRIPT_NAME'}}</dd>
23     </dl>
24 </body>
25 </html>
26 END

```



## Remote-Informationen

Remote-Informationen geben Auskunft über den „entfernten Computer“, also den Computer, der die Website aufgerufen hat. In der folgenden Tabelle sehen Sie die verfügbaren Variablen und deren Bedeutung. Im darunter stehenden Beispielpcode werden die meisten der in der Tabelle aufgelisteten Variablen ausgegeben.

<b>REMOTE_HOST</b>	Hostname des entfernten Computers (kann u. U. leer sein).
<b>REMOTE_ADDR</b>	IP-Adresse des entfernten Computers.
<b>REMOTE_PORT</b>	Lokaler Port des entfernten Computers (meistens zwischen 32768 - 65535).
<b>REMOTE_USER</b>	Benutzername des auf dem Server angemeldeten Benutzers des entfernten Computers (nur gesetzt, wenn die Seite in einem geschützten Bereich liegt).

```

1  #!/usr/bin/perl -w
2
3  use strict;
4
5  print <<END;
6  Content-Type: text/html;charset=UTF-8
7
8  <!DOCTYPE html>
9  <html>
10     <head>
11         <title>Remote-Informationen - Perl Code-Beispiel</title>
12
13         <meta charset="utf-8" />
14
15         <meta name="robots" content="noindex,nofollow" />
16         <meta name="publisher" content="Homepage-Webhilfe" />
17     </head>
18
19     <body>
20         <dl>
21             <dt>REMOTE_HOST</dt>
22             <dd>${ENV{'REMOTE_HOST'}}</dd>
23             <dt>REMOTE_ADDR</dt>
24             <dd>${ENV{'REMOTE_ADDR'}}</dd>
25             <dt>REMOTE_PORT</dt>
26             <dd>${ENV{'REMOTE_PORT'}}</dd>
27         </dl>
28     </body>
29 </html>
30 END

```



## Anfrage-Informationen

Informationen über die Anfrage befinden sich in den Variablen `REQUEST_METHOD` und `REQUEST_URI`. `REQUEST_URI` enthält die sogenannte URI, welche aus Pfad (mit Dateiname), Anfrage-Daten (URL-Parameter) und einem Fragment (normalerweise dme Anker) bestehen. `REQUEST_METHOD` enthält die Methode, mit welcher die Seite aufgerufen wurde. Dabei handelt es sich normalerweise um GET oder bei der Übertragung von Formulardaten auch oft um POST.

```

1  #!/usr/bin/perl -w
2
3  use strict;
4
5  print <<END;
6  Content-Type: text/html;charset=UTF-8
7
8  <!DOCTYPE html>
9  <html>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Perl](#) » [Umgebungsinformationen](#)

```

10     <head>
11         <title>Anfrage-Informationen - Perl Code-Beispiel</title>
12
13         <meta charset="utf-8" />
14
15         <meta name="robots" content="noindex,nofollow" />
16         <meta name="publisher" content="Homepage-Webhilfe" />
17     </head>
18
19     <body>
20         <dl>
21             <dt>REQUEST_METHOD</dt>
22             <dd>${ENV{'REQUEST_METHOD'}}</dd>
23             <dt>REQUEST_URI</dt>
24             <dd>${ENV{'REQUEST_URI'}}</dd>
25         </dl>
26     </body>
27 </html>
28 END

```



## HTTP-Informationen

Umgebungsvariablen zu HTTP-Informationen beginnen mit `HTTP_`. Die darin enthaltenen Informationen kommen größtenteils aus dem HTTP-Anfrage-Header. Die folgende Tabelle listet die verschiedenen HTTP-Umgebungsinformationen auf:

<b>HTTP_HOST</b>	Hostname oder IP-Adresse der angefragten Seite.
<b>HTTP_REFERER</b>	URI der zuvor aufgerufenen Webseite (kann zur Ermittlung des Besucher-Verlaufs verwendet werden).
<b>HTTP_USER_AGENT</b>	User-Agent des entfernten Computers (daraus kann zumeist der Browser und das Betriebssystem ermittelt werden).

Und hier auch noch ein Beispiel dazu:

```

1  #!/usr/bin/perl -w
2
3  use strict;
4
5  print <<END;
6  Content-Type: text/html;charset=UTF-8
7
8  <!doctype html>
9  <html>
10     <head>
11         <title>HTTP-Informationen - Perl Code-Beispiel</title>
12
13         <meta charset="utf-8" />
14
15         <meta name="robots" content="noindex,nofollow" />
16         <meta name="publisher" content="Homepage-Webhilfe" />
17     </head>
18
19     <body>
20         <dl>
21             <dt>HTTP_HOST</dt>
22             <dd>${ENV{'HTTP_HOST'}}</dd>
23             <dt>HTTP_REFERER</dt>
24             <dd>${ENV{'HTTP_REFERER'}}</dd>
25             <dt>HTTP_USER_AGENT</dt>
26             <dd>${ENV{'HTTP_USER_AGENT'}}</dd>
27         </dl>
28     </body>
29 </html>
30 END

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » Perl » Formulardaten

## Formulardaten

Bei Formulardaten handelt es sich um Daten, die (zumeist) **mit einem Formular von einer auf eine andere (oder die gleiche) Seite übertragen** werden. Verwendet bzw. verarbeitet werden solche Daten in den meisten Fällen von einer serverseitigen Skript- oder Programmiersprache. Dadurch ist es dann z. B. möglich, die im HTML-Formular angegebenen **Daten per E-Mail zu versenden** oder **in einer Datenbank zu speichern**. Da der E-Mail-Versand und die Datenbankverbindung nicht mit dem CGI-Modul in Verbindung stehen, werden wir an dieser Stelle nicht weiter darauf eingehen, sondern uns lediglich mit dem Auslesen dieser Formulardaten beschäftigen.

### Inhalt dieser Seite:

1. GET-Parameter
2. POST-Parameter

## GET-Parameter

GET-Parameter können entweder **über ein Formular** oder **über einen Link** übergeben bzw. direkt mit der URL angegeben werden. GET-Parameter sind im Gegensatz zu POST-Parametern „sichtbar“, da diese direkt **an die URL angehängt** werden. Die Daten werden nach dem Fragezeichen ? in der Form `name=wert` angegeben. Mehrere Werte werden mit einem &-Zeichen getrennt (z. B. `?name1=wert1&name2=wert2`). Diese Tatsache der Sichtbarkeit ist für einige Zwecke nützlich (z. B. bei einer Blätterfunktion in einem Shop oder Forum) zum anderen aber auch gefährlicher, denn die so übergebenen Daten werden beim Speichern eines Lesezeichens u. U. mit abgelegt und können zudem **leichter manipuliert werden**. Des Weiteren darf die GET-Methode **nicht bei der Übertragung von größeren Datenmengen** genutzt werden. Die Verwendung der GET-Methode in Kombination mit einem Kontaktformular ist daher weder „sicher“ noch sinnvoll. Für andere Zwecke (z. B. die oben erwähnte Blätterfunktion) ist die GET-Methode jedoch durchaus sinnvoll. Um den **Wert eines GET-Parameters zu ermitteln**, kann die Funktion `url_param()` genutzt werden. Dieser wird als Übergabewert der Name des Parameters übergeben. Als Rückgabewert erhalten Sie den Wert des Parameters oder `undef`, wenn der Parameter nicht mitgeschickt wurde bzw. nicht in der URL enthalten ist.

```

1  #!/usr/bin/perl -w
2
3  use strict;
4  use CGI;
5
6  my $cgi = new CGI();
7
8  # Parameter einlesen
9  my $a = $cgi->url_param('a') || '-';
10 my $b = $cgi->url_param('b') || '-';
11
12 print <<END;
13 Content-Type: text/html;charset=UTF-8
14
15 <!doctype html>
16 <html>
17   <head>
18     <title>GET-Parameter - Perl Code-Beispiel</title>
19
20     <meta charset="utf-8" />
21
22     <meta name="robots" content="noindex,nofollow" />
23     <meta name="publisher" content="Homepage-Webhilfe" />
24   </head>
25
26   <body>
27     Wert a: $a<br />
28     Wert b: $b<br />
29     <br />
30     <b>Links:</b><br />
31     <a href="formular-get.pl">A: -, B: -</a><br />
32     <a href="formular-get.pl?a=7">A: 7, B: -</a><br />
33     <a href="formular-get.pl?b=4">A: -, B: 4</a><br />
34     <a href="formular-get.pl?a=8&b=3">A: 8, B: 3</a>
35   </body>
36 </html>
37 END

```



## POST-Parameter

POST-Parameter werden außerhalb der URL übertragen, wodurch es auch möglich ist, **große Mengen an Daten** zu versenden. Ein Speichern der Daten in einem Hyperlink ist dann ebenfalls nicht mehr möglich. Möchten Sie POST-Parameter mittels der CGI-Moduls auslesen, so können Sie die Funktion `param()` verwenden. Dieser wird ebenfalls der Name des Parameters übergeben und als Rückgabe erhalten Sie auch hier den Wert des Parameters (oder `undef`, wenn der Parameter nicht mit versendet wurde).

```

1  #!/usr/bin/perl -w
2
3  use strict;
4  use CGI;
5
6  my $cgi = new CGI();
7
8  # Parameter einlesen
9  my $anrede = $cgi->param('anrede') || '';

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Perl](#) » [Formulardaten](#)

```

10 my $vorname = $cgi->param('vorname') || '';
11 my $nachname = $cgi->param('nachname') || '';
12
13 print <<END;
14 Content-Type: text/html;charset=UTF-8
15
16 <!DOCTYPE html>
17 <html>
18   <head>
19     <title>POST-Parameter - Perl Code-Beispiel</title>
20
21     <meta charset="utf-8" />
22
23     <meta name="robots" content="noindex,nofollow" />
24     <meta name="publisher" content="Homepage-Webhilfe" />
25   </head>
26
27   <body>
28     Wert "anrede": $anrede<br />
29     Wert "vorname": $vorname<br />
30     Wert "nachname": $nachname<br />
31     <br />
32     <form action="formular-post.pl" method="post">
33       <table>
34         <tr>
35           <td style="width: 100px;">Anrede:</td>
36           <td>
37             <select name="anrede">
38               <option value="H">Herr</option>
39               <option value="F">Frau</option>
40             </select>
41           </td>
42         </tr>
43         <tr>
44           <td style="width: 100px;">Vorname:</td>
45           <td><input type="text" name="vorname" /></td>
46         </tr>
47         <tr>
48           <td style="width: 100px;">Nachname:</td>
49           <td><input type="text" name="nachname" /></td>
50         </tr>
51         <tr>
52           <td></td>
53           <td><input type="submit" value="Absenden" /></td>
54         </tr>
55       </table>
56     </form>
57   </body>
58 </html>
59 END

```

**Wichtig:** Die Funktion `param()` kann auch bei GET-Parametern verwendet werden. Jedoch sollte für eine klarere Trennung zwischen GET und POST bevorzugt die Funktion `url_param()` verwendet werden.

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

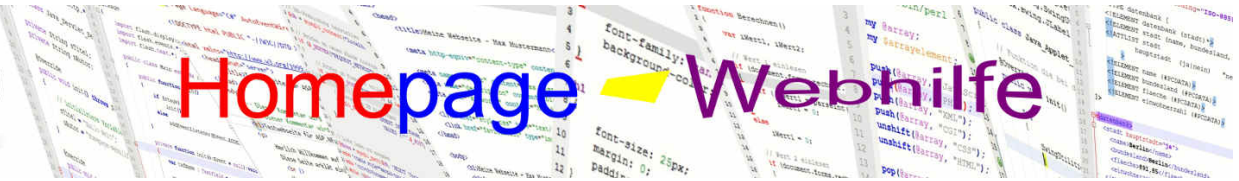
## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Perl](#) » [Cookies](#)

## Cookies

Cookies erlauben das **seiten- und sessionübergreifende Speichern von Informationen**. Cookies haben, so wie Formularfelder auch, einen Namen und einen Wert. Dadurch, dass jedes Cookie einen Namen besitzt, ist es möglich, dass eine Webseite mehrere Cookies (natürlich mit unterschiedlichen Namen) schreibt. Cookies werden vom Server geschrieben (HTTP-Headerfeld `Set-Cookie`) und vom Browser bei jedem Aufruf einer Seite der Website wieder an den Server geschickt (HTTP-Headerfeld `Cookie`). Die **Speicherung der Cookies erfolgt durch den Browser**. Cookies haben einen **Gültigkeitsbereich in Form eines Pfades und einer zeitlichen Begrenzung**.



Um mit Cookies zu arbeiten, stellt uns das `CGI`-Modul die Funktion `cookie()` zur Verfügung. Um ein **Cookie zu schreiben**, müssen Sie der Funktion eine Hash-Referenz mit den Elementen `-name` (Cookie-Name) und `-value` (Cookie-Wert) übergeben. Über den Elementnamen `-expires` lässt sich zudem die Gültigkeitsdauer festlegen. Als Wert ist eine absolute Zeitangabe (Ablaufzeitpunkt) oder ein zeitlicher Abstand (z. B. `+1h` für 1 Stunde Gültigkeit) möglich. Wird `-expires` weggelassen, so besteht das **Cookie bis zum Schließen des Browsers**. Mit `-path` lässt sich der Gültigkeitsort festlegen, um so z. B. ein Cookie auf ein bestimmtes Verzeichnis (und dessen Unterverzeichnisse) zu beschränken. Die Funktion `cookie()` gibt eine Zeichenkette zurück, welche in das Headerfeld `Set-Cookie` platziert werden muss. Dies kann „manuell“ gemacht werden (siehe Beispiel) oder dem Element `-cookie` der Funktion `header()` übergeben werden.

Möchten Sie den **Wert eines Cookies auslesen**, so verwenden Sie ebenfalls die Funktion `cookie()`. Dieser dürfen Sie dann jedoch nur den Namen des Cookies oder eine Hash-Referenz, in welcher lediglich das `-name`-Element vorkommt, übergeben. Als Rückgabewert erhalten Sie dann den Wert des Cookies oder `undef`, wenn das Cookie nicht gefunden werden konnte.

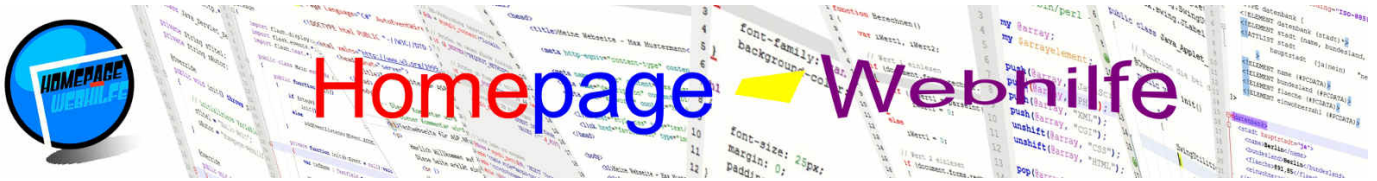
Im folgenden Beispiel werden die Formulareingaben für den Vornamen und Nachnamen in einem Cookie gespeichert, welches für 1 Stunde gültig ist:

```

1  #!/usr/bin/perl -w
2
3  use strict;
4  use CGI;
5
6  my $cgi = new CGI();
7
8  # Parameter versuchen einzulesen
9  my $vorname, my $nachname, my $cookieString = '';
10 if ($cgi->param('vorname') ne '' && $cgi->param('nachname') ne '')
11 {
12     # Parameter-Werte speichern
13     $vorname = $cgi->param('vorname');
14     $nachname = $cgi->param('nachname');
15
16     # Cookies an Browser schicken
17     $cookieString = 'Set-Cookie: '.$cgi->cookie({
18         -name => 'vorname',
19         -value => $vorname,
20         -expires => '+1h'
21     })."\nSet-Cookie: ".$cgi->cookie({
22         -name => 'nachname',
23         -value => $nachname,
24         -expires => '+1h'
25     })."\n\n";
26 }
27 else
28 {
29     # Cookies versuchen auszulesen
30     $vorname = $cgi->cookie('vorname') || '';
31     $nachname = $cgi->cookie('nachname') || '';
32 }
33
34 print <<END;
35 Content-Type: text/html;charset=UTF-8
36 $cookieString
37 <!doctype html>
38 <html>
39 <head>
40 <title>Cookie - Perl Code-Beispiel</title>
41
42 <meta charset="utf-8" />
43
44 <meta name="robots" content="noindex,nofollow" />
45 <meta name="publisher" content="Homepage-Webhilfe" />
46 </head>
47
48 <body>
49 <form action="cookie.pl" method="post">
50 <table>
51 <tr>
52 <td style="width: 100px;">Vorname:</td>
53 <td><input type="text" name="vorname" value="$vorname" /></td>
54 </tr>
55 <tr>
56 <td style="width: 100px;">Nachname:</td>

```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: Homepage-Webhilfe » Perl » Cookies

```
57         <td><input type="text" name="nachname" value="$nachname" /></td>
58     </tr>
59     <tr>
60         <td></td>
61         <td><input type="submit" value="Cookie schreiben" /></td>
62     </tr>
63 </table>
64 </form>
65 </body>
66 </html>
67 END
```



**Wichtig:** Speichern Sie in Cookies niemals sicherheitsrelevante Daten, da Cookies ausgelesen und manipuliert werden können.

**Übrigens:** Um einem Cookie einen neuen Wert zuzuweisen, müssen Sie das Cookie einfach erneut (mit dem „neuen“ Wert) schreiben. Dadurch wird das Cookie vom Browser automatisch ersetzt.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Perl](#) » [Abschluss](#)

## Abschluss

Wie Sie nun gesehen haben, ist das CGI-Modul von Perl nicht besonders komplex, jedoch enthält es alle Funktionen, die wir benötigen, um **HTTP-Header und HTML-Code zu erzeugen, Formulardaten zu verarbeiten und Cookies zu verwalten**.

Viele Fragen sich, ob Sie Perl bevorzugt von PHP oder PHP bevorzugt von Perl einsetzen sollten. Diese Frage kann jedoch nicht ohne weiteres beantwortet werden: Jede Sprache hat seine Vorteile und letztendlich ist es auch **Geschmackssache**. Wer dem **Trend** nachgehen möchte, sollte lieber auf PHP setzen. Zudem ist es auch nicht gewährleistet, dass Perl in Zukunft noch immer von den meisten Providern unterstützt wird. Wer Perl bei seinem Provider nutzen kann und **über Programmierkenntnisse und -erfahrungen in Perl verfügt**, dem wird die Einarbeitung in das CGI-Modul u. U. leichter fallen, als eine komplett neue Sprache zu erlernen. Letztendlich entscheiden Sie jedoch selbst und sollten Vor- und Nachteile speziell für Sie als Person und für Ihr Projekt abwägen.

Neben Perl und [PHP](#), zu welchem wir auf dieser Website auch ein Tutorial anbinden, gibt es noch andere serverseitige Technologien. Zu den Bekanntesten zählen hier noch [ASP.NET](#) und [Java EE](#).

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

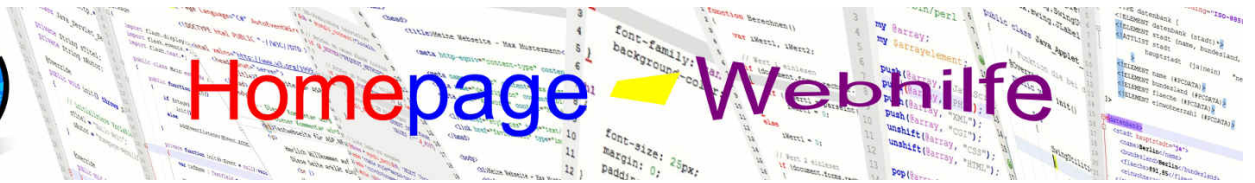
Java EE

XML

# E-Book

## ASP.NET





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [Einführung](#)

## Einführung



ASP.NET ist ein **Framework zur Entwicklung von professionellen und komplexen Websites und Webanwendungen**. Das ASP.NET-Framework ist dabei Teil des .NET Frameworks von Microsoft. ASP.NET wird auf ca. 16% der Websites im Internet als serverseitige Technologie eingesetzt und liegt damit zwar an Platz 2, jedoch trotzdem weit hinter PHP, bei welcher eine Verwendung von ca. 81% bekannt ist.

Bei ASP.NET handelt es sich, anders wie bei PHP, um keine Programmiersprache, sondern lediglich um ein Framework, d. h. ein Satz mit Namensräumen, Klassen und Interfaces. Oft wird auch von den ASP.NET-Technologien oder ASP.NET-Techniken gesprochen. Um eine ASP.NET-

Anwendung zu erstellen, benötigen Sie eine **.NET Programmiersprache** (in der Regel C# oder Visual Basic). Für dieses Tutorial sollten Sie bereits gute Kenntnisse in einer der Sprachen haben. Für einen Schnelleinstieg in C# können wir Ihnen unseren [Crashkurs](#) empfehlen. Ein [ausführliches Tutorial](#) zur Programmiersprache C# finden Sie auf unserer Partnerwebseite. Die Beispiele in diesem Kapitel sind alle in C# geschrieben, lassen sich jedoch weitestgehend in Visual Basic portieren.

ASP.NET unterstützt unterschiedliche Technologien bzw. Techniken (seltener auch als Verfahren bezeichnet), die dabei alle auf dem HTTP-Handler des ASP.NET-Framework aufsetzen. Durch die **unterschiedlichen Technologien** ist es möglich, Webanwendungen auf unterschiedliche Art und Weise zu programmieren. Die Wahl der Technologie hängt u. a. vom Umfang und Aufbau des Projekts sowie vom Geschmack des Programmierers / der Programmierer ab. Die unterschiedlichen Technologien werden wir [weiter unten](#) genauer erläutern.

Ein großer **Vorteil** von ASP.NET im Gegensatz zu anderen serverseitigen Technologien (wie z. B. PHP) ist, dass bereits ein enormer Satz an Klasse und Funktionen aus dem Framework zur Verfügung stehen. Wer sich also mit .NET gut auskennt, der wird größeres Interesse haben, ASP.NET kennenzulernen, als eine komplett neue Programmiersprache zu lernen. Als **Nachteil** von ASP.NET lässt sich vor allem (nur in Bezug auf das .NET Framework) die Gebundenheit an das Betriebssystem Windows und somit die höheren Hosting-Gebühren nennen.

## Geschichte

Die Entstehung von ASP.NET begann bereits im Jahre 1996, mit der Veröffentlichung der Technologie ASP. ASP-Anwendungen (heutzutage auch als **klassisches ASP** oder Classic ASP bezeichnet) konnten damals mit einer Skriptsprache wie VBScript und JScript erstellt werden.

Im Jahre 2002 wurde ASP durch die Veröffentlichung des .NET Frameworks mit ASP.NET abgelöst. ASP wird seither von Microsoft nicht mehr weiterentwickelt. Die Versionen von ASP.NET entsprechen den Versionen des .NET Frameworks, auch wenn die Veröffentlichung der ASP.NET-Versionen zeitweise verzögert zur Veröffentlichung der .NET Framework-Version stattfand.

Seit 2016 gibt es parallel zum ASP.NET-Framework das **ASP.NET-Core-Framework**. Dieses ist Teil des .NET Core Frameworks. Das .NET Core Framework wird ebenfalls von Microsoft entwickelt und existiert parallel zum .NET-Framework. Es wird entwickelt, um .NET auch für andere Plattformen verfügbar zu machen. Das klassische .NET-Framework ist, wie Ihnen vermutlich bekannt ist, auf Windows-Betriebssysteme limitiert.

## Entwicklung und Webserver

Als Entwicklungsumgebung für ASP.NET kommt in der Regel **Visual Studio** zum Einsatz. Ab Visual Studio 2013 gibt es eine Community Version. Diese ist vom Funktionsumfang nicht mehr eingeschränkt und kann für Privatanwender sowie für kleinere Unternehmen kostenfrei genutzt werden. Als Alternative zur IDE Visual Studio steht u. a. auch hier die IDE SharpDevelop zur Verfügung, jedoch ist die Verwendung von Visual Studio zu bevorzugen.

Klassischerweise werden ASP.NET-Anwendungen auf dem **Webserver IIS** (Internet Information Services) von Microsoft ausgeführt. Der IIS kann auf Windows-Betriebssystemen installiert werden. Zudem enthält Visual Studio eine Express-Version des IIS. Eine Ausführung auf Apache-Webservern ist nicht direkt, sondern nur über Umwege (z. B. über das Modul `mod_aspdotnet` oder `mod_mono`) möglich. Des Weiteren gibt es den Cassini-Webserver, welcher von Microsoft als Beispiel-Webserver veröffentlicht wurde. Der Produktiveinsatz von Cassini ist jedoch zu vermeiden.

## Funktionsweise und Technologien

In ASP.NET gibt es unterschiedliche Technologien, wovon alle auf die **HTTP-Laufzeitumgebung** (auch als HTTP-Handler bezeichnet) aufsetzen. ASP.NET-Anwendungen können nur von Webservern, welche ASP.NET unterstützen, ausgeführt und gehostet werden. Die bekanntesten Technologien sind ASP.NET WebForms, ASP.NET MVC, ASP.NET Web API und ASP.NET Web Pages.

**ASP.NET WebForms** ist der Grundbaustein von ASP.NET und ähnelt teilweise dem klassischen ASP. Bei ASP.NET können, wie bei PHP auch, mehrere Seiten bzw. Dateien (Endung `.aspx`) erstellt werden. Zudem kann ein ASP.NET WebForms-Projekt natürlich auch weitere C#-Klassen enthalten.

Eine weitere Technologie ist **ASP.NET MVC**, bei welcher eine strikte Trennung zwischen Datenmodell (engl. *model*), Ansicht (engl. *view*) und Steuerung (engl. *controller*) besteht. Diese Technologie kann dabei nicht direkt mit ASP.NET WebForms verglichen werden. ASP.NET MVC ist im Gegensatz zu ASP.NET WebForms auch Teil des neuen .NET Core Frameworks.

Sowohl für ASP.NET WebForms als auch für ASP.NET MVC bieten wir Ihnen hier ein Tutorial an. Wenn Sie im Gebiet der ASP.NET-Entwicklung komplett neu sind, empfehlen wir Ihnen dringend, zuallererst das Tutorial zu ASP.NET WebForms durchzulesen.

Die Technologien ASP.NET Web API und ASP.NET Web Pages werden seltener eingesetzt, weshalb wir diese hier nicht behandeln.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Grundlagen](#)

## Grundlagen

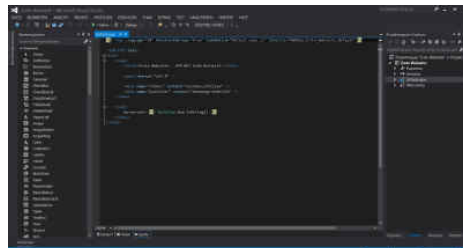
ASP.NET WebForms gilt als **zentraler Bestandteil** von ASP.NET und ist zudem vergleichbar mit dem „klassischen ASP“. WebForms bietet eine einfache Möglichkeit, den **HTML-Code** und die **Logik der Webanwendung miteinander zu verbinden**. Dies ist auf Grund des MVC-Konzepts nicht immer erwünscht, erleichtert jedoch oft die Erstellung einer Webseite. ASP.NET WebForms kann also teilweise mit PHP verglichen werden: Es gibt einen HTML-Code, in welchem Codeblöcke eingebettet werden. In diesen Codeblöcken können dann programmgesteuert (also z. B. verknüpft mit Abfragen oder Schleifen) HTML-Ausgaben oder anderes ausgeführt werden.

### Inhalt dieser Seite:

1. Entwicklung
2. Erste Webseite
3. Dateien und Syntax
4. Funktionsweise

## Entwicklung

Zur Entwicklung einer WebForms-Applikation wird üblicherweise die Microsoft-eigene Entwicklungsumgebung **Visual Studio** eingesetzt. Natürlich gibt es auch einige Alternativen zu dieser IDE. Jedoch ist zu erwähnen, dass die Verwendung von Visual Studio gerade für Anfänge oft einfacher ist. Hier ist es von Vorteil, dass seit Version 2015 die vollausgestattete IDE für Privatanwender **kostenlos zur Verfügung** gestellt wird.



Um ein WebForms-Projekt in Visual Studio zu erstellen, wählen Sie „ASP.NET-Web Forms-Anwendung“ unter dem Menüpunkt „Web“ aus. Dabei wird ein **Beispiel-Projekt** erzeugt, welches bereits einige Seiten und Programmcode enthält. Für ein übersichtliches Programm bzw. für ein eigenes Projekt ist es jedoch u. U. ratsamer, den Eintrag „Leere ASP.NET-Webanwendung“ auszuwählen. Dadurch erhalten Sie ein **leeres Projekt**, welchem Sie nun im Nachhinein einzelne WebForms-Seiten hinzufügen können. Hierzu klicken Sie, nachdem Sie das Projekt erstellt haben, auf Ihr Projekt und wählen „Hinzufügen“ » „Neues Element“. Im darauf erscheinenden Dialog können Sie „Web Form“ auswählen und einen Dateinamen eingeben. Anders als bei Apache-Servern wird für die Standardseite eines Verzeichnisses nicht der Name `index` (z. B. `index.html` oder `index.php`) verwendet, sondern `Default` (in Bezug auf ASP.NET: `Default.aspx`). Dies kann jedoch über die Webserverkonfiguration (Datei `Web.config`) geändert werden.

Beim Erstellen eines „Web Form“-Elements werden automatisch mehrere Dateien (z. B. bei Verwendung des Basisnamens `Default`) angelegt: `Default.aspx`, `Default.aspx.cs` und `Default.aspx.designer.cs`. Die Datei mit der Endung `.aspx` enthält die „Web Form“ (zu Deutsch das Webformular). Grundsätzlich kann der Inhalt der `.aspx`-Datei mit einer HTML-Datei verglichen werden. Neben HTML-Code können hier jedoch auch **ASP.NET-Codeblöcke** eingebunden werden (dazu später mehr). Des Weiteren gibt es spezielle HTML-Elemente, welche bevor diese an den Browser geschickt werden, ersetzt werden. Dadurch ist es dann leicht möglich, Benutzersteuerelemente (engl. *User Controls*) und andere Webserver-Steuerelemente (z. B. einen Kalender) einzubinden. Die `.aspx.cs`-Datei wird als **Code-Behind-Datei** bezeichnet und enthält den eigentlichen Programmcode (C# oder `.aspx.vb` für Visual Basic). Der Ansatz mit der Code-Behind-Datei wird von Microsoft auch als **Code-Behind-Modell** bezeichnet und dient zur Trennung zwischen „View“ und „Controller“. So ist das MVC-Konzept zwar bereits in ASP.NET WebForms enthalten, jedoch in ASP.NET MVC noch stärker ausgeprägt. Eine Datei mit der Endung `.aspx.designer.cs` wird von Visual Studio automatisch erstellt und erweitert die Klasse, welche in der Datei `.aspx.cs` definiert ist. Bei dem Inhalt der Datei `.aspx.designer.cs` handelt es sich um **Variablen Deklarationen** von Webserver-Steuerelementen und anderen serverseitigen Steuerelementen, welche ein `id`-Attribut besitzen.

Visual Studio enthält neben Codeervollständigung (IntelliSense), Syntax-Highlighting und der Projektverwaltung einen Debugger, einen Designer (dies gilt nicht nur für Windows Forms, sondern eben auch für ASP.NET), einen Compiler und den IIS Express. Beim **Debuggen einer Webanwendung** wird nach der Kompilierung der **IIS Express** gestartet und die Anwendung im Standardbrowser geöffnet. Das Debuggen ist also grundsätzlich so wie bei anderen Anwendungen auch.

Möchten Sie Ihre Projekte auf einem anderen Webserver ausführen und / oder veröffentlichen, so müssen Sie auf diesen lediglich die `.aspx`-Dateien sowie den Ordner `bin` kopieren. Der Ordner `bin` muss dabei standardmäßig im Root-Verzeichnis liegen. Für jedes Projekt gibt es eine eigene DLL. Die **DLL enthält den kompilierten Code der C#-Dateien** und somit auch den der Dateien `.aspx.cs` und `.aspx.designer.cs`. Des Weiteren benötigen Sie die Konfigurationsdatei `Web.config`.

**Wichtig:** Die aufgezählten Schritte und Bezeichnungen für die Erstellung eines Web-Projekts beziehen sich auf Visual Studio 2013 und das .NET Framework 4.0. Bei anderen Visual Studio Versionen oder anderen .NET Framework Versionen können sich diese Schritte und Bezeichnungen unterscheiden.

## Erste Webseite

Nach dieser theoriebelasteten Einführung wollen wir nun unsere **erste Webseite erstellen**. Hierzu erstellen wir ein leeres Projekt und legen ein Webformular mit dem Namen `Default.aspx` (wie oben beschrieben) an. Anschließend können Sie den automatisch erstellten HTML-Code an Ihre Vorstellungen anpassen oder diesen auch vorerst beibehalten. Nun fügen Sie im `body`-Element folgenden Code ein `<%= DateTime.Now.ToString() %>`. Durch diesen Code wird die aktuelle Zeit des Servers in die Ausgabedatei geschrieben, welche nach Bearbeitung der kompletten `.aspx`-Datei an den Webbrowser gesendet wird.

Das folgende Beispiel enthält lediglich die Ausgabe der Serverzeit. Über das Lupenicon können Sie sich das Beispiel ansehen. Mit dem ersten Icon können Sie sich die Projektmappe des Projekts downloaden. Für die Erstellung dieses und aller anderen Beispiele wurde Visual Studio 2013 Professional benutzt. Die Beispiele wurden mit dem .NET Framework 4.0 erstellt, worauf auch diese Tutorial basiert. Auf den Codeblock (beginnend mit `<%=` und endend mit `%>`) gehen wir weiter unten noch genauer ein. Der erste Code-Ausschnitt zeigt den ASPX-Code (welcher von uns geändert wurde). Im zweiten Code-Ausschnitt ist der C#-Code (von der Code-Behind-Datei) zu sehen, welche von uns noch nicht geändert wurde.

```

1  <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="HWhBsp.Erste_Webseite.Default" %>
2
3  <!DOCTYPE html>
4  <html>
5      <head>
6          <title>Erste Webseite - ASP.NET Code-Beispiel</title>
7
8          <meta charset="utf-8" />
9
10         <meta name="robots" content="noindex,nofollow" />
11         <meta name="publisher" content="Homepage-Webhilfe" />
12     </head>
13 
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

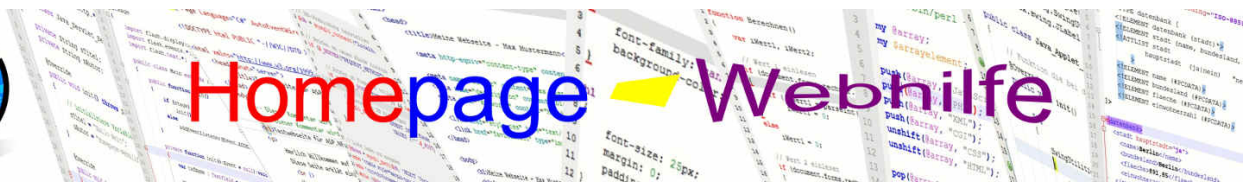
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Grundlagen](#)

```

14 |     <body>
15 |         Serverzeit: <%= DateTime.Now.ToString() %>
16 |     </body>
17 | </html>
1 | using System;
2 |
3 | namespace HwhBsp.Erste_Webseite
4 | {
5 |     public partial class Default : System.Web.UI.Page
6 |     {
7 |         protected void Page_Load(object sender, EventArgs e)
8 |         {
9 |
10 |         }
11 |     }
12 | }
    
```



## Dateien und Syntax

Grundsätzlich ist zwischen ASPX- und C#-Code zu differenzieren. Der „reine“ C#-Code befindet sich in der Code-Behind-Datei und somit in den Dateien `.aspx.cs` und `.aspx.designer.cs`. Dort kann, so wie in jeder anderen `.cs`-Datei auch, ganz normaler C#-Code notiert werden. Natürlich können Sie Ihrem Projekt auch weitere `.cs`-Dateien hinzufügen. In dem C#-Code einer ASP.NET-Anwendung stehen Ihnen selbstverständlich alle Klassen des .NET-Frameworks zur Verfügung. Für die Webanwendung selbst werden jedoch Klassen aus dem Namensraum `System.Web` und deren Sub-Namensräume verwendet. Die Basisklasse einer Seite ist `Page` aus dem Namensraum `System.Web.UI`. Der ASPX-Code besteht grundsätzlich mal aus HTML-Code (ggf. mit eingebettetem CSS- oder JavaScript-Code) und einer sogenannten Direktive (dazu gleich mehr).

Innerhalb des ASPX-Codes gibt es unterschiedliche Möglichkeiten, C#-Code einzubetten, um somit z. B. HTML-Code dynamisch auszugeben. C#-Code kann z. B. über das HTML-Element `script` mit dem Attribut `runat` und dem Wert `server` eingebettet werden. Diese Methode wird vor allem bei vielen Beispielen (überwiegend der von Microsoft) verwendet, um Event-Handler zu notieren. Wir empfehlen Ihnen jedoch auf Grund des MVC-Konzepts, diese in die Code-Behind-Datei auszulagern.

```

1 | <script runat="server">
2 | for (int i = 0; i < 10; i++)
3 | {
4 |     Response.Write(i + "<br />");
5 | }
6 | </script>
    
```

Geläufiger sind jedoch die Tags für sogenannten **Inline-Code**. In diesen kann, so wie im `script`-Element auch, eine Schleife, Abfrage oder anderes platziert werden, jedoch sind Variablen die innerhalb dieses Blocks notiert wurden, nur innerhalb dieses Blocks verfügbar. Inline-Codes werden innerhalb der Server-Tags `<%` und `%>` platziert.

```

1 | <%
2 | for (int i = 0; i < 10; i++)
3 | {
4 |     Response.Write(i + "<br />");
5 | }
6 | %>
    
```

Von dem öffnenden Tag `<%` gibt es noch einige andere Varianten. Dazu zählen `<%@`, `<%=`, `<%:`, `<%#` und `<%$`. Bei `<%@` handelt es sich um sogenannte Direktiven, auf welche wir nachher noch genauer eingehen. `<%=` und `<%:` wird dazu verwendet eine **Ausgabe in der resultierenden HTML-Datei** durchzuführen. Dabei entspricht der Code `<%= "Hallo!" %>` dem Code `<% Response.Write("Hallo!") %>`. Der Tag `<%=` ist natürlich hier zu bevorzugen, da dadurch ein **kürzerer Code** entsteht. Der Unterschied zwischen `<%=` und `<%:` ist, dass bei `<%:` der Inhalt vorher **HTML enkodiert** wird. Dadurch können Sonderzeichen von HTML (vor allem `<`, `>` und `&`, aber auch `"` und `'`) „sicher“ dargestellt werden. Je nach Seitenkodierung werden auch andere Zeichen in Sonderzeichen (Unicode-Format) umgewandelt. Eine solche Kodierungsfunktion findet sich in der Klasse `HttpUtility` (Funktion `HtmlEncode`). Bei Verwendung des Tags `<%:` wird diese Funktion automatisch aufgerufen, bei `<%=` hingegen nicht.

```

1 | <%= "Hallo Welt!" %>
1 | <%: "< Herzlich Willkommen >" %>
    
```

Der Tag `<%$` wird für sogenannte **Ausdrücke** benutzt. Dadurch ist es möglich, Werte aus Konfigurationsdateien (`AppSettings`) oder Ressourcendateien (`Resources`) zu holen. Als allgemeiner Syntax gilt hier `Ausdruckstyp:Ausdruckswert`:

```

1 | <%$ AppSettings:EmailAdresse %>
    
```

Mit Hilfe des Tags `<%#` ist es möglich, **Datenbindungen** vorzunehmen. Diese werden wir jedoch an dieser Stelle nicht weiter behandeln.

```

1 | <%# kontaktListe %>
    
```

Nun wollen wir uns auch noch mit den sogenannten **Direktiven** beschäftigen. Direktiven stehen (zumeist) in den obersten Zeilen von ASPX-Dateien oder vergleichbaren Dateien (z. B. ASAX-, ASCX- oder Master-Dateien) und sind dazu gedacht, Einstellungen für die Kompilierung festzulegen. Für die ersten Seiten (und auch auf der Seite im oberen Beispiel) wird lediglich die `Page`-Direktive benutzt.

```

1 | <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="MeineTestseite.Default" %>
    
```

Es gibt noch einige weitere Direktiven, wie z. B. `<%@ Control` für Benutzersteuerelemente und `<%@ Master` für Masterseiten. Neben den bereits genannten Direktiven gibt es noch ein paar weitere, die jedoch nicht sehr häufig verwendet werden.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Grundlagen](#)

Neben den bisher genannten Tags gibt es noch ein Tag-Paar für **serverseitige Kommentare** (außerhalb eines Codeblocks). Dabei handelt es sich um die Tags `<%--` und `--%>`. Diese Art der Kommentierung kann dazu genutzt werden, wenn ein bestimmter Teil (z. B. der ASPX-Datei) nicht an den Browser gesendet und nicht vom Interpreter bearbeitet werden soll.

```
1 | <%-- Dieser Text wird nicht an den Browser gesendet! --%>
```

### Funktionsweise

Wie bereits oben genannt, werden die einzelnen WebForms von der Klasse `Page` (Namensraum `System.Web.UI`) abgeleitet. Zwei wichtige Eigenschaften dieser Klasse sind `Request` und `Response`, welche auf ein Objekt der Klasse `HttpRequest` und `HttpResponse` verweisen. Mit Hilfe dieser Klasse ist es möglich, Daten der Anfrage (engl. *request*) abzufragen und die Antwort (engl. *response*) zu beeinflussen. Wie bereits oben angesprochen, kann mittels der Funktion `Response.Write()` Inhalt auf die HTML-Seite geschrieben werden. Auf weitere Eigenschaften und Funktionen dieser Klassen gehen wir später noch genauer ein.

ASP.NET stellt uns einige sogenannte Webserver-Steuerelemente zur Verfügung. Neben diesen ist es jedoch auch möglich, auf einfache HTML-Steuerelemente (z. B. ein `input`-Element) zuzugreifen. Dabei sind alle HTML-Elemente von der Basisklasse `HtmlControl` (Namensraum `System.Web.UI.HtmlControls`) abgeleitet. Für einige Elemente (z. B. das `input`-Element) gibt es eine von `HtmlControl` abgeleitete Klasse (z. B. `HtmlInputControl` für das `input`-Element). In beiden Fällen, also bei normalen HTML-Elementen und bei Webserver-Steuerelementen, muss in dem Element zwingend eine ID (Attribut `id`) und das Attribut `runat` mit dem Wert `server` gesetzt werden. Der mit dem Attribut `id` vergebene Name wird auch im `id`-Attribut des HTML-Codes gesetzt. Webserver-Steuerelemente besitzen als Tag-Namen immer das Präfix `asp` gefolgt von einem Doppelpunkt. Mehr dazu jedoch im Thema [Webserver-Steuerelemente](#). Zudem muss noch beachtet werden, dass alle Elemente auf die per ASP.NET zugegriffen werden soll innerhalb eines `form`-Elements platziert werden müssen. Das `form`-Element muss dann ebenfalls das Attribut `runat` mit dem Wert `server` besitzen.

Neben dem Zugriff auf Attribute und Werte von HTML-Elementen und Webserver-Steuerelementen können bei diesen auch **Ereignisse** registriert werden. Da das HTTP-Protokoll zustandslos ist und somit über mehrere Verbindungen sich Daten nicht merken, müssen alle Informationen einer Sitzung bzw. einer Seite serverseitig gespeichert oder clientseitig im HTML-Dokument hinterlegt und beim Absenden eines Formulars mitübertragen werden. Hier ist uns ASP.NET, im Gegensatz zu PHP, eine sehr große Hilfe. Ereignisse von Steuerelementen werden über sogenannte **Postbacks** (engl. zurückschicken) ausgelöst. Ereignisse solcher Art werden i. d. R. nicht direkt ausgelöst, sondern innerhalb des sogenannten Ansichtszustands gespeichert und beim nächsten Übertragen des Formulars vom Webserver ausgelöst. Sollen die Ereignisse direkt ausgelöst werden, was jedoch auch ein erneutes Laden der Seite zur Folge hat, so können Sie die Eigenschaft `AutoPostBack` des jeweiligen Elements setzen. Postbacks kommen jedoch auch beim Übertragen von Formularen zum Einsatz, wenn die Formulardaten an den Server übertragen bzw. „gepostet“ werden. Hier ist auch selber ausprobieren angesagt, denn wodurch lernt man besser, wie sich bestimmte Einstellungen auswirken, wie wenn man es selber sieht. Um zu prüfen, ob es sich bei der aktuellen Anfrage um einen Postback handelt, kann die Eigenschaft `IsPostBack` geprüft werden.

Neben den Ereignissen von Elementen gibt es noch ein paar **Seitenevents**, welche während des **Lebenszyklus einer Seite** aufgerufen werden. Die Events müssen standardmäßig nicht registriert werden, da diese durch die Eigenschaft `AutoEventWireUp` in der `Page`-Direktive, direkt an die Funktionen `Page_Eventname` gebunden werden (sofern diese vorhanden sind). Folgende Ereignisse stehen zu Verfügung und werden in der aufgelisteten Reihenfolge aufgerufen:

<b>PreInit</b>	Event, bevor die Initialisierung der Seite und deren Elemente ausgeführt wird
<b>Init</b>	Event, bevor die Initialisierung der Seite und deren Elemente ausgeführt wird, jedoch nachdem alle PreInit-Events ausgelöst wurden
<b>InitComplete</b>	Event, nachdem die Initialisierung der Seite und deren Elemente ausgeführt wurde
<b>PreLoad</b>	Event, bevor die Seite und deren Elemente geladen wird
<b>Load</b>	Event, bevor die Seite und deren Elemente geladen wird, jedoch nachdem alle PreLoad-Events ausgelöst wurden
<b>LoadComplete</b>	Event, nachdem die Seite und deren Elemente geladen wurde
<b>PreRender</b>	Event, bevor die Seite und Elemente „dargestellt“ wird
<b>Render</b>	Event, bevor die Seite und Elemente „dargestellt“ wird, jedoch nachdem alle PreRender-Events ausgelöst wurden
<b>Unload</b>	Event, nachdem die Seite und Elemente „dargestellt“ und somit komplett bearbeitet wurde

Bei den Ereignissen ist es auch sinnvoll, ein wenig selbst damit zu experimentieren. Wie Sie der Tabelle entnehmen können, werden Events wie `PreLoad` und `Load` sozusagen direkt nacheinander ausgeführt, jedoch werden zwischen dem Aufruf von `PreLoad` und `Load` der Seite die `PreLoad`-Ereignisse der einzelnen Steuerelemente ausgeführt. Die Funktion des `Load`-Ereignisses wird von Visual Studio standardmäßig in der Code-Behind-Datei angelegt. Das `Load`-Ereignis dient klassischerweise zum **Herstellen von Verbindungen** (z. B. einer Datenbankverbindung) oder zum Laden von Ressourcen. Das `Unload`-Event wird hingegen dazu verwendet, um **Verbindungen wieder abzubauen**. In diesem Moment ist die Datei, welche an den Browser geschickt wird, abgeschlossen und kann nicht mehr verändert werden.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Anfrage und Antwort](#)

## Anfrage und Antwort

Wie bereits im vorherigen Thema erklärt, besitzt die `Page`-Klasse die Eigenschaften `Request` und `Response`, mit welchen Objekte der Klassen `HttpRequest` und `HttpResponse` abgerufen werden können, um somit **Informationen der Anfrage abzurufen und die Antwort zu beeinflussen**. Die wichtigsten Eigenschaften und Funktionen der zwei Klassen werden wir in diesem Thema noch genauer behandeln.

### Inhalt dieser Seite:

1. Anfrage-Informationen
2. Antwort-Informationen
3. URL-Parameter
4. Weiterleitung und Header-Änderungen
5. Streams

## Anfrage-Informationen

Die `HttpRequest`-Klasse enthält einige Eigenschaften, mit welchen Informationen zur HTTP-Anfrage abgerufen werden können. Die wichtigsten Eigenschaften sind in der untenstehenden Tabelle aufgeführt und werden im unteren Beispiel ausgegeben (auf `PhysicalPath` wurde aus Sicherheitsgründen im Beispiel verzichtet).

<b>Path</b>	Der vollständige virtuelle Pfad der Website (FilePath + PathInfo).
<b>FilePath</b>	Der virtuelle Pfad der Website.
<b>PathInfo</b>	Der Zusatz einer Pfadangabe (spezielle ASP.NET-Funktion).
<b>PhysicalPath</b>	Der physikalische Pfad auf dem Webserver.
<b>Url</b>	Die URL der aktuellen Anfrage.
<b>UrlReferrer</b>	Die URL der zuvor aufgerufenen Seite.
<b>HttpMethod</b>	Die HTTP-Methode der Anfrage (GET, POST, aber auch z. B. HEAD).
<b>RequestType</b>	Der Typ der Anfrage (GET oder POST).
<b>IsAuthenticated</b>	Gibt an, ob es sich um eine authentifizierte Anfrage handelt.
<b>IsLocal</b>	Gibt an, ob die Anfrage vom lokalen Computer (gleicher PC wie der Server) ist.
<b>IsSecureConnection</b>	Gibt an, ob die Verbindung verschlüsselt ist.
<b>UserAgent</b>	Der User-Agent des Browsers.
<b>UserHostAddress</b>	Die IP-Adresse des Computers, von dem die Anfrage kommt.
<b>UserHostName</b>	Der Hostname des Computers, von dem die Anfrage kommt.

```

1 <table>
2   <tr>
3     <th>Pfad:</th>
4     <td><%= Request.Path %></td>
5   </tr>
6   <tr>
7     <th>Dateipfad:</th>
8     <td><%= Request.FilePath %></td>
9   </tr>
10  <tr>
11    <th>Pfad-Info:</th>
12    <td><%= Request.PathInfo %></td>
13  </tr>
14  <tr>
15    <th>URL:</th>
16    <td><%= Request.Url.ToString() %></td>
17  </tr>
18  <tr>
19    <th>URL der vorherigen Seite:</th>
20    <td><%= Request.UrlReferrer != null ? Request.UrlReferrer.ToString() : "-" %></td>
21  </tr>
22  <tr>
23    <th>HTTP-Methode:</th>
24    <td><%= Request.HttpMethod %></td>
25  </tr>
26  <tr>
27    <th>Anfrage-Typ:</th>
28    <td><%= Request.RequestType %></td>
29  </tr>
30  <tr>
31    <th>Authentifizierung:</th>
32    <td><%= Request.IsAuthenticated ? "ja" : "nein" %></td>
33  </tr>
34  <tr>
35    <th>Lokaler Computer:</th>
36    <td><%= Request.IsLocal ? "ja" : "nein" %></td>
37  </tr>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Anfrage und Antwort](#)

```

38     <tr>
39         <th>Sichere Verbindung:</th>
40         <td><%= Request.IsSecureConnection ? "ja" : "nein" %></td>
41     </tr>
42     <tr>
43         <th>User-Agent:</th>
44         <td><%= Request.UserAgent != null ? Request.UserAgent : "-" %></td>
45     </tr>
46     <tr>
47         <th>IP-Adresse:</th>
48         <td><%= Request.UserHostAddress %></td>
49     </tr>
50     <tr>
51         <th>Hostname:</th>
52         <td><%= Request.UserHostName %></td>
53     </tr>
54 </table>

```



## Antwort-Informationen

Die `HttpResponse`-Klasse enthält ebenfalls ein paar Eigenschaften zum Abrufen des aktuellen Status der Antwort. Dabei können diese Eigenschaften jedoch nicht nur gelesen sondern auch gesetzt werden, um somit die Antwort (engl. *response*) anpassen zu können. Auch hier haben wir wieder eine Tabelle zusammengestellt und ein nachfolgendes Beispiel erstellt.

<b>Charset</b>	Die Zeichenkodierung der Antwort.
<b>ContentType</b>	Der MIME-Typ der Antwort.
<b>Status</b>	Die vollständige Statuszeichenkette der Antwort (StatusCode + StatusDescription).
<b>StatusCode</b>	Der HTTP-Statuscode der Antwort.
<b>StatusDescription</b>	Der Beschreibungs-Text des HTTP-Status der Antwort.
<b>IsRequestBeingRedirected</b>	Gibt an, ob eine Weiterleitung aktiv ist.
<b>Expires</b>	Die Gültigkeitsdauer des Dokuments in Minuten.
<b>ExpiresAbsolute</b>	Der Ablaufzeitpunkt des Dokuments.

```

1 <table>
2     <tr>
3         <th>Zeichensatz:</th>
4         <td><%= Response.Charset %></td>
5     </tr>
6     <tr>
7         <th>MIME-Typ:</th>
8         <td><%= Response.ContentType %></td>
9     </tr>
10    <tr>
11        <th>Status:</th>
12        <td><%= Response.Status %></td>
13    </tr>
14    <tr>
15        <th>Status-Code:</th>
16        <td><%= Response.StatusCode %></td>
17    </tr>
18    <tr>
19        <th>Status-Beschreibung:</th>
20        <td><%= Response.StatusDescription %></td>
21    </tr>
22    <tr>
23        <th>Weiterleitung:</th>
24        <td><%= Response.IsRequestBeingRedirected ? "ja" : "nein" %></td>
25    </tr>
26    <tr>
27        <th>Ablaufzeitraum:</th>
28        <td><%= Response.Expires.ToString() %></td>
29    </tr>
30    <tr>
31        <th>Ablaufzeitpunkt:</th>
32        <td><%= Response.ExpiresAbsolute.ToString() %></td>
33    </tr>
34 </table>

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Anfrage und Antwort](#)

## URL-Parameter

Um mittels ASP.NET die URL-Parameter (GET-Parameter) auszulesen, können wir die Elemente des assoziativen Arrays (Objekt der Klasse `NameValueCollection`) `QueryString` der `HttpRequest`-Klasse abrufen. Wurde ein Parameter nicht gesetzt, so wird als Wert `null` zurückgegeben. Das folgende Beispiel zeigt die Verwendung der Eigenschaft `QueryString`:

```

1 Wert a: <%= Request.QueryString["a"] != null ? Request.QueryString["a"] : "-" %><br />
2 Wert b: <%= Request.QueryString["b"] != null ? Request.QueryString["b"] : "-" %><br />
3 <br />
4 <b>Links:</b><br />
5 <a href="Default.aspx">A: -, B: -</a><br />
6 <a href="Default.aspx?a=7">A: 7, B: -</a><br />
7 <a href="Default.aspx?b=4">A: -, B: 4</a><br />
8 <a href="Default.aspx?a=8&b=3">A: 8, B: 3</a>

```



## Weiterleitung und Header-Änderungen

Mittels der Funktionen `AddHeader()` oder `AppendHeader()` der `HttpResponse`-Klasse ist es möglich, **HTTP-Header** an die Antwort anzuhängen. Als Parameter werden zwei Zeichenketten übergeben: der Name der Header-Eigenschaft und der Wert der Header-Eigenschaft. Ein Vollzugriff auf alle Headers des HTTP-Responses kann über die Eigenschaft `Headers` abgerufen werden. Damit ist es dann z. B. auch möglich, bestehende Header-Eigenschaften zu ändern oder zu löschen (Funktion `Remove()`). Das Hinzufügen eines Wertes ist dort ebenfalls möglich (Funktion `Add()`). Die Funktion `Redirect()` erlaubt es, eine **Umleitung** durchzuführen. Als Parameter wird der Funktion eine Zeichenkette mit dem Ziel (Pfadangaben oder URLs) übergeben.

```

1 <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="HWhBsp.Weiterleitung_Header.Default" %>
2
3 <%
4     Response.AddHeader("Content-Language", "de");
5     Response.Redirect("https://www.homepage-webhilfe.de/");
6 %>

```



## Streams

Der **Eingangs-Stream** (eingehender Datenstrom) kann über die Eigenschaft `InputStream` des Objekts der Klasse `HttpRequest` abgerufen werden. Für „normale“ Webanwendungen wird dieser Stream jedoch nicht benötigt. Die `HttpResponse`-Klasse enthält ebenfalls einen Datenstrom (jedoch für ausgehende Daten). Der Stream kann über die Eigenschaft `OutputStream` abgerufen werden. Des Weiteren gibt es die Funktionen `Write()` und `WriteFile()`, welche es ermöglichen, Daten aus Variablen oder einer Datei auf den Ausgabe-Stream zu senden.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Formulardaten](#)

## Formulardaten

In diesem Thema wollen wir uns damit beschäftigen, wie Sie in ASP.NET Formulardaten abrufen können. Dabei beziehen wir uns bei diesen Beispielen lediglich auf die **HTTP-Methode POST**. Wie Sie GET-Parameter (oder auch als URL-Parameter bezeichnet) abrufen, haben wir ja bereits im [vorherigen Thema](#) kennengelernt. Das Abrufen von POST-Parametern funktioniert sehr ähnlich. Der einzige Unterschied ist, dass wir hier an Stelle des assoziativen Array `QueryString` das assoziative Array `Form` verwenden. Auch hier wird `null` als Wert zurückgegeben, wenn ein Parameter nicht mitgesendet wurde.

### Inhalt dieser Seite:

1. Einfaches Formular
2. Formular mit Serverzugriff

## Einfaches Formular

Im folgenden Beispiel haben wir ein einfaches HTML-Formular mit den Feldern `anrede`, `vorname` und `nachname`. Oberhalb des Formulars werden die Werte mit Hilfe der Server-Tags für Inline-Code ausgegeben.

```

1 Wert "anrede": <%= Request.Form["anrede"] != null ? Request.Form["anrede"] : "-" %><br />
2 Wert "vorname": <%= Request.Form["vorname"] != null ? Request.Form["vorname"] : "-" %><br />
3 Wert "nachname": <%= Request.Form["nachname"] != null ? Request.Form["nachname"] : "-" %><br />
4 <br />
5 <form action="Default.aspx" method="post">
6   <table>
7     <tr>
8       <td style="width: 100px;">Anrede:</td>
9       <td>
10        <select name="anrede">
11          <option value="H">Herr</option>
12          <option value="F">Frau</option>
13        </select>
14      </td>
15    </tr>
16    <tr>
17      <td style="width: 100px;">Vorname:</td>
18      <td><input type="text" name="vorname" /></td>
19    </tr>
20    <tr>
21      <td style="width: 100px;">Nachname:</td>
22      <td><input type="text" name="nachname" /></td>
23    </tr>
24    <tr>
25      <td></td>
26      <td><input type="submit" value="Absenden" /></td>
27    </tr>
28  </table>
29 </form>

```



## Formular mit Serverzugriff

Einen „besseren“ Zugriff auf Formulardaten bzw. Formularelemente haben wir, wenn wir den Formularelementen eine ID geben und das Attribut `runat` mit dem Wert `server` setzen. Nachdem dies erledigt ist, können wir ganz normal mittels der .NET-Sprache auf das Steuerelement zugreifen, da das Steuerelement über den Namen, der mittels des ID-Attributs vergeben wurde, als Eigenschaft Teil der Klasse von der jeweiligen Seite ist. Um den Wert eines Formularfelds abzurufen, kann die Eigenschaft `Value` des Objekts verwendet werden. Elemente die ein `id`- und `runat`-Attribut haben und somit ein Zugriff vom Server möglich ist, werden auch als **HTML-Server-Elemente** bezeichnet. Das folgende Beispiel ist vom Formular-Aufbau gleich wie das obere Beispiel, enthält jedoch nicht HTML-Elemente sondern HTML-Server-Elemente:

```

1 Wert "anrede": <%= inputAnrede.Value %><br />
2 Wert "vorname": <%= inputVorname.Value %><br />
3 Wert "nachname": <%= inputNachname.Value %><br />
4 <br />
5 <form action="Default.aspx" method="post" runat="server">
6   <table>
7     <tr>
8       <td style="width: 100px;">Anrede:</td>
9       <td>
10        <select name="anrede" id="inputAnrede" runat="server">
11          <option value="H">Herr</option>
12          <option value="F">Frau</option>
13        </select>
14      </td>
15    </tr>
16    <tr>
17      <td style="width: 100px;">Vorname:</td>
18      <td><input type="text" name="vorname" id="inputVorname" runat="server" /></td>
19    </tr>
20    <tr>
21      <td style="width: 100px;">Nachname:</td>
22      <td><input type="text" name="nachname" id="inputNachname" runat="server" /></td>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Formulardaten](#)

```

23         </tr>
24         <tr>
25             <td></td>
26             <td><input type="submit" value="Absenden" /></td>
27         </tr>
28     </table>
29 </form>

```



**Wichtig:** Das Attribut `runat` im `form`-Element darf nicht fehlen, andernfalls wird ein Fehler ausgelöst.

**Übrigens:** Wie Ihnen vielleicht aufgefallen ist, werden die Werte nach dem Klick auf „Absenden“ nicht verworfen, so wie es im ersten Beispiel der Fall ist oder wie wir es von PHP gewohnt sind. Das Speichern der Werte wird durch den sogenannten Ansichtszustand verwaltet. Ist dies für ein Element nicht erwünscht, so kann die Eigenschaft `EnableViewState` auf `false` gesetzt werden.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

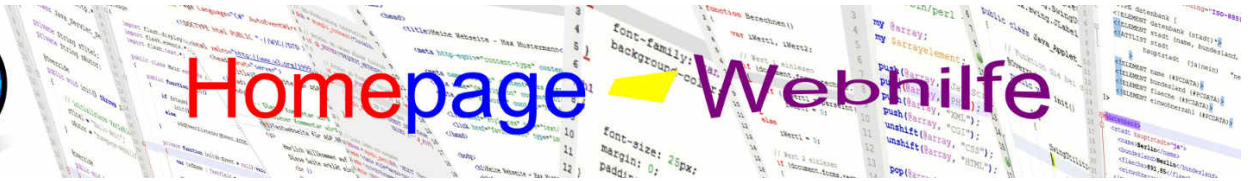
Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung





Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Webserver-Steuerelemente](#)

## Webserver-Steuerelemente

### Inhalt dieser Seite:

1. Basiselemente
2. Formularelemente
3. Kalender
4. Tabelle
5. Diagramm
6. Validierung
7. Ansichten
8. AJAX-Technologie

Webserver-Steuerelemente befinden sich im Namensraum `System.Web.UI.WebControls` und haben im Gegensatz zu HTML-Server-Elementen (Namensraum `System.Web.UI.HtmlControls`) ein paar Vorteile. Ein Vorteil ist z. B., dass bei **Webserver-Steuerelementen Events registriert werden** können. Ereignisse können nicht direkt ausgelöst werden, sondern deren **Ereignisstatus wird lediglich im Ansichtszustand gespeichert**. Beim nächsten Übertragen des Formulars wird der **Ereignisstatus an den Server zurückgeschickt**, welcher dann letztendlich die Event-Behandlungsmethoden ausführt. Sollen Ereignisse direkt ausgelöst werden, so kann bei einigen Steuerelementen die Eigenschaft `AutoPostBack` gesetzt werden. Dadurch wird die Seite bei einer Änderung **direkt an den Server „gepostet“**. Dies hat ein Neuladen der Seite zur Folge, was nicht immer erwünscht ist.

Im Gegensatz zu HTML-Elementen haben Webserver-Steuerelemente „**einfachere**“ **Einstellungsmöglichkeiten**. Diese können in ASP.NET bzw. C# mittels wenig Code festgelegt werden, wobei bei HTML-Elementen mehr Code notwendig wäre. Für einfachere Anwendungszwecke sind jedoch HTML-Server-Elemente oder sogar auch nur HTML-Elemente oft ausreichend. Webserver-Steuerelemente werden auch gerne bei **komplexeren Steuerelementen** eingesetzt, welche in HTML in einfacher Form nicht verfügbar sind. Hierbei lässt sich z. B. ein Kalender- oder Diagramm-Steuerelement nennen.

Webserver-Steuerelemente können im ASP.NET-Code (Datei `.aspx`) direkt eingefügt werden. Diese Elemente werden wie HTML-Elemente notiert und werden vor dem Senden an den Browser in „reinen“ HTML-Code (zumeist in Verbindung mit CSS- und JavaScript-Code) übersetzt. Die Elementnamen von Webserver-Steuerelementen beginnen mit dem Präfix `asp` gefolgt von einem Doppelpunkt und dem Namen des jeweiligen Steuerelements. Die Attribut-Namen entsprechen dem Namen der Eigenschaft der jeweiligen Klasse. Eine Ausnahme stellen Events dar, bei welchen das Schlüsselwort `On` vorangestellt werden muss.

Alle Webserver-Steuerelemente müssen das Attribut `runat` mit dem Wert `server` besitzen. Ist ein Zugriff auf das Element erwünscht, so muss noch zusätzlich das `ID`-Attribut angegeben werden. Mit dem dort definierten Namen kann **auf das Steuerelement zugegriffen werden**. Alle Webserver-Steuerelemente sind von der Basisklasse `WebControl` abgeleitet und besitzen somit ein paar gleiche Attribute:

<b>ForeColor</b>	Vordergrundfarbe bzw. Textfarbe
<b>Font</b>	Schrifteinstellungen (Objekt der Klasse <code>FontInfo</code> )
<b>BackColor</b>	Hintergrundfarbe
<b>BorderColor</b>	Farbe des Rahmens
<b>BorderStyle</b>	Stil des Rahmens
<b>BorderWidth</b>	Breite des Rahmens
<b>CssClass</b>	CSS-Klasse des Steuerelements
<b>Height</b>	Höhe des Steuerelements
<b>Width</b>	Breite des Steuerelements
<b>Enabled</b>	Gibt an, ob das Steuerelement aktiviert ist
<b>Visible</b>	Gibt an, ob das Steuerelement angezeigt werden soll
<b>ToolTip</b>	Text, welcher beim Darüberfahren über das Element angezeigt werden soll

**Wichtig:** Wie bereits erwähnt, werden für einige Steuerelemente JavaScript benötigt. Ist JavaScript im Browser deaktiviert, so funktionieren einige Elemente u. U. nicht mehr korrekt.

## Basiselemente

In ASP.NET gibt es einige Steuerelemente, die sehr ähnlich zu HTML-Elementen sind. Dazu zählen z. B. ein Bild (`asp:Image`), ein Link (`asp:HyperLink`), eine Liste (`asp:BulletedList`), eine Tabelle (`asp:Table`) und ein Textblock (`asp:Label`).

Mit dem Steuerelement `Image` können wir ein **Bild darstellen**. Die URL zum Bild wird mittels der Eigenschaft `ImageUrl` festgelegt. Breite und Höhe können über die Eigenschaften `Width` und `Height` festgelegt werden.

Das Steuerelement `HyperLink` stellt einen **Link / Verweis** dar. Die Eigenschaft `NavigateUrl` gibt die URL an, auf welcher der Link verweist. Mit der Eigenschaft `Target` wird das Zielfenster festgelegt. Als Werte sind hier die gleichen Werte wie beim Attribut `target` des HTML-Elements `a` verfügbar. Mittels der Eigenschaften `ImageUrl` (URL des Bilds), `ImageHeight` (Höhe des Bilds) und `ImageWidth` (Breite des Bilds) wird das Bild für den Hyperlink festgelegt. Die Parameter werden weggelassen, wenn kein Bild für den Link erwünscht ist. Die Eigenschaft `Text` legt den anzuzeigenden Link-Text fest.

`BulletedList` stellt eine **geordnete oder auch ungeordnete Liste** dar (vergleichbar mit dem `ul`- und `ol`-Element in HTML). Der Stil der Liste kann mittels der Eigenschaft `BulletStyle` festgelegt werden. Folgende Werte aus der Enumeration `BulletStyle` sind möglich: `Circle` (leerer Kreis), `Disc` (ausgefüllter Kreis), `Square` (ausgefülltes Quadrat), `LowerAlpha` (kleine lateinische Buchstaben), `LowerRoman` (kleine römische Buchstaben), `UpperAlpha` (große lateinische Buchstaben), `UpperRoman` (große römische Buchstaben), `Numbered` (numerisch), `CustomImage` (Bild, welches über `BulletImageUrl` festgelegt ist) oder `NotSet` (nicht festgelegt, Wahl vom Browser). Mit der Eigenschaft `DisplayMode` kann der „Anzeigemodus“ der Listeneinträge festgelegt werden: `Text` (reiner Text ohne Link), `HyperLink` (URL-Link, Link festgelegt über Eigenschaft `Value` des einzelnen Eintrags) oder `LinkButton` („Server-Link“, durch den Klick wird die Seite „gepostet“ und ein Ereignis ausgelöst). Die Listeneinträge können über die Eigenschaft `Items` abgerufen werden. Innerhalb des HTML-Codes können die Einträge mit dem Element `asp:ListItem` angegeben werden (siehe folgendes Beispiel).

Auf die `Table`-Klasse gehen wir weiter unten noch separat ein.

Das Steuerelement `Label` stellt eine „Beschriftung“ oder allgemein gesagt einen Textblock dar. Der Inhalt bzw. die **Beschriftung** kann mittels der Eigenschaft `Text` festgelegt werden.

Im folgenden Beispiel werden die Steuerelemente `Image`, `HyperLink` und `BulletedList` verwendet. Für das `BulletedList`-Element wurde ein `Click`-

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Webserver-Steuerelemente](#)

Ereignis (Attribut `OnClick`) registriert, wodurch das angeklickte Element rot gefärbt und das ausgewählte Element angezeigt wird.

```

1 <form runat="server">
2   <asp:Image ImageUrl="ASP-NET.png" Width="100" Height="100" runat="server" />
3   <br />
4   <asp:HyperLink Text="Zum ASP.NET Kapitel" NavigateUrl="https://www.homepage-webhilfe.de/ASP-NET/" Target="_blank"
5   runat="server" />
6   <br />
7   <asp:BulletedList ID="AuswahlListe" BulletStyle="Square" DisplayMode="LinkButton" OnClick="AuswahlListe_Click"
8   runat="server">
9     <asp:ListItem Value="PHP" />
10    <asp:ListItem Value="Perl" />
11    <asp:ListItem Value="ASP.NET" />
12    <asp:ListItem Value="Java EE" />
13  </asp:BulletedList>
14  <asp:Label ID="AuswahlAnzeige" Text="Sie haben keine Auswahl getroffen!" runat="server" />
15 </form>
16
17 using System;
18 using System.Web.UI.WebControls;
19
20 namespace HwhBsp.Basiselemente
21 {
22   public partial class Default : System.Web.UI.Page
23   {
24     protected void Page_Load(object sender, EventArgs e)
25     {
26     }
27
28     protected void AuswahlListe_Click(object sender, BulletedListEventArgs e)
29     {
30       AuswahlListe.Items[e.Index].Attributes.CssStyle.Add("color", "red");
31       AuswahlAnzeige.Text = "Sie haben auf das " + (e.Index + 1) + ". Element geklickt";
32     }
33   }
34 }

```



## Formularelemente

Die Formularelemente von ASP.NET sind weitestgehend selbsterklärend und vergleichbar mit denen, welche wir vom „reinen“ HTML bereits kennen. Trotzdem wollen wir auf diese noch kurz etwas eingehen. Als Elemente sind `asp:TextBox` (Textfeld), `asp:CheckBox` (Kontrollkästchen), `asp:RadioButton` (Element einer Auswahlgruppe), `asp:DropDownList` (Auswahlliste) und `asp:Button` (Schaltfläche) zu nennen. Für das **Textfeld** kann mittels der Eigenschaft `ReadOnly` festgelegt werden, ob der Inhalt des Textfelds schreibgeschützt ist. Die Eigenschaft `Wrap` steuert den automatischen Zeilenumbruch. Mittels der Eigenschaft `TextMode` kann festgelegt werden, ob es sich um ein einzelzeiliges Textfeld (`SingleLine`), ein mehrzeiliges Textfeld (`MultiLine`), ein Passwort (`Password`) oder ein anderes spezielles Textfeld (z. B. `Date` für Datumseingabe oder `Time` für Uhrzeiteingabe) handelt. Für ein **Kontrollkästchen** kann mittels der Eigenschaft `Checked` abgerufen oder festgelegt werden, ob das Kontrollkästchen ausgewählt wurde. `Radio-Buttons` (**Auswahlgruppen**) werden über die Eigenschaft `GroupName` gruppiert, können jedoch alle eine individuelle ID besitzen, um somit vom Server separiert darauf zuzugreifen (siehe Beispiel). Um abzurufen, ob ein Element einer Auswahlgruppe ausgewählt wurde, kann die Eigenschaft `Checked` verwendet werden. Der Wert des Textfelds, des Kontrollkästchens oder des Elements der Auswahlgruppe kann über die Eigenschaft `Text` abgerufen werden. Eine **Auswahlliste** enthält Elemente des Typs `ListItem`. Diese verfügen über die Eigenschaften `Text` (anzeigende Beschriftung) und `Value` („interner“ Wert). Wird auf eine **Schaltfläche** geklickt, so wird die Seite „gepostet“. Die vier genannten Webserver-Steuerelemente verfügen auch über ein paar Ereignisse: `TextChanged` (für Textfelder), `CheckedChanged` (für Kontrollkästchen und Elemente einer Auswahlgruppe), `SelectedIndexChanged` (für Auswahllisten) oder `Click` (für eine Schaltfläche). Hierzu nun ein etwas komplexeres Beispiel:

```

1 <form action="Default.aspx" method="post" runat="server">
2   <table>
3     <tr>
4       <td style="width: 100px;">Anrede:</td>
5       <td>
6         <asp:RadioButton ID="AnredeH" GroupName="Anrede" Text="Herr" Checked="true" runat="server" />
7         <asp:RadioButton ID="AnredeF" GroupName="Anrede" Text="Frau" runat="server" />
8       </td>
9     </tr>
10    <tr>
11     <td style="width: 100px;">Vorname:</td>
12     <td><asp:TextBox ID="Vorname" runat="server" /></td>
13   </tr>
14   <tr>
15     <td style="width: 100px;">Nachname:</td>
16     <td><asp:TextBox ID="Nachname" runat="server" /></td>
17   </tr>
18   <tr>
19     <td style="width: 100px;">Grund:</td>
20     <td>
21       <asp:DropDownList ID="Grund" runat="server">
22         <asp:ListItem Value="PHP" Text="PHP" />

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

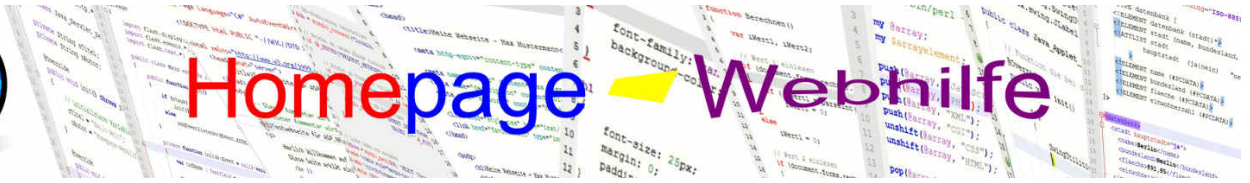
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Webserver-Steuerelemente](#)

```

23         <asp:ListItem Value="Perl" Text="Perl" />
24         <asp:ListItem Value="ASP" Text="ASP.NET" />
25         <asp:ListItem Value="Java" Text="Java EE" />
26     </asp:DropDownList>
27 </td>
28 </tr>
29 <tr>
30 <td style="width: 100px;">Nachricht:</td>
31 <td><asp:TextBox ID="Nachricht" TextMode="MultiLine" runat="server" /></td>
32 </tr>
33 <tr>
34 <td style="width: 100px;">Newsletter:</td>
35 <td><asp:CheckBox ID="Newsletter" runat="server" /></td>
36 </tr>
37 <tr>
38 <td></td>
39 <td><asp:Button ID="BeispielBtn" Text="Beispielwerte" OnClick="BeispielBtn_Click" runat="server" /></td>
40 </tr>
41 <tr>
42 <td></td>
43 <td><asp:Button ID="PruefBtn" Text="Prüfen" OnClick="PruefBtn_Click" runat="server" /></td>
44 </tr>
45 <tr>
46 <td></td>
47 <td><asp:Button ID="VersandBtn" Text="Versenden" OnClick="VersandBtn_Click" Enabled="false" runat="server"
/></td>
48 </tr>
49 </table>
50 </form>
1 using System;
2 using System.Drawing;
3
4 namespace HwhBsp.Formularelemente
5 {
6     public partial class Default : System.Web.UI.Page
7     {
8         protected void Page_Load(object sender, EventArgs e)
9         {
10
11         }
12
13         protected void BeispielBtn_Click(object sender, EventArgs e)
14         {
15             // Felder mit Beispielwerten füllen
16             AnredeF.Checked = false;
17             AnredeH.Checked = true;
18             Vorname.Text = "Max";
19             Nachname.Text = "Mustermann";
20             Grund.SelectedValue = "ASP";
21             Nachricht.Text = "Hier steht der Inhalt der Nachricht ...";
22             Newsletter.Checked = false;
23
24             // Prüf-Funktion aufrufen
25             PruefBtn_Click(null, EventArgs.Empty);
26         }
27
28         protected void PruefBtn_Click(object sender, EventArgs e)
29         {
30             bool bError = false;
31
32             // Überprüfen ob Eingabefelder leer sind
33             if (Vorname.Text == "")
34             {
35                 Vorname.BackColor = Color.Red;
36                 bError = true;
37             }
38             else
39                 Vorname.BackColor = Color.Transparent;
40             if (Nachname.Text == "")
41             {
42                 Nachname.BackColor = Color.Red;
43                 bError = true;
44             }
45             else
46                 Nachname.BackColor = Color.Transparent;
47             if (Nachricht.Text == "")
48             {
49                 Nachricht.BackColor = Color.Red;

```

**Über uns**

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

**Community**

- Blog
- Forum
- News

**Nachschlagewerk**

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Webserver-Steuerelemente](#)

```

50         bError = true;
51     }
52     else
53         Nachricht.BackColor = Color.Transparent;
54
55     // Versand-Button aktivieren oder deaktivieren
56     VersandBtn.Enabled = !bError;
57 }
58
59 protected void VersandBtn_Click(object sender, EventArgs e)
60 {
61     // Formularfelder deaktivieren
62     AnredeH.Enabled = false;
63     AnredeF.Enabled = false;
64     Vorname.Enabled = false;
65     Nachname.Enabled = false;
66     Grund.Enabled = false;
67     Nachricht.Enabled = false;
68     Newsletter.Enabled = false;
69     BeispielBtn.Enabled = false;
70     PruefBtn.Enabled = false;
71     VersandBtn.Enabled = false;
72 }
73 }
74 }
    
```



**Wichtig:** Beachten Sie, dass Ereignisse nicht direkt ausgelöst werden. Ist dieser Effekt erwünscht (dadurch wird auch die Seite neu geladen), so kann die Eigenschaft `AutoPostBack` auf `true` gesetzt werden.

**Übrigens:** Eine Auswahlgruppe kann an Stelle mit `RadioButton`-Steuerelementen auch mit einem `RadioButtonList`-Steuerelement und mehreren `ListItems`-Elementen gelöst werden.

## Kalender



Ausgewähltes Datum: Dienstag, 25. Oktober 2016

Mit dem Webserver-Steuerelement `Calendar` wird uns eine Auswahl für einen Tag (`Day`), einen Tag oder eine Woche (`DayWeek`) oder einen Tag, eine Woche oder einen Monat (`DayWeekMonth`) zur Verfügung gestellt. Dieser Selektierungsmodus kann mittels der Eigenschaft `SelectionMode` festgelegt werden. Die Auswahl spiegelt sich grundsätzlich in Form eines Kalenders wieder. Der Kalender besitzt für die Darstellung diverse Eigenschaften, welche sich in die Gruppen aktuelles Datum (`TodayDayStyle`), ausgewähltes Datum (`SelectedDayStyle`), Wochenende (`WeekendDayStyle`), anderer Monat (`OtherMonthDayStyle`) und anderer Tag (`DayStyle`) aufteilen lässt. Das Auswählen eines Tags, einer Woche oder eines Monats hat zur Folge, dass die Seite neu geladen wird. Das selektierte Datum bzw. die selektierten Daten lassen sich über `SelectedDate` bzw. `SelectedDates` abrufen oder setzen. Als Ereignisse sind `SelectionChanged` (Auswahl wurde geändert) und `VisibleMonthChanged` (Monat, welcher angezeigt wird, wurde geändert).

```

1 <form runat="server">
2     <asp:Calendar ID="Kalender" SelectedDayStyle-BackColor="Blue" OtherMonthDayStyle-BackColor="Yellow" WeekendDayStyle-
   BackColor="Orange" TodayDayStyle-BackColor="Red" runat="server" />
3     <p>Ausgewähltes Datum: <%= Kalender.SelectedDate != DateTime.MinValue ? Kalender.SelectedDate.ToLongDateString() : "-"
   %></p>
4 </form>
    
```



## Tabelle

Mit der Klasse `Table` und dem dazugehörigen Steuerelement können Sie eine Tabelle darstellen. Vorteil dieses Webserver-Steuerelements gegenüber des klassischen HTML-Elements `table` ist, dass der Zugriff auf die Zeilen und Zellen leichter ist und somit die sogenannte Manipulation schneller und einfacher erfolgen kann. Der Aufbau dieser ASP.NET-Tabelle ist gleich wie der, der HTML-Tabelle: es gibt Zeilen (`TableRow`), welche eine oder mehrere Zellen (`TableCell`) enthalten können. Des Weiteren existieren die Steuerelemente `TableHeaderRow` (Kopfzeile), `TableFooterRow` (Fußzeile) und `TableHeaderCell` (Header-Zelle). Im Beispiel wird der Inhalt der Tabelle dynamisch über ein zwei-dimensionales Array im `Load`-Ereignis der Seite gefüllt:

```

1 <form runat="server">
2     <asp:Table ID="KundenListe" runat="server">
3         <asp:TableHeaderRow>
4             <asp:TableCell Text="ID" />
5             <asp:TableCell Text="Vorname" />
6             <asp:TableCell Text="Nachname" />
7         </asp:TableHeaderRow>
8     </asp:Table>
9 </form>
10
11 using System;
12 using System.Web.UI.WebControls;
13
14 namespace HWhBsp.Tabelle
15 {
16     public partial class Default : System.Web.UI.Page
    
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

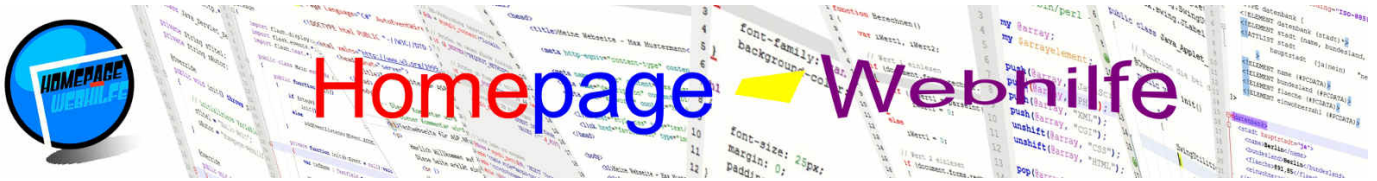
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Webserver-Steuerelemente](#)

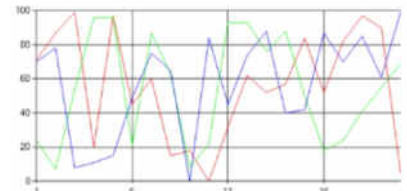
```

7     {
8         protected void Page_Load(object sender, EventArgs e)
9         {
10            string[][] aKundenListe = new string[][] {
11                new string[] { "0", "Steve", "Peters" },
12                new string[] { "1", "Lisa", "Meyer" },
13                new string[] { "2", "Mark", "Kirsch" },
14                new string[] { "3", "Nathalie", "Lehmann" },
15                new string[] { "4", "Ulrike", "Wagner" }
16            };
17
18            // Zeilen hinzufügen
19            for (int i = 0; i < aKundenListe.Length; i++)
20            {
21                TableRow oRow = new TableRow();
22                // Zellen hinzufügen
23                for (int j = 0; j < aKundenListe[i].Length; j++)
24                {
25                    TableCell oCell = new TableCell();
26                    oCell.Text = aKundenListe[i][j];
27                    oRow.Cells.Add(oCell);
28                }
29                KundenListe.Rows.Add(oRow);
30            }
31        }
32    }
33 }
    
```



## Diagramm

Das Steuerelement `asp:Chart` macht es möglich, **ohne weitere externe Libraries** ein Diagramm darzustellen. Innerhalb des `Chart`-Elements werden weitere Elemente angegeben, um die Diagrammbereiche (`ChartAreas`), Datenreihen (`Series`) und Legenden (`Legends`) zu definieren. Natürlich kann dies auch über die Code-Behind-Datei gemacht werden. Der Aufbau der Eigenschaften und Klassen ist dabei den Diagramm-Klassen für Windows Forms Applikationen (Namensraum `System.Windows.Forms.DataVisualization.Charting`) ähnlich. Im Beispiel gibt es einen Diagrammbereich mit drei Datenreihen. Jede Datenreihe besitzt 20 Werte, welche dynamisch generiert (Klasse `Random`) werden.



```

1 <form runat="server">
2     <asp:Chart ID="Diagramm" runat="server" Width="600" Height="300">
3         <Series>
4             <asp:Series ChartType="Line" Color="Red" Name="SerieA"></asp:Series>
5             <asp:Series ChartType="Line" Color="Lime" Name="SerieB"></asp:Series>
6             <asp:Series ChartType="Line" Color="Blue" Name="SerieC"></asp:Series>
7         </Series>
8         <ChartAreas>
9             <asp:ChartArea Name="Bereich">
10                <AxisX IsMarginVisible="false" />
11            </asp:ChartArea>
12        </ChartAreas>
13    </asp:Chart>
14 </form>
    
```

```

1 using System;
2
3 namespace HwhBsp.Diagramm
4 {
5     public partial class Default : System.Web.UI.Page
6     {
7         protected void Page_Load(object sender, EventArgs e)
8         {
9             Random oZufall = new Random();
10            for (int i = 0; i < 20; i++)
11            {
12                Diagramm.Series["SerieA"].Points.Add(oZufall.Next(0, 101));
13                Diagramm.Series["SerieB"].Points.Add(oZufall.Next(0, 101));
14                Diagramm.Series["SerieC"].Points.Add(oZufall.Next(0, 101));
15            }
16        }
17    }
18 }
    
```



**Wichtig:** Um ein `Chart`-Element nutzen zu können, muss innerhalb der ASPX-Datei ein Assembly „registriert“ werden (Direktive `Register`). Dies sieht wie folgt

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

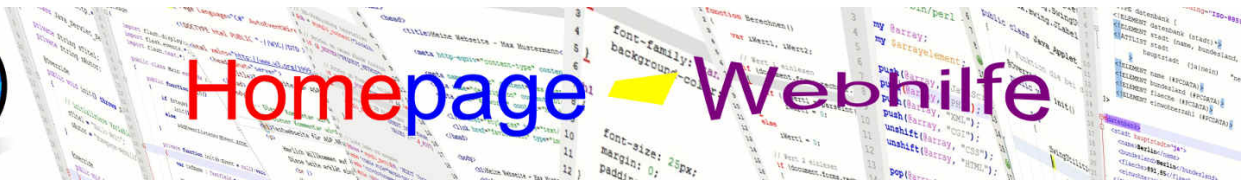
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Webserver-Steuerelemente](#)

aus:

```
1 <%@ Register Assembly="System.Web.DataVisualization, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"
  Namespace="System.Web.UI.DataVisualization.Charting" TagPrefix="asp" %>
```

Des Weiteren muss das Assembly in der Konfigurationsdatei `Web.config` hinzugefügt werden. Zudem muss ein „Handler“ hinzugefügt werden. Die Konfigurationsdatei von unserem Beispiel sieht wie folgt aus:

```
1 <?xml version="1.0"?>
2
3 <configuration>
4   <system.web>
5     <compilation debug="true" targetFramework="4.0">
6       <assemblies>
7         <add assembly="System.Web.DataVisualization, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
8       </assemblies>
9     </compilation>
10  </system.web>
11  <system.webServer>
12    <handlers>
13      <add name="ChartImg" verb="*" path="ChartImg.axd" type="System.Web.UI.DataVisualization.Charting.ChartHttpHandler,
  System.Web.DataVisualization, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
14    </handlers>
15  </system.webServer>
16 </configuration>
```

### Validierung

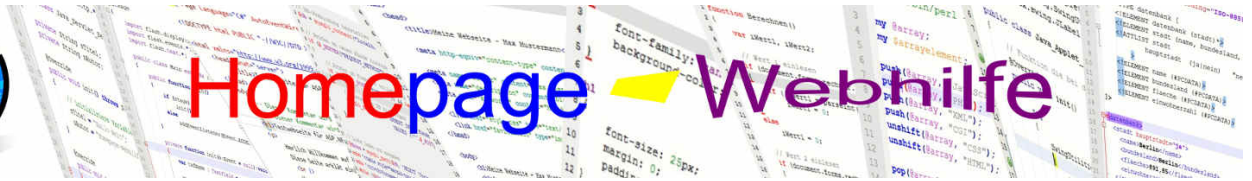
ASP.NET bietet ein paar Webserver-Steuerelemente, welche dazu genutzt werden, den **Inhalt oder Wert eines anderen Feldes zu prüfen**. Für diese sogenannte „Validierungs-Steuerelemente“ wird im Browser **JavaScript-Code** ausgeführt. Die Validierung erfolgt jedoch auch **zusätzlich auf dem Webserver** (dazu gleich mehr). Die folgenden Validierungs-Steuerelemente wollen wir hier detaillierter erklären: `RequiredFieldValidator`, `RangeValidator` und `RegularExpressionValidator`. Alle Validierungs-Elemente besitzen die Eigenschaft `ControlToValidate`, mit welcher das zu überwachende Steuerelement festgelegt wird, und `ErrorMessage`, mit welcher der anzuzeigende Fehlertext angegeben wird. Die Überprüfung kann mittels der Eigenschaft `Enabled` deaktiviert werden. Über die Eigenschaft `Display` und einen Wert der Enumeration `ValidatorDisplay` wird das sogenannte **Anzeigeverhalten** festgelegt. Folgende Werte sind verfügbar: `Dynamic` (Meldung wird je nach Bedarf dynamisch ins Layout hinzugefügt oder wieder entfernt), `Static` (Meldung ist fester Bestandteil des Seitenlayouts und wird lediglich ein- und ausgeblendet) und `None` (Meldung wird nie inline angezeigt).

Für `RequiredFieldValidator` müssen keine weiteren Eigenschaften gesetzt werden. Die Fehlermeldung wird ausgegeben, wenn das verwiesene **Feld nicht ausgefüllt wurde**. Für das Steuerelement `RangeValidator` müssen noch die Eigenschaften `MinimumValue` und `MaximumValue` als **Unter- und Obergrenze** für den zulässigen Wertebereich gesetzt werden. Das Webserver-Steuerelement `RegularExpressionValidator` führt eine Validierung mittels eines **regulären Ausdrucks** aus. Dieser wird in der Eigenschaft `ValidationExpression` angegeben.

Die Validierung dieser Steuerelemente wird sowohl client- als auch serverseitig durchgeführt. Die clientseitige Überprüfung (und Anzeige der Fehlermeldungen) wird automatisch ausgeführt, sofern JavaScript im Browser aktiviert ist. Um eine **serverseitige Validierung** durchzuführen, muss als erstes die Funktion `Validate()` der Seite aufgerufen werden. Anschließend kann über die Eigenschaft `IsValid` geprüft werden, ob das Formular gültig ist. Diese Eigenschaft ist auch bei den Validator-Steuerelementen verfügbar, um somit ein einzelnes Steuerelement bzw. eine einzelne Regel auf Gültigkeit zu prüfen.

```
1 <form action="Default.aspx" method="post" runat="server">
2   <%
3     // Eingaben nur prüfen, wenn das Formular übertragen wurde
4     if (this.Request.HttpMethod == "POST")
5     {
6       Validate();
7       if (!this.IsValid)
8         Response.Write("Ihre Eingaben enthalten Fehler!");
9     }
10    else
11      Response.Write("Ihre Eingaben sind gültig!");
12  }
13  %>
14  <table>
15    <tr>
16      <td style="width: 100px;">Alter:</td>
17      <td>
18        <asp:TextBox ID="Alter" runat="server" />
19        <asp:RangeValidator ControlToValidate="Alter" MinimumValue="12" MaximumValue="40" ErrorMessage="Sie müssen
  zwischen 12 und 40 Jahre alt sein!" Type="Integer" Display="Dynamic" runat="server" />
20      </td>
21    </tr>
22    <tr>
23      <td style="width: 100px;">E-Mail-Adresse:</td>
24      <td>
25        <asp:TextBox ID="EmailAdresse" runat="server" />
26        <asp:RequiredFieldValidator ControlToValidate="EmailAdresse" ErrorMessage="Bitte geben Sie eine E-Mail-
  Adresse ein!" Display="Dynamic" runat="server" />
27        <asp:RegularExpressionValidator ControlToValidate="EmailAdresse" ErrorMessage="Das Format der E-Mail-Adresse
  ist ungültig!" ValidationExpression="[a-zA-Z0-9.!#$%&'*/-/?^_`{|}~]{1,}@[a-zA-Z0-9.!#$%&'*/-/?^_`{|}~]{1,}].[a-zA-
  Z0-9.!#$%&'*/-/?^_`{|}~]{1,}" Display="Dynamic" runat="server" />
28      </td>
29    </tr>
30  </table>
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Webserver-Steuerelemente](#)

```

28     </tr>
29     <tr>
30         <td></td>
31         <td><input type="submit" value="Absenden" /></td>
32     </tr>
33 </table>
34 </form>
    
```



## Ansichten

In ASP.NET können mittels der Webserver-Steuerelemente `asp:MultiView` und `asp:View` mehrere Anzeigen erzeugt werden, zwischen welchen umgeschaltet werden kann. Die **Umschaltung der Ansichten ist nur serverseitig möglich**, da an den Client lediglich der Inhalt der gewählten Ansicht gesendet wird. Im Beispiel kann mittels der Auswahlgruppe die Ansicht ausgewählt werden. Diese Selektierung löst das `SelectedIndexChanged`-Ereignis aus, wodurch die angegebene Ansicht selektiert wird. Über das `ActiveViewChanged`-Event, welches bei Änderung der Ansicht eintritt, wird in ein `Label`-Steuerelement die Nummer der Ansicht geschrieben. Die Selektierung einer Ansicht kann entweder mit der Funktion `SetActiveView()` oder über die Eigenschaft `ActiveViewIndex` erfolgen.

```

1  <form runat="server">
2      <asp:Label ID="AktualIndex" Text="Aktuelle Ansicht: 0" runat="server" />
3      <asp:RadioButtonList ID="SeitenSelektierung" AutoPostBack="true"
4          OnSelectedIndexChanged="SeitenSelektierung_SelectedIndexChanged" runat="server">
5          <asp:ListItem Text="Seite A (rot)" Value="0" Selected="True" />
6          <asp:ListItem Text="Seite B (blau)" Value="1" />
7          <asp:ListItem Text="Seite C (grün)" Value="2" />
8      </asp:RadioButtonList>
9      <asp:MultiView ID="Tabs" OnActiveViewChanged="Tabs_ActiveViewChanged" runat="server">
10         <asp:View ID="TabA" runat="server">
11             <div style="background-color: red; width: 400px; height: 300px"></div>
12         </asp:View>
13         <asp:View ID="TabB" runat="server">
14             <div style="background-color: blue; width: 400px; height: 300px"></div>
15         </asp:View>
16         <asp:View ID="TabC" runat="server">
17             <div style="background-color: lime; width: 400px; height: 300px"></div>
18         </asp:View>
19     </asp:MultiView>
20 </form>
21
22 using System;
23
24 namespace HWhBsp.Ansichten
25 {
26     public partial class Default : System.Web.UI.Page
27     {
28         protected void Page_Load(object sender, EventArgs e)
29         {
30             Tabs.SetActiveView(TabA);
31         }
32
33         protected void SeitenSelektierung_SelectedIndexChanged(object sender, EventArgs e)
34         {
35             Tabs.ActiveViewIndex = Convert.ToInt32(SeitenSelektierung.Selected.Value);
36         }
37
38         protected void Tabs_ActiveViewChanged(object sender, EventArgs e)
39         {
40             AktualIndex.Text = "Aktuelle Seite: " + Tabs.ActiveViewIndex;
41         }
42     }
43 }
    
```



**Wichtig:** Der Aufruf der Funktion `SetActiveView()` im Event-Handler des `Load`-Ereignisses der Seite ist notwendig, um eine Seite zu selektieren, falls noch keine Ansicht gewählt wurde. Wird eine Ansicht über die Auswahlgruppe ausgewählt, so wird also zuerst die 1. Ansicht selektiert (über das `Load`-Ereignis) und anschließend die gewünschte Ansicht (über das `SelectedIndexChanged`-Ereignis).

**Übrigens:** Um HTML-Elemente, HTML-Server-Steuerelemente und Webserver-Steuerelemente zu gruppieren, kann das Webserver-Steuerelement `asp:Panel` genutzt werden. Dort gibt es jedoch keine besonderen zu erwähnenden Eigenschaften oder Ereignisse.

## AJAX-Technologie



Mittels der AJAX-Technologie (welche wir bereits im [Kapitel JavaScript](#) erläutert haben) ist es möglich, **Seitenbestandteile zur Laufzeit nachzuladen oder diese zu aktualisieren**. In ASP.NET wird uns diese Technologie sehr vorteilhaft angeboten. Als Steuerelemente stehen uns hier `ScriptManager`, `UpdatePanel` und `Timer` zur Verfügung. Das Steuerelement `ScriptManager` muss lediglich einmal deklariert werden und dient dazu das **AJAX-Modul einzubinden**. Mit dem Steuerelement `UpdatePanel` kann ein Bereich (Seitenabschnitt) definiert werden, welcher bei einem AJAX-Ereignis neu geladen werden soll. Hierfür wird innerhalb des Elements ein Element mit dem Tagnamen `ContentTemplate` (entspricht der gleichnamigen Eigenschaft) notiert. In diesem können wiederum beliebig viele Elemente

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Webserver-Steuerelemente](#)

(sowohl HTML- als auch ASP.NET-Elemente) angegeben werden. Der Inhalt dieses Panels kann nun über verschiedene **Trigger** verändert werden. Tritt einer der Trigger ein, so wird eine Anfrage an den Server gesendet und der **Inhalt des Abschnitts komplett neu geladen**. Da diese Anfrage jedoch mittels der AJAX-Technologie und somit „versteckt“ bzw. im Hintergrund ausgeführt wird, wird dieses Laden durch den Besucher zumeist nicht registriert. Das (eintellige) Element `Timer` wird als Trigger für das `UpdatePanel` benutzt, um somit den Inhalt eines **Seitenabschnitts zyklisch zu aktualisieren**. Hierfür wird im `Timer`-Element die Eigenschaft `Interval` (Zeitabstand in Millisekunden) sowie das Ereignis `Tick` gesetzt. Um den Trigger im `UpdatePanel`-Element zu registrieren, muss das Element `AsyncPostBackTrigger` innerhalb der Eigenschaft `Triggers` (siehe Beispiel) platziert werden. Der Trigger wird über die Eigenschaften `ControlID` (Name des Elements, in diesem Fall des Timers) und `EventName` (Name des entsprechenden Events, in diesem Fall `Tick`) an ein Ereignis gebunden. Das Beispiel zeigt zwei `UpdatePanel`-Steuerelemente, wovon das erste sekundlich aktualisiert wird und das zweite lediglich beim Klick auf den Button.

```

1  <form runat="server">
2  <asp:ScriptManager ID="AjaxManager" runat="server" />
3
4  <!-- Zufallszahlen-Aktualisierung (sekundlich durch Timer) -->
5  <asp:Timer ID="SekundenTimer" Interval="1000" OnTick="SekundenTimer_Tick" runat="server" />
6  <asp:UpdatePanel runat="server">
7      <Triggers>
8          <asp:AsyncPostBackTrigger ControlID="SekundenTimer" EventName="Tick" />
9      </Triggers>
10     <ContentTemplate>
11         <b>Zufallszahl:</b>
12         <br />
13         <asp:Label ID="ZufallsWert" Text="-" runat="server" />
14     </ContentTemplate>
15 </asp:UpdatePanel>
16
17 <br />
18
19 <!-- Aktualisierung beim Buttonklick -->
20 <asp:UpdatePanel runat="server">
21     <ContentTemplate>
22         <b>Letzte Aktualisierung:</b>
23         <br />
24         <asp:Label ID="LetzteAktualisierung" Text="-" runat="server" />
25         <br />
26         <asp:Button ID="ZeitButton" Text="Jetzt aktualisieren" OnClick="ZeitButton_Click" runat="server" />
27     </ContentTemplate>
28 </asp:UpdatePanel>
29 </form>

```

```

1  using System;
2
3  namespace HWhBsp.AJAX_Technologie
4  {
5      public partial class Default : System.Web.UI.Page
6      {
7          private Random oZufall = new Random();
8
9          protected void Page_Load(object sender, EventArgs e)
10         {
11         }
12
13         protected void SekundenTimer_Tick(object sender, EventArgs e)
14         {
15             ZufallsWert.Text = oZufall.Next(1, 1001).ToString();
16         }
17
18         protected void ZeitButton_Click(object sender, EventArgs e)
19         {
20             LetzteAktualisierung.Text = DateTime.Now.ToString();
21         }
22     }
23 }
24

```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » ASP.NET » ASP.NET WebForms » Ansichtszustand, Session und Cookies

## Ansichtszustand, Session und Cookies



Immer wieder werden Sie in die Situation kommen, dass Sie **Daten des aktuellen Besuchers speichern** möchten. Diese Daten sollen dann z. B. nur für diese Seite, nur für diese Sitzung oder sogar länger gelten. Wichtig dabei ist jedoch immer, dass die Daten eben nur für diesen einen bestimmten Besucher gelten. In PHP haben Sie dafür Sessions und Cookies kennengelernt. Diese gibt es auch in ASP.NET. Zudem bietet uns ASP.NET die Möglichkeit, Daten im sogenannten Ansichtszustand zu speichern. In diesem Thema wollen wir Ihnen die verschiedenen Speicherungsmöglichkeiten erklären und mit einem Beispiel verdeutlichen.

### Inhalt dieser Seite:

1. Ansichtszustand
2. Session
3. Cookies

### Ansichtszustand

Mit dem Ansichtszustand (engl. *view state*) ist es möglich, **Daten auf der Seite zu speichern**. Diese Methode wird unter anderem auch dazu verwendet, die Werte und Ereignisstatusse von Steuerelementen nach einem Postback beizubehalten. Diese Speicherung geschieht jedoch bereits automatisch. Die Daten des Ansichtszustands werden in **versteckten Eingabefeldern** gespeichert und mit BASE-64 enkodiert. Diese Zeichenkette kann ohne viel Aufwand dekodiert, manipuliert und wieder zurück enkodiert werden. Speichern Sie hier also niemals sicherheitsrelevante Daten. Im Ansichtszustand können Daten innerhalb von **Variablen** gespeichert werden. Um auf diese Variablen zuzugreifen, können wir die Eigenschaft `ViewState` (des aktuellen `Page`-Objekts) nutzen. Der lesende Zugriff erfolgt mittels den eckigen Klammern `[]` (wie bei Arrays). Existiert eine Variable nicht, wird `null` zurückgegeben. Dieser Syntax kann natürlich auch verwendet werden, wenn ein Wert überschrieben werden soll. Hier sollte jedoch bevorzugt die Funktion `Add()` verwendet werden, da diese auch dazu genutzt werden kann, eine neue Variable zu deklarieren. Existiert die Variable schon, so ersetzt die Funktion `Add()` den Wert der Variablen durch den neuen, welcher angegeben wurde. Als Parameter übergeben wir der Funktion den Namen der Variable und den Wert (dabei kann es sich um eine Zeichenkette, eine Zahl oder einen anderen Wert handeln). Mittels der Funktion `Remove()` kann eine bestimmte Variable des Ansichtszustands wieder entfernt werden. Im Beispiel wird eine Zufallszahl generiert und diese im Ansichtszustand gespeichert. Beim Klicken auf den Button wird die Seite „gepostet“ und eine neue Zufallszahl generiert und angezeigt. Zudem wird die vorherige Zahl ausgelesen und ebenfalls angezeigt. Dieser Vorgang kann mehrmals wiederholt werden, wovon immer die letzte Zufallszahl „erhalten“ bleibt.

```
1 <form runat="server">
2   <b>Aktuelle Zufallszahl:</b> <%= iZufallAktuell %><br />
3   <b>Letzte Zufallszahl:</b> <%= sZufallLetzte %><br />
4   <br />
5   <input type="submit" value="Seite erneut aufrufen" />
6 </form>

1 using System;
2
3 namespace HwhBsp.Ansichtszustand
4 {
5     public partial class Default : System.Web.UI.Page
6     {
7         protected int iZufallAktuell;
8         protected string sZufallLetzte;
9
10        protected void Page_Load(object sender, EventArgs e)
11        {
12            iZufallAktuell = (new Random()).Next(1, 1001);
13            sZufallLetzte = ViewState["LetzterZufall"] != null ? (string)ViewState["LetzterZufall"] : "-";
14            ViewState.Add("LetzterZufall", iZufallAktuell.ToString());
15        }
16    }
17 }
```



**Wichtig:** Wenn Sie die Seite „manuell“ erneut aufrufen (z. B. durch erneute Eingabe der URL oder durch klicken auf einen Link), so sind die Daten des Ansichtszustands verloren. Die Daten bleiben also lediglich zwischen Postbacks erhalten.

### Session

Session-Variablen (zu Deutsch Sitzungs-Variablen) ermöglichen es, **Daten über eine Seite hinaus zu speichern**. Technisch realisiert werden Sessions mit einem Cookie, welches eine Dauer von 0 hat und somit bis zum Schließen des Browsers gültig ist. Dadurch sind auch die Daten bis zum Schließen des Browsers gültig. Die Session-Variablen werden **auf dem Server gespeichert**. Im Browser wird lediglich eine **Session-ID** (in Form eines Cookies, wie oben beschrieben) gespeichert. Der Browser schickt bei jeder Anfrage diese Session-ID zurück an den Webserver, wodurch dieser die Variablen, welche zur Session gehören, zuordnen kann. Der Zugriff auf Session-Variablen ist vergleichbar mit dem für den Ansichtszustand. Als Eigenschaft wird hier `Session` verwendet. `Session` enthält ein Objekt, bei welchem auch über die Indizierung (mittels einem eckigen Klammerspaar) zugegriffen werden kann. Die Funktionen `Add()` und `Remove()` sind hier ebenfalls verfügbar. Des Weiteren gibt es hier die Funktion `RemoveAll()`, mit welcher alle Variablen der Sitzung gelöscht werden können. Das folgende Beispiel ist mit dem Ansichtszustands-Beispiel vergleichbar, speichert jedoch die Zufallszahl auch über einen erneuten (manuellen) Seitenaufruf hinaus.

```
1 <form runat="server">
2   <b>Aktuelle Zufallszahl:</b> <%= iZufallAktuell %><br />
3   <b>Letzte Zufallszahl:</b> <%= sZufallLetzte %><br />
4   <br />
5   <input type="submit" value="Seite erneut aufrufen" />
6 </form>

1 using System;
2
3 namespace HwhBsp.Session
```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Ansichtszustand, Session und Cookies](#)

```

4  {
5  public partial class Default : System.Web.UI.Page
6  {
7  protected int iZufallAktuell;
8  protected string sZufallLetzte;
9
10 protected void Page_Load(object sender, EventArgs e)
11 {
12     iZufallAktuell = (new Random()).Next(1, 1001);
13     sZufallLetzte = Session["LetzterZufall"] != null ? (string)Session["LetzterZufall"] : "-";
14     Session.Add("LetzterZufall", iZufallAktuell.ToString());
15 }
16 }
17 }

```



## Cookies

Cookies erlauben das **Speichern von Informationen über eine Sitzung hinaus**. Cookies haben eine bestimmte **Gültigkeitsdauer**, welche beim Setzen des Cookies angegeben werden muss. Des Weiteren kann ein Cookie auf einen **Pfad** beschränkt werden. Cookies werden vom Server „geschrieben“, vom Browser gespeichert und verwaltet und bei einer Anfrage an den Server wieder zurück an den Server übertragen. Durch die Tatsache, dass die Cookie-Werte im Browser gespeichert werden, können diese Werte ebenfalls manipuliert werden. Speichern Sie also auch hier niemals sicherheitsrelevante Daten. Cookies werden in ASP.NET durch ein Objekt der Klasse `HttpCookie` repräsentiert. Die Eigenschaft `Name` legt den Namen fest, `Value` den Wert, `Path` den Pfad, in welchem das Cookie gültig ist (gilt auch für dessen Unterverzeichnisse), und `Expires` den Ablaufzeitpunkt, ab wann das Cookie nicht mehr gültig ist (und somit vom Browser verworfen wird). Dem Konstruktor der Klasse `HttpCookie` kann der Name und Wert des Cookies übergeben werden, wovon der Parameter für den Wert optional ist. Die Cookies, welche vom Browser gesendet und vom Server empfangen wurden, können über die Eigenschaft `Cookies` der `HttpRequest`-Klasse abgerufen werden. Ein Zugriff auf die (neuen) Cookies, welche „geschrieben“ bzw. „gesetzt“ werden sollen, erhalten Sie über die Eigenschaft `Cookies` der `HttpResponse`-Klasse. Um ein neues Cookie zu „deklarieren“, können wir die Funktion `Add()` verwenden.

```

1  <form runat="server">
2  <b>Aktuelle Zufallszahl:</b> <%= iZufallAktuell %><br />
3  <b>Letzte Zufallszahl:</b> <%= sZufallLetzte %><br />
4  <br />
5  <input type="submit" value="Seite erneut aufrufen" />
6  </form>

```

```

1  using System;
2  using System.Web;
3
4  namespace HWhBsp.Cookies
5  {
6  public partial class Default : System.Web.UI.Page
7  {
8  protected int iZufallAktuell;
9  protected string sZufallLetzte;
10
11 protected void Page_Load(object sender, EventArgs e)
12 {
13     iZufallAktuell = (new Random()).Next(1, 1001);
14     sZufallLetzte = Request.Cookies["LetzterZufall"] != null ? Request.Cookies["LetzterZufall"].Value : "-";
15     Response.Cookies.Add(new HttpCookie("LetzterZufall", iZufallAktuell.ToString())
16     {
17         Expires = DateTime.Now.AddHours(1)
18     });
19 }
20 }
21 }

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET WebForms](#) » [Navigation und Masterseiten](#)

## Navigation und Masterseiten

In diesem Thema wollen wir uns mit zwei wichtigen Themen beschäftigen, die vor allem bei der Erstellung von kompletten Websites und **komplexeren Webanwendungen** interessant sind. Dabei geht es hier zum einen um die Seitennavigation, welche nur einmal definiert wird und an mehreren Stellen in verschiedenen Formen eingebunden werden kann, und zum anderen um die sogenannten Masterseiten, wodurch ASP.NET uns eine Art Template-Engine anbietet. Dazu jedoch später mehr.

In den folgenden Beispielen werden wir verschiedene **Navigationsleisten** vorstellen. Hierfür bietet uns ASP.NET bereits einige vorgefertigte Webserver-Steuerelemente an. Diese greifen im Regelfall auf ein Webserver-Dokument zu, welches die **Sitemap** der (kompletten) Website enthält. Bei diesem Dokument, mit dem Namen `Web.sitemap`, handelt es sich um ein XML-Dokument. Als Wurzelement wird `siteMap` verwendet. Dort wird über das Attribut `xmlns`, der Namensraum des Dokuments festgelegt. Als Unterelemente werden `siteMapNode`-Elemente angegeben, welche nach Belieben verschachtelt werden können. Üblicherweise werden im `siteMapNode`-Element die Attribute `url` (Pfad zur Seite), `title` (anzuzeigender Name) und `description` (anzuzeigender Tool-Tip-Text) angegeben. In den folgenden vier Beispielen verwenden wir die folgende Sitemap:



### Inhalt dieser Seite:

1. Klassisches Menü
2. Baumansicht
3. Navigationspfad
4. Masterseiten

```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
3    <siteMapNode url="/" title="ASP.NET Beispiel" description="ASP.NET Beispiel">
4      <siteMapNode url="/Default.aspx" title="Startseite" description="Startseite" />
5      <siteMapNode url="/DirA/" title="Verzeichnis A" description="Verzeichnis A">
6        <siteMapNode url="/DirA/Page1.aspx" title="Seite A.1" description="Seite A.1" />
7        <siteMapNode url="/DirA/Page2.aspx" title="Seite A.2" description="Seite A.2" />
8        <siteMapNode url="/DirA/Page3.aspx" title="Seite A.3" description="Seite A.3" />
9      </siteMapNode>
10     <siteMapNode url="/DirB/" title="Verzeichnis B" description="Verzeichnis B">
11       <siteMapNode url="/DirB/Page1.aspx" title="Seite B.1" description="Seite B.1" />
12       <siteMapNode url="/DirB/Page2.aspx" title="Seite B.2" description="Seite B.2" />
13       <siteMapNode url="/DirB/Page3.aspx" title="Seite B.3" description="Seite B.3" />
14     </siteMapNode>
15   </siteMapNode>
16 </siteMap>

```

Übrigens: Das Kürzel `~/` dient zur Pfadangabe und bezieht sich auf das Root-Verzeichnis der Anwendung / des Projekts.

## Klassisches Menü

Unter einem klassischen Menü versteht man ein Menü, welches **vertikal oder horizontal positioniert** ist und üblicherweise nur **1 Navigationsebene** (auf einmal) anzeigt. Ein solches Menü kann mittels des Webserver-Steuerelements `Menu` realisiert werden. Um das Steuerelement nutzen zu können, müssen wir diesem eine Datenquelle zuweisen. Hierfür benötigen wir noch das Webserver-Steuerelement `SiteMapDataSource`. Die ID, welche diesem Element zugewiesen wird, muss im Steuerelement `Menu` der Eigenschaft `DataSourceID` zugewiesen werden. Für das Element `SiteMapDataSource` gibt es zwei wichtige Eigenschaften: `ShowStartingNode` und `StartingNodeOffset`. `ShowStartingNode` legt fest, ob das Wurzelement bzw. der sogenannte **Startknoten** im Menü angezeigt werden soll oder nicht. Die Eigenschaft `StartingNodeOffset` gibt die **Hierarchieebene** an, ab welcher das Menü „starten“ soll. Im Webserver-Steuerelement `Menu` wird über die Eigenschaft `Orientation` festgelegt, ob das Menü horizontal (`Horizontal`) oder vertikal (`Vertical`) angeordnet ist. Im Beispiel wird zudem `StaticEnableDefaultPopOutImage` auf `false` gesetzt, um das Standard **Erweiterungssymbol** zu deaktivieren. Da die Eigenschaft `StaticPopOutImageUrl`, welche die URL des Erweiterungssymbols festlegt, nicht gesetzt ist, wird kein Symbol angezeigt. Um für die verschiedenen Ebenen unterschiedliche bzw. **individuelle Style** festzulegen, können wir der Eigenschaft `LevelMenuItemStyles` `MenuItemStyle`-Steuerelemente unterordnen. Jedes `MenuItemStyle`-Element repräsentiert dabei eine Ebene / Hierarchie.

```

1  <form runat="server">
2    <asp:SiteMapDataSource ID="SitemapData" ShowStartingNode="false" runat="server" />
3    <asp:Menu DataSourceID="SitemapData" Orientation="Vertical" StaticEnableDefaultPopOutImage="false" runat="server">
4      <LevelMenuItemStyles>
5        <asp:MenuItemStyle BackColor="Blue" ForeColor="White" Width="150" BorderColor="Red" BorderWidth="1"
6        BorderStyle="Solid" HorizontalPadding="10" VerticalPadding="10" />
7        <asp:MenuItemStyle BackColor="LightBlue" ForeColor="White" Width="100" HorizontalPadding="10" VerticalPadding="10"
8      />
9    </LevelMenuItemStyles>
10  </asp:Menu>
11 </form>

```



## Baumansicht

Eine Baumansicht in ASP.NET kann mittels des Steuerelements `asp:TreeView` dargestellt werden. Eine Baumansicht zeigt mehrere Ebenen auf einmal an. Die Zuordnung zur Sitemap wird auch hier über das Element `SiteMapDataSource` und die Eigenschaft `DataSourceID` durchgeführt. Die Eigenschaft `NodeIndent` legt die sogenannte **Einzugsgröße** in Pixeln fest. Das Festlegen von **Anzeige-Stylen** ist auch hier möglich: Für die Baumansicht werden jedoch die Elemente `TreeNodeStyle` und die Eigenschaft `LevelStyles` verwendet.

```

1  <form runat="server">
2    <asp:SiteMapDataSource ID="SitemapData" ShowStartingNode="false" runat="server" />
3    <asp:TreeView DataSourceID="SitemapData" NodeIndent="25" runat="server">
4      <LevelStyles>
5        <asp:TreeNodeStyle BackColor="Blue" ForeColor="White" Width="150" BorderColor="Red" BorderWidth="1"
6        BorderStyle="Solid" HorizontalPadding="10" VerticalPadding="10" />
7      </LevelStyles>
8    </asp:TreeView>
9  </form>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » ASP.NET » ASP.NET WebForms » Navigation und Masterseiten

```

6      <asp:TreeNodeStyle BackColor="LightBlue" ForeColor="White" Width="125" HorizontalPadding="10" VerticalPadding="10"
7  />
8      </LevelStyles>
9  </asp:TreeView>
10 </form>

```



## Navigationspfad

Der Navigationspfad (auch bekannt als Breadcrumb-Navigation oder Brotkrümel-Navigation) ist eine spezielle Navigation, welche den **aktuellen Standpunkt in der Sitemap** anzeigt und als eine Leiste dargestellt wird. Das Webserver-Steuerelement `SiteMapPath`, welches einen Navigationspfad repräsentiert, besitzt unter anderem die Eigenschaften `PathDirection`, `PathSeparator` und `RenderCurrentNodeAsLink`. `PathDirection` legt die **Reihenfolge des Navigationspfads** fest: von links nach rechts (`RootToCurrent`) oder von rechts nach links (`CurrentToRoot`). `PathSeparator` legt das Zeichen bzw. die Zeichenkette fest, welche **zwischen den Pfadelementen** angezeigt werden soll. `RenderCurrentNodeAsLink` erwartet als Parameter einen Wert vom Typ `bool`. Wird hier `true` festgelegt, so wird das **aktuelle Element ebenfalls als Link** angezeigt. Das Element `SiteMapPath` benötigt kein `SiteMapDataSource`-Element. Im Beispiel wird dieses Element lediglich für das Steuerelement `Menu` benötigt.

```

1  <form runat="server">
2      <asp:SiteMapPath RenderCurrentNodeAsLink="true" ID="SitemapPath" runat="server" /><br /><br />
3      <asp:SiteMapDataSource ID="SitemapData" StartingNodeOffset="0" ShowStartingNode="false" runat="server" />
4      <asp:Menu DataSourceID="SitemapData" Orientation="Vertical" StaticEnableDefaultPopOutImage="false" runat="server">
5          <LevelMenuItemStyles>
6              <asp:MenuItemStyle BackColor="Blue" ForeColor="White" Width="150" BorderColor="Red" BorderWidth="1"
7  BorderStyle="Solid" HorizontalPadding="10" VerticalPadding="10" />
8              <asp:MenuItemStyle BackColor="LightBlue" ForeColor="White" Width="100" HorizontalPadding="10"
9  VerticalPadding="10" />
10 </LevelMenuItemStyles>
11 </asp:Menu>
12 </form>

```



## Masterseiten

Mit Masterseiten haben Sie die Möglichkeit, das **HTML-Layout** nur einmal (in der Masterseite) zu definieren, diese Seite auf den sogenannten **Inhaltsseiten** einzubinden und über **Platzhalter** Informationen in die Seite einzufügen. Dieses Feature erlaubt uns, ein **durchgängiges Layout leicht erstellen zu können**. Von Vorteil ist dies vor allem dann, wenn das Layout geändert werden soll: Im Idealfall muss nur die Masterseite (und CSS-Skripte) geändert werden, da die Platzhalter gleich bleiben und somit sich an den Inhaltsseiten nichts direkt ändert. Resultat der Masterseiten ist also zum einen die **einfache Möglichkeit einer Layout-Änderung** und zum anderen natürlich ein **wesentlich kürzerer Code**, da der HTML-Code für Seitenbestandteile wie Kopfzeile, Navigation und Fußzeile nur einmal definiert ist. PHP kennt so eine Funktion, welche der Masterseiten ähnlich ist, nicht, jedoch wird hier oft ein einfacher Trick angewendet: Es gibt die Dateien `header.php` und `footer.php`, welche von allen Inhaltsseiten inkludiert werden, wovon der Inhalt der Seite zwischen den `include`-Befehlen notiert wird. Enthält die Seite **mehrere Platzhalter**, so ist dies nicht mehr so einfach möglich, was hingegen in ASP.NET ziemlich leicht realisierbar ist.

Eine Masterseite ist eine Datei mit der Endung `.Master`. Zudem besitzt eine Masterseite auch eine **Code-Behind-Datei** (Endung `.Master.cs`) sowie eine vom Designer generierte Datei (Endung `.Master.designer.cs`). Im Gegensatz zu den bisherigen Seiten enthält die Masterseite nicht die `Page`-Direktive, sondern die `Master`-Direktive. Auch hier werden für gewöhnlich die Attribute `Language`, `AutoEventWireup`, `CodeBehind` und `Inherits` festgelegt. Der Code der Seite selbst kann, so wie andere Seiten auch, HTML-Elemente, HTML-Server-Steuerelemente und Webserver-Steuerelemente enthalten. Um einen **Platzhalter zu definieren**, notieren wir in der Masterseite das Webserver-Steuerelement `ContentPlaceHolder`. Über das Attribut `ID` muss diesem eine ID zugewiesen werden.

In der Inhaltsseite selbst wird weiterhin die `Page`-Direktive verwendet. Dort werden jedoch noch zusätzlich die Attribute `Title` (Titel der Seite, dieser wird in das `title`-Element eingesetzt) und `MasterPageFile` (Pfad mit Dateinamen zur Masterseite) angegeben. Um einen **Platzhalter mit Inhalt zu füllen**, wird das Webserver-Steuerelement `Content` angegeben. Über das Attribut `ContentPlaceHolderID` wird die Referenz zum Platzhalter festgelegt.

### Masterseite:

```

1  <%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Website.master.cs" Inherits="HwhBsp.Masterseiten.Website" %>
2
3  <!DOCTYPE html>
4  <html>
5      <head runat="server">
6          <title>Masterseite - Masterseiten - ASP.NET Code-Beispiel</title>
7
8          <meta charset="utf-8" />
9
10         <meta name="robots" content="noindex,nofollow" />
11         <meta name="publisher" content="Homepage-Webhilfe" />
12
13         <style type="text/css">
14             body
15             {
16                 width: 800px;
17                 font-family: Arial;
18             }
19
20             #SitemapPath
21             {
22                 display: block;

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » ASP.NET » ASP.NET WebForms » Navigation und Masterseiten

```

23         margin: 5px 5px 10px;
24         padding: 8px;
25         background-color: red;
26         font-size: 13px;
27         color: white;
28     }
29     #SitemapPath a
30     {
31         color: white;
32         text-decoration: none;
33     }
34
35     #PageMenu
36     {
37         float: left;
38         width: 150px;
39         margin: 5px;
40         font-size: 14px;
41     }
42
43     #PageContent
44     {
45         float: right;
46         width: 600px;
47         font-size: 15px;
48     }
49 </style>
50 </head>
51
52 <body>
53     <form runat="server">
54         <asp:SiteMapPath RenderCurrentNodeAsLink="true" ID="SitemapPath" runat="server" />
55         <asp:SiteMapDataSource ID="SitemapData" StartingNodeOffset="0" ShowStartingNode="false" runat="server" />
56         <div id="PageMenu">
57             <asp:Menu ID="SitemapMenu" SkipLinkText="" DataSourceID="SitemapData" Orientation="Vertical"
58             StaticEnableDefaultPopOutImage="false" runat="server">
59                 <LevelMenuItemStyles>
60                     <asp:MenuItemStyle BackColor="Blue" ForeColor="White" Width="150" BorderColor="Red" BorderWidth="1"
61                     BorderStyle="Solid" HorizontalPadding="10" VerticalPadding="10" />
62                     <asp:MenuItemStyle BackColor="LightBlue" ForeColor="White" Width="100" HorizontalPadding="10"
63                     VerticalPadding="10" />
64                 </LevelMenuItemStyles>
65             </asp:Menu>
66         </div>
67         <asp:Panel ID="PageContent" runat="server">
68             <asp:ContentPlaceHolder ID="ContentPlaceholder" runat="server" />
69         </asp:Panel>
70     </form>
71 </body>
72 </html>

```

## Einzelseite (Default.aspx):

```

1  <%@ Page Title="Startseite - Masterseiten - ASP.NET Code-Beispiel" Language="C#" AutoEventWireup="true"
2  MasterPageFile="Website.Master" CodeBehind="Default.aspx.cs" Inherits="HwHbSp.Masterseiten.Default" %>
3  <asp:Content ContentPlaceHolderID="ContentPlaceholder" runat="server">
4      <div style="padding: 10px; text-align: center;">
5          <h1 style="margin: 0px; font-size: 30px;">Startseite</h1>
6          <p>
7              ...
8          </p>
9      </div>
10 </asp:Content>

```

**Wichtig:** Um den Titel der Seite über die Inhaltsseite setzen zu können, muss das `head`-Element in der Masterseite vorhanden sein und zudem das Attribut `runat` mit dem Wert `server` besitzen. Wie Sie vielleicht gesehen haben, wird auf der Inhaltsseite kein HTML-Code (auch kein `form`-Element) außerhalb von den `asp:Content`-Elementen platziert. Dies ist ebenfalls Vorschrift.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

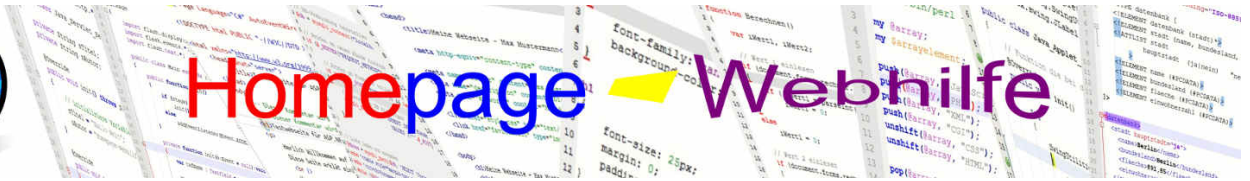
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Grundlagen](#)

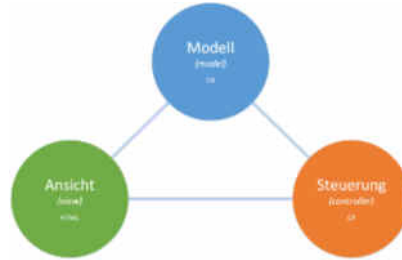
## Grundlagen

ASP.NET MVC ist neben ASP.NET WebForms ein weiterer wichtiger Bestandteil von ASP.NET. Bei ASP.NET MVC wird das **MVC-Konzept** verwendet, um somit die klare Trennung zwischen dem **Modell** (engl. *model*), der **Ansicht** (engl. *view*) und der **Steuerung** (engl. *controller*) zu ermöglichen.

Diese Trennung ermöglicht zum einen eine **Strukturierung der Software** und zum anderen die Möglichkeit zur **Aufteilung der Entwicklung** der Webanwendung. Bei der Aufteilung muss im Voraus das Modell von beiden beteiligten Personen festgelegt werden. Anschließend kann der Webdesigner die Ansicht erstellen und der Programmierer die Steuerung (und somit die Logik, welche hinter der Anwendung steht) programmieren.

Im Gegensatz zu ASP.NET WebForms ist ASP.NET MVC auch Teil des neueren Frameworks ASP.NET Core. Dadurch ist es möglich, MVC-Anwendungen auch auf Systemen mit der Softwareplattform .NET Core auszuführen. Die Plattform .NET Core kommt dabei vor allem auf **anderen Betriebssystemen wie Windows** vor, da unter Windows üblicherweise das .NET Framework zum Einsatz kommt, auf welchem sowohl ASP.NET WebForms als auch ASP.NET MVC ausgeführt werden kann.

Auf dieser Seite wollen wir uns mit einem ersten kurzen Beispiel sowie einigen theoretischen Themen beschäftigen, sodass Sie über eine gute Basis verfügen, bevor Sie sich den nächsten 3 Themen widmen, welche wir in Steuerung, Ansicht und Modell aufgeteilt haben.



### Inhalt dieser Seite:

1. Erstes Beispiel
2. Dateien und Ordner
3. Razor
4. Funktionsweise

## Erstes Beispiel

Um eine ASP.NET MVC Webanwendung zu erstellen, wählen wir beim Erstellen eines neuen Projekts unter dem Eintrag Web „ASP.NET MVC 4-Webanwendung“ aus. Anschließend erscheint ein weiteres Fenster, in welchem eine Vorlage ausgewählt werden kann. Hier wählen wir „Leer“ aus. In der Auswahlliste „Ansichtsmodul“ wählen wir „Razor“. Alternativ könnte hier auch „ASPX“ gewählt werden, wodurch der Web Forms View Engine verwendet wird. Das **Ansichtsmodul** (engl. *view engine*) wird, wie der Name schon vermuten lässt, in der Ansicht für die **Einbettung der dynamischen Inhalte** verwendet. Auf Grund des einfacheren Syntax sollte für MVC-Anwendungen vorzugsweise **Razor** verwendet werden. Die hier genannten Angaben beziehen sich auf Visual Studio 2013. Ab der .NET Framework Version 4.5 gibt es bei der Projekterstellung nur noch den allgemeinen Typ „ASP.NET-Webanwendung“. Wird dieser Typ verwendet, so kann im nächsten Dialog eine Vorlage (in diesem Falle eine leere Vorlage) gewählt werden. Zudem muss dann bei „Ordner und Kernverweise hinzufügen für:“ die Option „MVC“ gewählt werden.

Nachdem wir ein solches leeres Projekt erstellt haben, können wir nun Steuerungen, Ansichten, Modelle sowie andere Dateien hinzufügen. Wenn wir das bisher erstellte (leere) Projekt starten, so bekommen wir eine Fehlermeldung, dass die gewünschte Ressource nicht existiert. Dies ist soweit korrekt, denn wir haben ja bisher noch keine Dateien bzw. „Seiten“ erstellt. Um die eingehenden Anfragen zu bearbeiten, benötigen wir **als erstes eine Steuerung** (Controller). Steuerungen befinden sich im Ordner `Controllers` und können über das Kontextmenü mittels „Hinzufügen“ » „Controller“ hinzugefügt werden. Als Dateiname für den Controller geben wir `HomeController.cs` an. Als Template muss u. U. eine leere Steuerung gewählt werden.

Im erstellten Controller wird automatisch die Funktion `Index()` mit dem Rückgabotyp `ActionResult` erstellt, welche den Rückgabewert der Funktion `View()` zurückgibt. Des Weiteren ist die erstellte Klasse (welche die Steuerung repräsentiert) von der Klasse `Controller` abgeleitet. Für unser erstes Beispiel ändern wir den Rückgabotyp der Funktion `Index()` in `string` und geben einen beliebigen Wert (z. B. die Zeichenkette „Hallo Welt!“) zurück. Dies ist notwendig, da wir bisher noch keine Ansicht erstellt haben und somit ebenfalls eine Fehlermeldung bekommen würden.

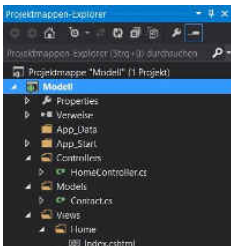
```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6
7  namespace HwHbSp.Erstes_Beiispiel.Controllers
8  {
9      public class HomeController : Controller
10     {
11         public string Index()
12         {
13             return "Hallo Welt!";
14         }
15     }
16 }
    
```



**Wichtig:** Die Steuerung muss unbedingt den Namen `Home` tragen. Dies ist auf Grund der vordefinierten Route (dazu später mehr) notwendig.

## Dateien und Ordner



Bei der Erstellung eines MVC-Projekts bzw. eines leeren Projekts mit den „Ordnern und Kernverweisen“ für ASP.NET MVC werden einige Ordner und Dateien erstellt. Diese automatisch erstellten Ordner und Dateien werden wir nun in den folgenden Abschnitten erläutern.

Der Ordner `App_Data` enthält, wie der Name schon sagt, sogenannte **Anwendungsdaten**. Der Ordner kann MDF-, XML- und andere sogenannte Datenspeicherungs-Dateien enthalten. Für uns ist dieser Ordner jedoch nicht von weiterer Bedeutung.

Im Ordner `App_Start` befinden sich ein paar C#-Dateien, welche jeweils über eine statische Funktion verfügen. Diese Funktionen werden von der Funktion `Application_Start()` aus der Datei `Global.asax.cs` aufgerufen. Alle Dateien werden zur Konfiguration der Webanwendung verwendet. Die Dateien im Ordner `App_Start` werden von Visual Studio automatisch erstellt und besitzen bereits die Standard-Konfiguration. Für einige Fälle ist es jedoch notwendig, diese zu ändern. Die Datei `FilterConfig.cs` enthält eine Funktion zum Registrieren von sogenannten **Aktions-Filtern** (dazu [später mehr](#)). In der Datei `RouteConfig.cs` werden

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

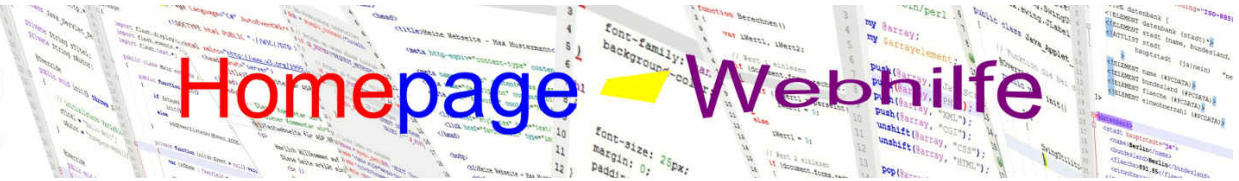
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Grundlagen](#)



die **Routes** für die Anwendung angelegt. Standardmäßig ist hier eine Route mit dem Namen „Default“ angelegt. Mit dem Thema Routing werden wir uns jedoch **später auch noch beschäftigen**. Zuletzt ist noch die Datei `WebApiConfig.cs` zu erwähnen. Dort werden **Routes für die ASP.NET Web API Technologie registriert**. ASP.NET Web API ist mit ASP.NET MVC vergleichbar und kann zudem auch miteinander „kombiniert“ werden. Bei ASP.NET Web API werden vor allem clientseitige Skripte zum Laden oder Ändern

von Daten (mittels AJAX) verwendet. Dieses Thema werden wir hier jedoch nicht weiter behandeln und uns voll und ganz auf ASP.NET MVC konzentrieren. Der Ordner `Controllers` enthält die verschiedenen **Steuerungs-Klassen**. Steuerungen enthalten am Ende immer den Zusatz `Controller` und die Dateiendung `.cs` (oder natürlich `.vb` für Visual Basic). Beim Erstellen eines leeren Projekts ist dieser Ordner leer.

Im Ordner `Models` werden die **Datenmodelle** abgelegt, wovon es sich bei einem Modell um normale Klassen handelt, die zumeist über einige Eigenschaften verfügen. Auch dieser Ordner ist bei der Projekterstellung eines leeren Projekts leer.

Der letzte Ordner, welcher neben den Ordnern `Controllers` und `Models` auch direkt zum MVC-Konzept gehört, ist `Views`. Dort werden die **Ansichten** abgelegt. Ansichten verfügen über die Dateiendung `.cshtml` (bei Verwendung des View Engine Razor) oder `.aspx` (bei Verwendung des View Engine ASPX). Dieser Ordner enthält standardmäßig lediglich die Datei `Web.config`.

Direkt im Projektverzeichnis befinden sich neben den Konfigurationsdateien `Web.config`, `Web.Debug.config`, `Web.Release.config` und `packages.config` noch die Dateien `Global.asax` und `Global.asax.cs`. Die letzten zwei genannten Dateien werden für den **Lebenszyklus der Anwendung** (nicht zu verwechseln mit dem Lebenszyklus einer Anfrage) benötigt. Wie bereits oben erwähnt, werden durch die Methode `Application_Start()`, welche beim Start der Anwendung aufgerufen wird, die verschiedenen Einstellungen (aus den Dateien im `App_Start`-Ordner) angewendet.

Im Gegensatz zu ASP.NET WebForms unterscheiden sich bei ASP.NET MVC die **URLs** (welche vom Benutzer aufgerufen werden) mit den tatsächlichen / **physikalischen Pfaden**. Wurde bei WebForms die URL `VerzeichnisA/SeiteB.aspx` aufgerufen, so wurde auch die Datei `SeiteB.aspx` aus dem Ordner `VerzeichnisA` geladen. Bei MVC ist dies anders: Hier teilt sich die URL (im Normalfall und somit nur bei Beachtung der Standard-Route) in drei Teile. Das Format lautet wie folgt: `{controller}/{action}/{id}`. Rufen wir also bspw. die URL `Home/Index/1` auf, so wird die Aktion `Index()` des Controllers `Home` (Datei `Controllers/HomeController.cs`) ausgeführt. Dieser Aktion wird zudem als ID die `1` übergeben. Gibt die Aktion `Index()` den Rückgabewert der Funktion `View()` zurück, um somit die Ansicht der Aktion anzuzeigen, so wird die Ansicht aus der Datei `Views/Home/Index.cshtml` (physikalischer Pfad) geladen. Mittels Routing sind jedoch auch andere Zusammensetzungen und Aufbauten der URLs möglich.

### Razor

Razor ist eine **serverseitige Auszeichnungssprache**, welche bei MVC dazu genutzt wird, Code-Blöcke in die Ansicht einzubetten. Die Anzahl, aber vor allem die Komplexität dieser Code-Blöcke, sollte auf Grund des MVC-Konzepts und somit auf Grund der Trennung zwischen Ansicht und Steuerung relativ **gering gehalten** werden.

Razor verfügt über ein paar unterschiedliche **Syntax-Regeln**, welche wir in den folgenden Abschnitten vorstellen werden. Bei allen wird jedoch das `@`-Zeichen vorangestellt. Razor ermöglicht auf einfache Art und Weise das Einbinden von Code-Blöcken direkt in den HTML-Code, um somit den Code „zu mischen“. Ein solches Beispiel werden wir gleich noch bei Bedingungen und Schleifen genauer anschauen.

Wollen wir einen Wert einer Variablen ausgeben, so notieren wir einfach das `@`-Zeichen gefolgt von dem Variablennamen: z. B. `@variablenName`. An Stelle eine Variable auszugeben, kann natürlich auch der Rückgabewert einer Funktion ausgegeben werden. Dieser Syntax ruft indirekt die Funktion `Response.Write()` auf.

Benötigen wir innerhalb unserer Ansicht einen Code-Block, welcher aus mehreren Anweisungen besteht oder keinen Wert zurückgibt, so gilt der Syntax `@{ }`. Innerhalb der geschweiften Klammern kann nun C#, aber auch HTML-Code, notiert werden.

```
1  @{:
2      string sUserAgent = Request.UserAgent != null ? Request.UserAgent : "-";
3      <b>Ihr User-Agent lautet:</b> @sUserAgent
4  }
```

Eine **Abfrage** kann in Razor direkt (ohne Code-Block) notiert werden, wovon der Verzweigungs-Block jedoch automatisch wieder einen Code-Block mit sich bringt. Bei der Abfrage kann es sich um eine einfache Verzweigung (`if-else`) oder eine mehrfache Verzweigung (`switch-case`) handeln. Auch hier ist das direkte Mischen von C# und HTML-Code möglich.

```
1  @if (Request.UserAgent == null || Request.UserAgent == "")
2  {
3      <b>Sie haben keinen User-Agent.</b>
4  }
5  else
6  {
7      <b>Ihr User-Agent lautet:</b> @Request.UserAgent
8  }
```

Das gleiche Prinzip gilt bei **Schleifen**, wie das folgende Beispiel zeigt:

```
1  <ul>
2      @for (int i = 0; i < 10; i++)
3      {
4          <li>@i</li>
5      }
6  </ul>
```

**Wichtig:** Wie Sie erkennen können, muss die Person, welche die Ansichten erstellt (dies gilt natürlich nur, sofern Sie die Entwicklung der Webanwendung aufgeteilt haben), ebenfalls über einfache C#-Kenntnisse verfügen und zudem mit dem Razor-Syntax vertraut sein.

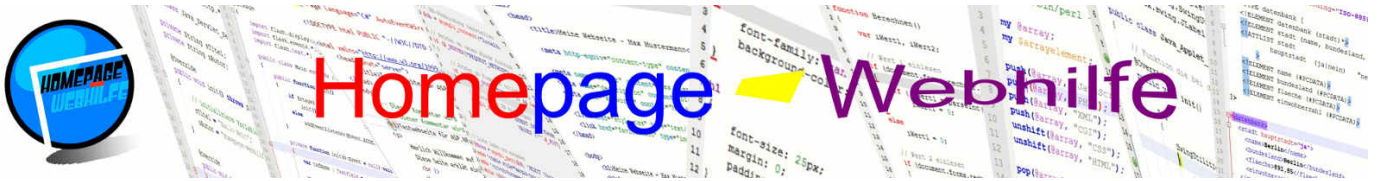
**Wichtig:** Versuchen Sie große und komplexe Code-Blöcke weitestgehend zu vermeiden und diese in den Controller zu verlagern. Es ist jedoch durchaus legitim in der Ansicht z. B. eine `for`-Schleife mittels Razor zu nutzen, um die Datensätze, welche im Datenmodell abgelegt sind, auszugeben.

### Funktionsweise

In ASP.NET MVC muss zu allererst zwischen dem Anwendungs-Lebenszyklus und dem Seiten-Lebenszyklus unterschieden werden.

Der **Anwendungs-Lebenszyklus** startet mit dem Start des Webserver (meistens der IIS von Microsoft) und endet mit dem Stopp des Webserver. Standardmäßig

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Grundlagen](#)

werden beim **Applikationsstart** lediglich die verschiedenen Einstellungen für die Anwendung gesetzt. Die Dateien `Global.asax` und `Global.asax.cs` repräsentieren dabei die MVC-Anwendung. Die in den Dateien enthaltene Klasse `MvcApplication` wird von der Klasse `HttpApplication` (Namensraum `System.Web`) abgeleitet. Beim Erstellen eines Projekts enthält diese Klasse lediglich die Funktion `Application_Start()`, welche beim Start der MVC-Applikation (also beim Start des Webservers) aufgerufen wird, und die verschiedenen Konfigurationen (Filter, Routen und Web-API-Routen), welche in den Dateien im Ordner `App_Start` angegeben sind, durchführt. Änderungen an der Datei `Global.asax` (und deren Code-Behind-Datei `Global.asax.cs`) wirkt sich direkt auf die sogenannte **Anwendungsebene** aus. In diesem Tutorial werden wir jedoch hier keine Änderungen durchführen (abgesehen von den Routen in der Datei `App_Start/RouteConfig.cs`).

Während eines Anwendungs-Lebenszyklus kann es mehrere **Anfrage-Lebenszyklen** geben. Der Anfrage-Lebenszyklus startet mit einer ganz normalen **HTTP-Anfrage** (engl. *request*). Wurde diese vom Server empfangen, so wird die Anfrage über die angefragte URL mittels einem **Routing-Verfahren** der Applikation und anschließend der Steuerung (Controller) zugeordnet. Nun wird ein Objekt der Steuerungs-Klasse instanziiert und anschließend die jeweilige Aktions-Methode ausgeführt. Wird von der Aktions-Methode eine Ansicht zurückgegeben, so wird die Ansicht von dem sogenannten **View Engine** gerendert. Am Ende wird aus den Daten (Header und Seiteninhalt) eine Antwort (engl. *response*) erstellt, die dann an den Client (üblicherweise an den Webbrowser des Besuchers) gesendet wird.

MVC-Klassen (z. B. `ActionResult`, `HtmlHelper`, `ViewPage` und `WebViewPage`) befinden sich im Namensraum `System.Web.Mvc` und dessen Sub-Namensräume (z. B. `System.Web.Mvc.Html` und `System.Web.Mvc.Routing`). Auf einige der Klassen gehen wir im Laufe dieses Tutorials noch genauer ein.

Für eine „funktionsfähige“ Anwendung ist **mindestens ein Controller notwendig**. Der Controller enthält dabei üblicherweise eine oder mehrere Aktionen. Bei den **Aktionen** handelt es sich um Funktionen, welche im Regelfall ein Objekt einer von `ActionResult` abgeleiteten Klasse zurückgeben. Über den Funktionsaufruf `View()` wird die Ansicht geladen. Jedoch ist es nicht zwingend erforderlich, dass eine Aktion eine Ansicht zurückgibt. Wir werden in den weiteren Themen noch genauer auf die einzelnen Bestandteile des MVC-Konzepts eingehen.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

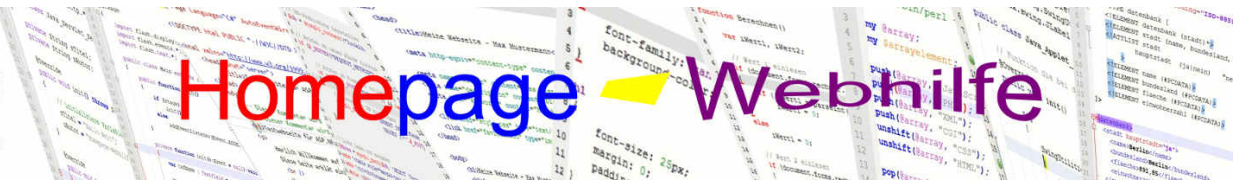
Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung





Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Steuerung](#)

## Steuerung



Steuerungen (oder auch Controller genannt) sind der **Grundbaustein für MVC-Anwendungen** und stehen nach dem Erhalt einer **HTTP-Anfrage** an erster Stelle. Der für die URL zugehörige Controller sowie die für die URL zugehörige Anwendung wird mittels eines **Routing-Verfahrens** vom MVC-Framework ermittelt, welches wir im nächsten Abschnitt genauer erklären werden. Die Steuerung repräsentiert die sogenannte **Anwendungslogik**. Jede öffentliche Funktion (Zugriffsmodifizierer `public`) stellt eine sogenannte **Aktionsmethode** dar. Eine Aktionsmethode kann in der Regel von außen indirekt über die angegebene URL aufgerufen werden. Als Rückgabebetyp verfügen Aktionsmethoden üblicherweise über die Klasse `ActionResult` (bzw. über eine von der Klasse `ActionResult` abgeleitete Klasse). Aktionsmethoden können dabei Übergebeparameter besitzen. Dies können zum einen individuelle Parameter (festgelegt durch Routen) und zum anderen ein Datenmodell sein (bei POST-Aktionen). Darauf werden wir jedoch später noch genauer eingehen.

Prinzipiell sind Steuerungen nichts anderes als C#-Klassen. Jedoch sind diese von der Klasse `Controller` (Namensraum `System.Web.Mvc`) abgeleitet und müssen am Ende des Klassennamens über den Suffix `Controller` verfügen. Der andere Teil des Klassennamens stellt den eigentlichen **Controller-Namen** dar. Der Name der Aktionsmethode entspricht, sofern dieser nicht über den `ActionName`-Selektor (dazu später mehr) geändert wurde, auch dem Namen, welcher in der URL zum Aufruf der Aktion angegeben wird.

**Inhalt dieser Seite:**

1. Routing
2. Aktionen
3. Filter
4. Selektoren

## Routing

Wie bereits erklärt, werden die URLs mittels eines Routing-Verfahrens zu allererst **einer Anwendung, anschließend einer Steuerung und letztendlich einer Aktionsmethode zugeordnet**. Dabei gilt für die URL innerhalb einer MVC-Anwendung folgender Aufbau: `{controller}/{action}/{id}`. Ein Aufruf von `Home/Index/1` wird also dem Controller `Home` und der Aktionsmethode `Index()` zugewiesen. Dieser kann (sofern gewünscht) ein Parameter übergeben werden, welcher im obigen Beispiel dem Wert `1` entsprechen würde. Ein Aufruf von `Kunden/Editierung/4781` würde die Aktionsmethode `Editierung()` des Controllers `Kunden` (Datei `KundenController.cs`) aufrufen und der Methode den Wert `4781` übergeben. Dies ist wohl ein typisches Beispiel für ASP.NET MVC, wie die URL für die Editierung eines Kunden in einem Firmen-Portal aussehen könnte.

Möchten wir diesen Standardaufbau ändern oder erweitern, so müssen wir **weitere Routen registrieren oder vorhandene anpassen**. Eine Route ist beim Erstellen eines MVC-Projekts bereits vorhanden. Diese, welche wir oben bereits erklärt haben, wird auch als **Standard-Route** (engl. *default route*) bezeichnet. Routen werden auf Anwendungsebene definiert und sind standardmäßig in der Funktion `RegisterRoutes()` der Datei `App_Start/RouteConfig.cs` definiert. Der Funktion wird als Parameter ein Objekt der Klasse `RouteCollection` übergeben. In diesem werden die Routen registriert.

Innerhalb der Funktion `RegisterRoutes()` wird nun die Funktion `MapRoute()` aufgerufen, mit welcher URL-Routen festgelegt werden können. Als Parameter werden **der Routenname, das URL-Muster und die Standard-Werte** übergeben. Der Routenname kann zur Identifizierung der Route und für Routenlinks verwendet werden. Der **Routenname** wird in der Routing-Tabelle als Index verwendet, weshalb der Name eindeutig sein muss. Das URL-Muster für Routen besteht aus **Platzhaltern**, Schrägstrichen (zur Trennung) sowie bei Bedarf Konstanten. Zwischen Platzhaltern ist es zwingend erforderlich, dass ein Schrägstrich vorhanden ist. Platzhalter werden in geschweifte Klammern notiert. Die einfachsten Beispiele sind die in der Standard-Route enthaltenen Platzhalter `{controller}` und `{action}`. `{controller}` verweist dabei direkt auf den Namen der Steuerung und `{action}` auf den Namen der Aktionsmethode. Weitere Platzhalter können nach Belieben definiert werden. Der Wert solcher Platzhalter werden ebenfalls über die eingegebene URL spezifiziert und der Aktionsmethode als Parameter übergeben. Die Zuordnung des Platzhalterwerts zu dem Parameter wird über den Namen getroffen, d. h. der Name des Platzhalters muss auch dem Parameternamen der Aktionsmethode entsprechen. Über den dritten Parameter der Funktion `MapRoute()` werden die **Standardwerte** der Platzhalter festgelegt. Dafür wird in der Regel ein anonymes Objekt erzeugt und dieses der Funktion übergeben. Das (anonyme) Objekt enthält Eigenschaften, die die Standardwerte der Platzhalter repräsentieren. Hierfür muss der Name der Eigenschaft dem Namen des Platzhalters entsprechen. Die Standardwerte (engl. *default values*) werden immer dann benötigt, wenn einer der Werte nicht angegeben wurde. Dies ist auch der Grund, warum beim Aufruf des Root-Verzeichnisses der Anwendung standardmäßig die Steuerung `Home` und die Aktionsmethode `Index` aufgerufen wird. Existiert z. B. ein weiterer Controller mit dem Namen `Produkt`, so wird beim Aufruf von `~/Produkt` die Aktionsmethode `Index()` des `Produkt`-Controllers aufgerufen. Möchten Sie einen Platzhalter nicht mit einem Standard-Wert belegen, sondern diesen als **optional** markieren, so können Sie der Eigenschaft des Objekts für Standardwerte die Konstante `UrlParameter.Optional` zuweisen. Wird der Platzhalter in der URL nicht angegeben, so wird der Aktionsmethode als Wert `null` übergeben. Hierbei ist darauf zu achten, dass der Datentyp den Wert `null` unterstützt. Die Funktion `IgnoreRoute()` erlaubt es mit Hilfe eines URL-Musters, bestimmte Routen zu ignorieren, um somit z. B. einen Controller „zu deaktivieren“.

Im unteren Beispiel gibt es zwei Controller: `Home` und `Info`. Beide Controller verfügen über mehrere Aktionsmethoden. Die Aktionsmethode `MeineID()` des Controllers `Home` verwendet den Platzhalter bzw. Parameter `id` und gibt dessen Wert aus. In der Routenkonfiguration wurde eine Route hinzugefügt, mit welcher die Aktionsmethode auch über die URL `~/ID` aufgerufen werden kann. Die folgende Tabelle zeigt die verschiedenen Aufrufmöglichkeiten für die Webanwendung und die dazugehörigen Controller sowie Aktionsmethoden:

URL	Controller	Aktionsmethode	id-Parameter
~/	Home	Index()	null
~/Home	Home	Index()	null
~/Home/Index	Home	Index()	null
~/Home/DatumUhrzeit	Home	DatumUhrzeit()	null
~/Home/MeineID	Home	MeineID()	null
~/Home/MeineID/123	Home	MeineID()	123
~/Info/Anfrage	Info	Anfrage()	null
~/Info/Antwort	Info	Antwort()	null
~/ID	Home	MeineID()	null
~/ID/123	Home	MeineID()	123

**Übrigens:** Ein Aufruf von `~/Info` ist ungültig (bzw. führt zu einem 404 HTTP-Status-Fehler), da der `Info`-Controller über keine `Index()`-Aktionsmethode verfügt.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Steuerung](#)

Hier der Quellcode der zwei Controller sowie der Routenkonfiguration:

## HomeController.cs:

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.Mvc;
6
7 namespace HWhBsp.Routing.Controllers
8 {
9     public class HomeController : Controller
10    {
11        public string Index()
12        {
13            return "Hallo Welt!";
14        }
15
16        public string DatumUhrzeit()
17        {
18            return "Aktuelles Datum und Uhrzeit: " + DateTime.Now.ToString();
19        }
20
21        public string MeineID(string id)
22        {
23            return "ID: " + ((id == null) ? "-" : id);
24        }
25    }
26 }

```

## InfoController.cs:

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.Mvc;
6
7 namespace HWhBsp.Routing.Controllers
8 {
9     public class InfoController : Controller
10    {
11        public string Anfrage()
12        {
13            return "User-Agent: " + Request.UserAgent;
14        }
15
16        public string Antwort()
17        {
18            return "Zeichenkodierung: " + Response.Charset;
19        }
20    }
21 }

```

## RouteConfig.cs:

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.Mvc;
6 using System.Web.Routing;
7
8 namespace HWhBsp.Routing
9 {
10    public class RouteConfig
11    {
12        public static void RegisterRoutes(RouteCollection routes)
13        {
14            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
15
16            routes.MapRoute(
17                name: "ID-Ausgabe",
18                url: "ID/{id}",
19                defaults: new { controller = "Home", action = "MeineID", id = UrlParameter.Optional }
20            );
21            routes.MapRoute(
22                name: "Default",
23                url: "{controller}/{action}/{id}",
24                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Steuerung](#)

```
25 |     );
26 | }
27 | }
28 | }
```



**Wichtig:** Die Routen werden von oben nach unten abgearbeitet. Daher muss die Standard-Route immer als letztes notiert werden.

**Übrigens:** Bei der Angabe von Parametern kann zusätzlich zum Wert auch noch der Name angegeben werden (`name= wert`). Dies wird gemacht, sodass der Programmierer direkt erkennen kann, welcher Wert zu welchem Parameter gehört (ohne die Parameter-Reihenfolge zu kennen). Diese „Beschriftung“ kann jedoch ohne weiteres weggelassen werden. Es gibt zudem einige Funktionen, welche „Benannte Argumente“ verwenden. Bei diesen muss der Parametername angegeben werden, da darüber die Zuordnung stattfindet. Für Sie mag diese Art der Parameterangabe evtl. neu sein, jedoch werden Sie sehen, dass diese beiden Verfahren oft bei ASP.NET MVC verwendet werden.

Stellen wir uns ein praxisnahes Szenario vor: Sie möchten auf Grund von **Suchmaschinenoptimierung** der URL zusätzlich zum Namen des Controllers und der Aktionsmethode ein **Sprachkürzel** hinzufügen. So soll es z. B. an Stelle der URL `~/Produkte/Liste` die URLs `~/DE/Produkte/Liste` und `~/EN/Produkte/Liste` geben. Das URL-Muster könnte in einem solchen Fall z. B. so aussehen: `{lang}/{controller}/{action}/{id}`. Der Parameter `lang` wird dabei den Aktionsmethoden übergeben, wodurch dieser den Inhalt der jeweiligen Seite laden kann. Hier zwei Beispielcodeausschnitte:

```
1 | public string Index(string lang)
2 | {
3 |     return "Ihre Sprache ist: " + lang;
4 | }
5 |
6 | routes.MapRoute(
7 |     name: "Default",
8 |     url: "{lang}/{controller}/{action}/{id}",
9 |     defaults: new { lang = "DE", controller = "Home", action = "Index", id = UrlParameter.Optional }
10 | );
```

**Wichtig:** Parameter am Anfang oder in der Mitte einer URL müssen angegeben werden. Im obigen Beispiel kann der Parameter `lang` also nicht **weggelassen** werden. Grundsätzlich können Parameter also immer nur in **umgekehrter Reihenfolge** und somit vom Ende zum Anfang weggelassen werden.

## Aktionen

Aktionsmethoden können **unterschiedliche Rückgabetypen** haben. Bisher haben wir der Einfachheit halber den Rückgabetypp `string` verwendet. Im Regelfall wird jedoch der Rückgabetypp `ActionResult` verwendet. Die Klasse `ActionResult` ist eine abstrakte Klasse, d. h. von dieser kann kein Objekt erzeugt werden. Jedoch gibt es einige Klassen, welche von der Klasse `ActionResult` abgeleitet sind. Trotzdem kann als Rückgabetypp stets `ActionResult` angegeben werden. Dies ermöglicht z. B. das Zurückgeben von einem Objekt der Klasse `RedirectResult` oder `ContentResult` innerhalb der gleichen Aktionsmethode. Um ein Objekt solcher Klassen zu erzeugen, können oft **Hilfsfunktionen** eingesetzt werden. Die folgende Tabelle zeigt eine Auflistung der wichtigsten Klassen, welche von `ActionResult` abgeleitet sind, sowie der Hilfsfunktionen und einer Beschreibung:

Klasse	Hilfsfunktion	Beschreibung
<code>ContentResult</code>	<code>Content()</code>	Gibt einen Inhalt (ggf. mit einem angegebenen Inhaltstyp) zurück.
<code>FileResult</code>	<code>File()</code>	Gibt den Inhalt der angegebenen Datei (ggf. mit dem angegebenen Inhaltstyp) zurück.
<code>HttpNotFoundResult</code>	-	Gibt einen 404 HTTP-Status-Fehler bzw. eine Umleitung zur dazugehörigen Fehlerseite zurück.
<code>HttpStatusCodeResult</code>	-	Gibt einen beliebigen HTTP-Status-Fehler bzw. eine Umleitung zur dazugehörigen Fehlerseite zurück.
<code>HttpUnauthorizedResult</code>	-	Gibt einen 403 HTTP-Status-Fehler bzw. eine Umleitung zur dazugehörigen Fehlerseite zurück.
<code>RedirectResult</code>	<code>Redirect()</code>	Gibt eine Umleitung an Hand einer URL zurück.
<code>RedirectToRouteResult</code>	<code>RedirectToAction()</code>	Gibt eine Umleitung an Hand eines Controllers und einer Aktion zurück.
	<code>RedirectToRoute()</code>	Gibt eine Umleitung an Hand einer Route (Angabe aller Platzhalter möglich) zurück.
<code>ViewResult</code>	<code>View()</code>	Gibt eine Ansicht zurück (dazu später mehr).

Im Folgenden sehen Sie ein Beispiel mit mehreren Aktionsmethoden. Zur klaren Darstellung wurde als Rückgabetypp explizit die jeweils verwendete Klasse angegeben und nicht, wie es üblich ist, die Klasse `ActionResult`.

```
1 | using System;
2 | using System.Collections.Generic;
3 | using System.Linq;
4 | using System.Web;
5 | using System.Web.Mvc;
6 |
7 | namespace HwhBsp.Aktionen.Controllers
8 | {
9 |     public class HomeController : Controller
10 |     {
11 |         public string Index()
12 |         {
13 |             return "Hallo Welt!";
14 |         }
15 |
16 |         public RedirectResult UmleitungStartseite()
17 |         {
18 |             return Redirect("~/");
19 |         }
20 |     }
21 | }
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Steuerung](#)

```

20
21     public RedirectToRouteResult UmleitungRoute()
22     {
23         return RedirectToAction("InhaltHTML");
24     }
25
26     public ContentResult InhaltHTML()
27     {
28         return Content("<b>Hallo</b> Welt!", "text/html");
29     }
30
31     public ContentResult InhaltXML()
32     {
33         return Content("<a>\r\n <b>123</b>\r\n <b>456</b>\r\n</a>", "text/xml");
34     }
35
36     public HttpStatusCodeResult ServerFehler()
37     {
38         return new HttpStatusCodeResult(500);
39     }
40
41     public HttpUnauthorizedResult AuthentifizierungsFehler()
42     {
43         return new HttpUnauthorizedResult();
44     }
45
46     public HttpNotFoundResult NichtGefundenFehler()
47     {
48         return new HttpNotFoundResult();
49     }
50
51     public FileResult Datei()
52     {
53         return File("~/Controllers/HomeController.cs", "text/plain");
54     }
55 }
56 }

```



## Filter

Filter erlauben das Anwenden einer bestimmten Logik auf eine Aktion. Diese **Logik** wird vor (lat. *pre*) oder nach (lat. *post*) der Ausführung der Aktionsmethode ausgeführt. Bei Filtern handelt es sich um Attribut-Klassen, d. h. der Attribut-Name der Klasse wird innerhalb von eckigen Klammern oberhalb der Funktion oder der Klasse angegeben. Da das Thema Filter und vor allem das Erstellen von benutzerdefinierten Filtern komplexer ist und es eher für einfachere Anwendungen nicht benötigt wird, werden wir uns hier lediglich mit dem Filter `OutputCache` beschäftigen. Das Attribut `OutputCache` ermöglicht das Festlegen des **Cache-Verhaltens** für die jeweilige Aktion. Mit der Eigenschaft `Duration` können Sie bestimmen, wie lange (angegeben in Sekunden) die Aktion gecacht werden darf. Im unteren Beispiel wird die Aktion `Index()` für 5 Sekunden gecacht. Dieses Verhalten ist im Beispiel auf Grund der Datum- und Uhrzeitausgabe gut ersichtlich.

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.Mvc;
6
7 namespace HWhBsp.Filter.Controllers
8 {
9     public class HomeController : Controller
10    {
11        [OutputCache(Duration=5)]
12        public string Index()
13        {
14            return DateTime.Now.ToString();
15        }
16    }
17 }

```



## Selektoren

Selektoren dienen dazu, Aktionsmethoden anzupassen, um somit das **Verhalten bei Anfragen** zu steuern. Selektoren unterstützen dabei also das Routing-Verfahren. In ASP.NET MVC gibt es drei Selektoren: `ActionName`, `NonAction` und `ActionVerbs`. `ActionName` und `NonAction` werden direkt als Attribute oberhalb der Funktion notiert. Mit `ActionName` können Sie den **Namen der Aktionsmethode ändern**. Der Aktionsname wird im Konstruktor übergeben. Durch dieses Attribut ändert sich jedoch nicht der Name der Funktion, sondern lediglich der Zugriffsname, welcher in der URL zum Aufruf der Aktion verwendet wird. Haben wir also eine Aktionsmethode mit dem Namen `GetDatumUndUhrzeit()`, welcher wir nun den Namen `DatumUhrzeit` zuweisen, so heißt die Funktion weiterhin `GetDatumUndUhrzeit()`. Ein Zugriff mittels einer URL kann jedoch nur noch mit dem Aktionsnamen `DatumUhrzeit` erfolgen. `GetDatumUndUhrzeit` kann

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Steuerung](#)

dann nicht mehr als Zugriffsname in der URL verwendet werden. Mit dem Attribut `NonAction` können Sie eine Funktion markieren, dass diese **nicht als Aktion über die URL aufgerufen werden kann**. Dieses Attribut benötigen Sie immer dann, wenn eine öffentliche Funktion nicht für den HTTP-Zugriff bestimmt ist. ActionVerbs sind spezielle Selektoren, welche Aktionen **auf bestimmte HTTP-Methoden begrenzen**. Die jeweilig anzugebenden Attribute setzen sich aus `Http` und dem Namen der HTTP-Methode (in Camel-Case-Schreibweise) zusammen (z. B. `HttpGet` und `HttpPost`). Hier nun ein Beispiel zum Thema Selektoren:

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6
7  namespace HwhBsp.Selektoren.Controllers
8  {
9      public class HomeController : Controller
10     {
11         [HttpGet]
12         public string Index()
13         {
14             return "<form action=\"" + Url.Content("~/Home/FormularVerarbeitung/") + "\" method=\"post\"><input
15             type=\"submit\" value=\"Formular per POST senden\" /></form>";
16         }
17
18         [HttpPost]
19         public string FormularVerarbeitung()
20         {
21             return "<a href=\"" + Url.Content("~/") + "\">Seite per GET anzeigen</a>";
22         }
23
24         [ActionName("DatumUhrzeit")]
25         public string GetDatumUndUhrzeit()
26         {
27             return GetDatum() + " " + GetUhrzeit();
28         }
29
30         [NonAction]
31         public string GetDatum()
32         {
33             return DateTime.Now.ToShortDateString();
34         }
35
36         [NonAction]
37         public string GetUhrzeit()
38         {
39             return DateTime.Now.ToShortTimeString();
40         }
41     }

```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

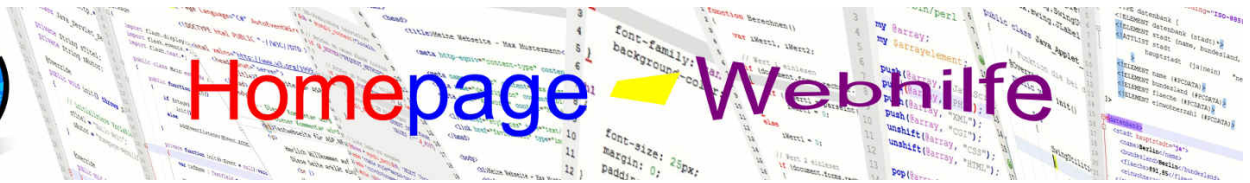
## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Ansicht](#)

## Ansicht

Nachdem wir uns im vorherigen Thema ausführlich mit der Steuerung beschäftigt haben, wollen wir uns nun der Ansicht (engl. *view*), also dem, was unsere Besucher nachher sehen, widmen. Eine Ansicht ist grundsätzlich mal nichts anderes als eine HTML-Datei. Die Datei kann HTML-Code sowie CSS- und JavaScript-Code enthalten. Um eine Verbindung zwischen der Anwendungslogik (dem Controller) und der Ansicht herzustellen, wird das Datenmodell verwendet. Mit dem Datenmodell wollen wir uns jedoch erst im nächsten Thema beschäftigen. Die dynamischen Inhalte werden mit Code-Blöcken eingebettet. Als Programmiersprache kommt C# (oder Visual Basic) zum Einsatz. Der Syntax unterscheidet sich dabei je nach gewähltem Ansichtsmodul. Alle Ansichten befinden sich im Ordner `Views`. Dabei besitzen die Ansichten standardmäßig den gleichen Namen wie die Aktion, welche die Ansicht verwendet, und befinden sich in einem Unterordner, welcher den gleichen Namen wie der jeweilige Controller trägt. Für Ansichten gibt es zwei unterschiedliche **Ansichtsmodule** (engl. *view engine*): ASPX und Razor. **ASPX** kennen Sie vom Syntax ja bereits vom [ASP.NET WebForms Tutorial](#). Den Syntax von **Razor** haben wir im [Thema Grundlagen](#) erläutert und werden wir auch für alle weiteren Beispiele verwenden. Dateien mit dem Ansichtsmodul Razor besitzen die Dateiendung `.cshtml`.



### Inhalt dieser Seite:

1. Ausgaben
2. Layouts

Im folgenden Beispiel haben wir eine Steuerung (Home) mit der Aktionsmethode `Index()`, welche mittels der Funktion `View()` eine Ansicht zurückgibt. Die Ansicht, welche aus dem Verzeichnis `Views/Home/Index.cshtml` geladen wird, enthält ein HTML-Grundgerüst sowie die Ausgabe des aktuellen Datum und der aktuellen Uhrzeit (Zeile 16). Der erste Razor-Code-Block dient zum **Setzen von ein paar Eigenschaften**, welche die Seite betreffen. Standardmäßig, also beim Erstellen einer neuen Ansicht in Visual Studio, wird hier die Eigenschaft `Layout` auf `null` gesetzt, welche zum Festlegen der Layout-Seite verwendet wird, welche in diesem Beispiel nicht vorhanden ist.

### Steuerung (HomeController.cs):

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.Mvc;
6
7 namespace HwHsp.Ansicht.Controllers
8 {
9     public class HomeController : Controller
10    {
11        public ActionResult Index()
12        {
13            return View();
14        }
15    }
16 }

```

### Ansicht (Index.cshtml):

```

1 @{
2     Layout = null;
3 }
4 <!DOCTYPE html>
5 <html>
6 <head>
7     <title>Ansicht - ASP.NET Code-Beispiel</title>
8
9     <meta charset="utf-8" />
10
11     <meta name="robots" content="noindex,nofollow" />
12     <meta name="publisher" content="Homepage-Webhilfe" />
13 </head>
14
15 <body>
16     <b>Datum und Uhrzeit:</b> @DateTime.Now.ToString()
17 </body>
18 </html>

```



## Ausgaben

Ansichten sind von der Klasse `ViewPage` (beim ASPX View Engine) oder `WebViewPage` (beim Razor View Engine) abgeleitet. Neben den Eigenschaften `Request`, `Response` und `Session`, welche Sie bereits von WebForms kennen, besitzen beide Klassen die Eigenschaft `Html`, welche eine Instanz der `HtmlHelper`-Klasse enthält. Die `HtmlHelper`-Klasse enthält einige Funktionen, welche uns, wie der Name schon sagt, helfen, **HTML-Code zu erzeugen**.

Mit der Methode `ActionLink()` können wir einen **Link** erzeugen, welcher auf einen Controller und eine Aktionsmethode verweist. Als Parameter wird der anzuzeigende Text sowie der Aktionsname und bei Bedarf der Name des Controllers übergeben. Der Controllername muss dabei nur übergeben werden, wenn der Link sich auf einen anderen Controller bezieht.

Um einen Link zu erzeugen, welcher es erlaubt, **alle Routenparameter** festzulegen, so verwenden wir üblicherweise die Funktion `RouteLink()`. Dieser werden als Übergabeparameter der anzuzeigende Linktext und die Routenwerte übergeben (meist in Form eines anonymen Objekts). Gibt es mehrere Routen, so muss der Funktion als weiterer Parameter (anzugeben zwischen Linktext und Routenwerte) der Name der Route übergeben werden. Mit diesem erfolgt dann die Zuordnung

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Ansicht](#)

```

37 |         </tr>
38 |         <tr>
39 |             <td></td>
40 |             <td><input type="submit" value="Absenden" /></td>
41 |         </tr>
42 |     </table>
43 | </form>
44 | <br />
45 | @Html.ActionLink("Hier klicken um 2. Seite anzuzeigen ...", "ZweiteSeite")
46 | </body>
47 | </html>
    
```



## Layouts

Normalerweise besitzt eine Website ein **einheitliches und durchgängiges Layout und Design**. Style-Regeln (i. d. R. CSS-Regeln) sind in den meisten Fällen in separaten Dateien notiert und werden mittels eines `link`-Elements eingebunden. Das „rohe“ Layout (also z. B. die verschiedenen `div`-Blöcke, welche den **Seitenaufbau** angeben) wird hingegen in HTML-Dateien notiert. Wenn es nun aber mehrere Ansichten gibt, wird dann dieses Layout mehrmals notiert? Grundsätzlich müsste dies so sein, da ja auf jeder Seite das Layout benötigt wird. Dies wäre jedoch unnötige Verschwendung von Speicherplatz und vor allem ein enormer Aufwand, wenn Sie z. B. das Layout ändern wollen. Um dieses Problem zu umgehen, haben Sie in ASP.NET WebForms die Technologie der Masterseiten kennengelernt. Für ASP.NET MVC-Anwendungen gibt es eine ähnliche Technologie, welche sich **Layout-Seiten** nennt.

Eine Layout-Seite befindet sich, nachdem diese erstellt wurde, ebenfalls im Ordner `Views` und beginnt im Namen in der Regel mit einem Unterstrich (z. B. `_Layout.cshtml` oder `_LayoutPage.cshtml`). Eine Layout-Seite kann dabei ebenfalls im ASPX- oder Razor-Syntax geschrieben werden. Die **Verbindung zwischen Layout- und Inhaltsseite** wird über die Eigenschaft `ViewBag` geschaffen. In dieser können nach belieben Eigenschaften „erzeugt“ werden und von der Layout- und Inhaltsseite verwendet werden. Die Layout-Seite verwendet die `ViewBag`-Eigenschaft zur Ausgabe von Inhalten in Form von **Platzhaltern**, wohingegen in der Inhaltsseite die Eigenschaften und somit die Platzhalter mit Inhalt gefüllt werden. Zudem wird in der Inhaltsseite über den Funktionsaufruf `RenderBody()` der **Inhalt der Inhaltsseite in der Layout-Seite eingebettet**. Um in einer Inhaltsseite die Referenz zu einer Layout-Seite festzulegen, wird die Eigenschaft `Layout` der Seite gesetzt.

Im folgenden Beispiel sehen Sie zwei Ansichten (`Index` und `Seite2`), wovon beide die Layout-Seite `_LayoutPage` verwenden. Die Inhaltsseiten unterscheiden sich dabei hauptsächlich vom Inhalt. Über die `ViewBag`-Eigenschaft werden zudem für jede Seite individuelle Titel sowie eine unterschiedliche Hintergrund-Farbe (zur Verdeutlichung des Beispiels) festgelegt.

### Steuerung (HomeController.cs):

```

1 | using System;
2 | using System.Collections.Generic;
3 | using System.Linq;
4 | using System.Web;
5 | using System.Web.Mvc;
6 |
7 | namespace HWhBsp.Layout.Controllers
8 | {
9 |     public class HomeController : Controller
10 |    {
11 |        public ActionResult Index()
12 |        {
13 |            return View();
14 |        }
15 |
16 |        public ActionResult Seite2()
17 |        {
18 |            return View();
19 |        }
20 |    }
21 | }
    
```

### Layout-Seite (\_LayoutPage.cshtml):

```

1 | <!DOCTYPE html>
2 | <html>
3 |     <head>
4 |         <title>@ViewBag.Title - Layout - ASP.NET Code-Beispiel</title>
5 |
6 |         <meta charset="utf-8" />
7 |
8 |         <meta name="robots" content="noindex,nofollow" />
9 |         <meta name="publisher" content="Homepage-Webhilfe" />
10 |    </head>
11 |
12 |    <body>
13 |        <div style="width: 600px; padding: 5px; background-color: @ViewBag.Farbe">
14 |            <h1>@ViewBag.Title</h1>
15 |            @RenderBody()
16 |            <br /><br />
17 |            <i>Copyright &copy; 2016 by Homepage-Webhilfe</i>
18 |        </div>
19 |    </body>
    
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Ansicht](#)

20 | [</html>](#)

## Ansicht (Index.cshtml):

```

1  @{
2      ViewBag.Title = "Startseite";
3      ViewBag.Farbe = "yellow";
4      Layout = "~/Views/_LayoutPage.cshtml";
5  }
6  <p>...</p>
7  <p>...</p>
8  <br />
9  @Html.ActionLink("Zur Seite 2", "Seite2")

```

## Ansicht (Seite2.cshtml):

```

1  @{
2      ViewBag.Title = "Seite 2";
3      ViewBag.Farbe = "orange";
4      Layout = "~/Views/_LayoutPage.cshtml";
5  }
6  <p>...</p>
7  <p>...</p>
8  <p>...</p>
9  <br />
10 @Html.ActionLink("Zur Startseite", "Index")

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

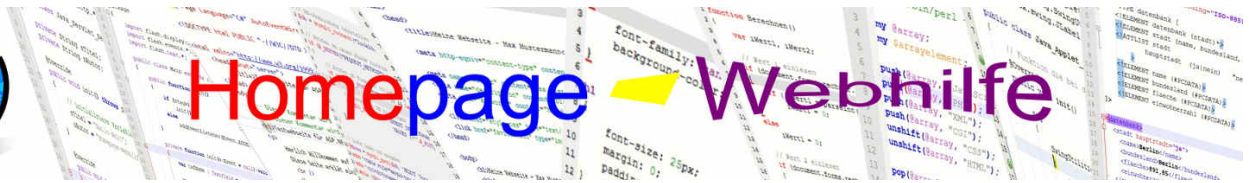
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Modell](#)

## Modell

**Inhalt dieser Seite:**  
1. Metadaten



Bei dem Modell einer MVC-Anwendung handelt es sich um eine einfache C#-Klasse. Die Daten, welche im Modell „abgelegt“ werden sollen, werden dabei in **Eigenschaften gespeichert**. Die Eigenschaften müssen über den Zugriffsmodifizierer `public` verfügen. Das Modell (oder auch als Datenmodell bezeichnet) wird sowohl von der Steuerung (zum **Laden und Speichern der Daten**) als auch von der Ansicht (zum **Anzeigen der Daten**) verwendet. Datenmodelle befinden sich im Ordner `Models`.

Bei Formularen, wo in der Regel auch Modelle zum Einsatz kommen, ist die Aktionsmethode meist überladen, wovon die eine für die HTTP-Methode GET und die andere für die HTTP-Methode POST bestimmt ist. Die POST-Aktionsmethode besitzt als Übergabeparameter ein Datenmodell. Dieses Datenmodell enthält bereits die Daten aus dem Formular. Die Daten werden dabei vom MVC-Framework mit Hilfe der eingehenden Anfrage in dem Objekt gespeichert und an die Aktionsmethode übergeben. Deshalb kann dieses auch wieder direkt der `View()`-Methode übergeben werden. Dadurch bleiben dann die Formularinhalte erhalten. In der GET-Methode wird im Regelfall (und nur in Bezug auf Formulare) eine „einfache“ Objektinstanz übergeben, andernfalls müssten alle Zugriffe auf das Datenmodell in der Ansicht auf `null` abgefangen werden.

Um der Ansicht ein Modell zuzuweisen, notieren wir `@model` gefolgt von dem Klassennamen inkl. dem Namensraum. Wollen wir als Modell eine Liste verwenden, so können wir z. B. auch `@model System.Collections.Generic.List<MeineKlasse>` notieren. Das Datenmodell wird vom **Controller der Ansicht übergeben**. Dafür wird das Modell-Objekt der `View()`-Methode übergeben. Um innerhalb der Ansicht auf das Modell zuzugreifen, können wir die Eigenschaft `Model` verwenden. Die Eigenschaft `Model` ist `null`, wenn kein Modell übergeben wurde. Im Regelfall ist eine Notation dieser Eigenschaft jedoch nicht notwendig, da unsere Ansicht bei Verwendung eines Datenmodells nicht mehr von der Klasse `ViewPage` bzw. `WebViewPage` abgeleitet ist, sondern von der Klasse `ViewPage<TModel>` bzw. `WebViewPage<TModel>`. Um **Eingabefelder**, Auswahllisten etc. zu erstellen, welche in Verbindung mit einer Eigenschaft aus dem Datenmodell stehen, nutzen wir ebenfalls HTML-Hilfsfunktionen. Jedoch verwenden wir nun z. B. an Stelle der Funktion `TextBox()` die Funktion `TextBoxFor()`, statt der Funktion `DropDownList()` `DropDownListFor()` etc.. Alle diese Hilfsfunktionen für die „Formularerstellung“ lassen sich also mit dem Hinzufügen von `For` in den Funktionsnamen in diese „neuen“ Hilfsfunktionen umwandeln. Bei diesen Funktionen fallen die Parameter für Feldname und Feldwert natürlich weg. Als erster Parameter muss immer ein Ausdruck (mittels LINQ) formuliert werden, mit welcher die **Eigenschaft im Datenmodell selektiert** wird. Die gerade genannten Funktionen (wie z. B. `TextBoxFor()`) kommen übrigens ebenfalls nicht mehr aus der Klasse `HtmlHelper`, sondern aus der Klasse `HtmlHelper<TModel>`. Diese Technologie ermöglicht uns deshalb, dass wir uns direkt, und somit ohne weitere Angabe auf das Datenmodell, welches in der Eigenschaft `Model` hinterlegt ist, beziehen.

Das folgende Beispiel stellt ein typisches Beispiel für die **Realisierung eines Kontaktformulars** dar. Dabei gibt es die Aktionsmethode `Index()` im `HomeController`, welche die Ansicht anzeigt. Der Controller und die Ansicht greifen auf das Datenmodell, welches hier durch die Klasse `Contact` beschrieben wird, zu. Auf das Speichern von Informationen in einer Datenbank oder das Senden einer E-Mail, welches in der Aktionsmethode des Controllers durchgeführt werden würde, haben wir aus Gründen der Übersicht und Länge des Beispiels verzichtet.

### Steuerung (HomeController.cs):

```

1 using HWhBsp.Modell.Models;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Web;
6 using System.Web.Mvc;
7
8 namespace HWhBsp.Modell.Controllers
9 {
10     public class HomeController : Controller
11     {
12         [HttpGet]
13         public ActionResult Index()
14         {
15             return View(new Contact());
16         }
17
18         [HttpPost]
19         public ActionResult Index(Contact oContact)
20         {
21             return View(oContact);
22         }
23     }
24 }
    
```

### Modell (Contact.cs):

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5
6 namespace HWhBsp.Modell.Models
7 {
8     public class Contact
9     {
10         private Dictionary<string, string> _ssAnreden = new Dictionary<string, string>()
11         {
12             { "h", "Herr" },
13             { "f", "Frau" }
14         };
15     }
16 }
    
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Modell](#)

```

15     public Dictionary<string, string> Anreden { get { return _ssAnreden; } }
16
17     public string Anrede { get; set; }
18     public string Name { get; set; }
19     public string Betreff { get; set; }
20     public string Nachricht { get; set; }
21 }
22 }
    
```

Ansicht (Index.cshtml):

```

1  @model HwhBsp.Modell.Models.Contact
2  @{
3      Layout = null;
4  }
5  <!DOCTYPE html>
6  <html>
7      <head>
8          <title>Modell - ASP.NET Code-Beispiel</title>
9
10         <meta charset="utf-8" />
11
12         <meta name="robots" content="noindex,nofollow" />
13         <meta name="publisher" content="Homepage-Webhilfe" />
14     </head>
15
16     <body>
17         <form method="post">
18             <table>
19                 <tr>
20                     <td><b>Anrede:</b></td>
21                     <td>@Html.DropDownListFor(m => m.Anrede, new SelectList(Model.Anreden, "Key", "Value"))</td>
22                 </tr>
23                 <tr>
24                     <td><b>Name:</b></td>
25                     <td>@Html.TextBoxFor(m => m.Name)</td>
26                 </tr>
27                 <tr>
28                     <td><b>Betreff:</b></td>
29                     <td>@Html.TextBoxFor(m => m.Betreff)</td>
30                 </tr>
31                 <tr>
32                     <td><b>Nachricht:</b></td>
33                     <td>@Html.TextAreaFor(m => m.Nachricht)</td>
34                 </tr>
35                 <tr>
36                     <td></td>
37                     <td><input type="submit" value="Absenden" /></td>
38                 </tr>
39             </table>
40         </form>
41     </body>
42 </html>
    
```



## Metadaten

Mit Hilfe von Metadaten ist es möglich, die Eigenschaften eines Modells genauer zu beschreiben. Metadaten werden in Form von Attributen oberhalb der Eigenschaft angegeben. Die Metadaten-Attribute befinden sich im Namensraum `System.ComponentModel`. Die folgende Tabelle zeigt die wichtigsten Attribute auf:

<b>DefaultValue</b>	Legt den Standard-Wert der Eigenschaft fest.
<b>DisplayName</b>	Legt den Anzeigenamen der Eigenschaft fest (dieser wird u. a. bei den Standard-Fehlermeldungen verwendet).
<b>ReadOnly</b>	Legt fest, dass die Eigenschaft bzw. das Feld nur gelesen werden kann.

Neben den oben genannten Attributen gibt es noch einige weitere Attribute, die vor allem zur **Validierung** benutzt werden. Diese Attribute befinden sich im Namensraum `System.ComponentModel.DataAnnotations` und werden als Datenanmerkungen (engl. *data annotations*) bezeichnet.

<b>DataType</b>	Legt den Datentyp der Eigenschaft fest.
<b>Range</b>	Legt den Wertebereich (Unter- und Obergrenze) der Eigenschaft fest.
<b>RegularExpression</b>	Legt das Format in Form eines regulären Ausdrucks der Eigenschaft fest.
<b>Required</b>	Legt fest, dass die Eigenschaft bzw. das Feld ein Pflichtfeld ist.
<b>StringLength</b>	Legt die Zeichenkettenlänge (Unter- und Obergrenze) der Eigenschaft fest.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Modell](#)

Um innerhalb der Ansicht auf den festgelegten **Anzeigenamen** zuzugreifen, können wir die Funktion `DisplayNameFor()` der `HtmlHelper`-Klasse verwenden. Für einzeilige Eingabefelder empfiehlt sich zudem die Verwendung der Funktion `EditorFor()` an Stelle von `TextBoxFor()`, da dadurch das `type`-Attribut im `input`-Element passend zum festgelegten Datentyp gesetzt wird.

Um innerhalb der Ansicht auf die Validierungsinformation zuzugreifen, können wir die Funktion `ValidationSummary()` aufrufen. Diese gibt als Rückgabe eine **Liste mit allen Fehlern** zurück. Die Funktion `ValidationMessage()` kann dazu verwendet werden, um den Fehlertext für ein einzelnes Feld zurückzugeben. Die Fehlermeldungen werden entweder vom MVC-Framework gewählt oder können über die Eigenschaft `ErrorMessage` der verschiedenen Attribute festgelegt werden. Um innerhalb des Controllers die **Gültigkeit des Datenmodells zu prüfen**, kann die Eigenschaft `ModelState.IsValid` geprüft werden.

Das folgende Beispiel zeigt ein erweitertes Beispiel des obigen Kontaktformulars:

## Steuerung (HomeController.cs):

```

1  using HWhBsp.Modell_Metadaten.Models;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Web;
6  using System.Web.Mvc;
7
8  namespace HWhBsp.Modell_Metadaten.Controllers
9  {
10     public class HomeController : Controller
11     {
12         [HttpGet]
13         public ActionResult Index()
14         {
15             return View(new Contact());
16         }
17
18         [HttpPost]
19         public ActionResult Index(Contact oContact)
20         {
21             return View(oContact);
22         }
23     }
24 }

```

## Modell (Contact.cs):

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.ComponentModel.DataAnnotations;
5  using System.Linq;
6  using System.Web;
7
8  namespace HWhBsp.Modell_Metadaten.Models
9  {
10     public class Contact
11     {
12         private Dictionary<string, string> _ssAnreden = new Dictionary<string, string>()
13         {
14             { "h", "Herr" },
15             { "f", "Frau" }
16         };
17         public Dictionary<string, string> Anreden { get { return _ssAnreden; } }
18
19         [Required]
20         public string Anrede { get; set; }
21
22         [Required]
23         public string Name { get; set; }
24
25         [Required]
26         [Range(6, 150)]
27         public int Alter { get; set; }
28
29         [DisplayName("E-Mail-Adresse")]
30         [DataType(DataType.EmailAddress)]
31         [Required]
32         public string EMailAddress { get; set; }
33
34         [Required]
35         public string Betreff { get; set; }
36
37         [Required]
38         [StringLength(5000, MinimumLength=25, ErrorMessage="Die Nachricht muss zwischen 25 und 5000 Zeichen haben!")]
39         public string Nachricht { get; set; }
40     }

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [ASP.NET MVC](#) » [Modell](#)

```
41 | }
```

## Ansicht (Index.cshtml):

```
1  @model HWhBsp.Modell_Metadaten.Models.Contact
2  @{
3      Layout = null;
4  }
5  <!DOCTYPE html>
6  <html>
7      <head>
8          <title>Modell Metadaten - ASP.NET Code-Beispiel</title>
9
10         <meta charset="utf-8" />
11
12         <meta name="robots" content="noindex,nofollow" />
13         <meta name="publisher" content="Homepage-Webhilfe" />
14     </head>
15
16     <body>
17         <form method="post">
18             <table>
19                 <tr>
20                     <td colspan="2">@Html.ValidationSummary()/>
21                 </tr>
22                 <tr>
23                     <td><b>@Html.DisplayNameFor(m => m.Anrede):</b></td>
24                     <td>@Html.DropDownListFor(m => m.Anrede, new SelectList(Model.Anreden, "Key", "Value"))</td>
25                 </tr>
26                 <tr>
27                     <td><b>@Html.DisplayNameFor(m => m.Name):</b></td>
28                     <td>@Html.EditorFor(m => m.Name)</td>
29                 </tr>
30                 <tr>
31                     <td><b>@Html.DisplayNameFor(m => m.Alter):</b></td>
32                     <td>@Html.EditorFor(m => m.Alter)</td>
33                 </tr>
34                 <tr>
35                     <td><b>@Html.DisplayNameFor(m => m.EmailAddress):</b></td>
36                     <td>@Html.EditorFor(m => m.EmailAddress)</td>
37                 </tr>
38                 <tr>
39                     <td><b>@Html.DisplayNameFor(m => m.Betreff):</b></td>
40                     <td>@Html.EditorFor(m => m.Betreff)</td>
41                 </tr>
42                 <tr>
43                     <td><b>@Html.DisplayNameFor(m => m.Nachricht):</b></td>
44                     <td>@Html.TextAreaFor(m => m.Nachricht)</td>
45                 </tr>
46                 <tr>
47                     <td></td>
48                     <td><input type="submit" value="Absenden" /></td>
49                 </tr>
50             </table>
51         </form>
52     </body>
53 </html>
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

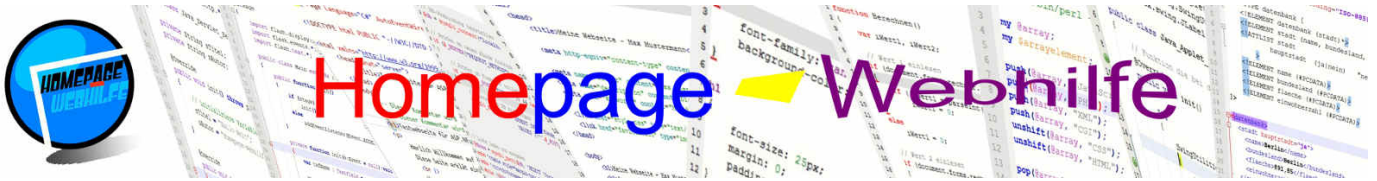
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislöcher



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [ASP.NET](#) » [Abschluss](#)

## Abschluss

In diesem Kapitel und den 2 Unterkapiteln „ASP.NET WebForms“ und „ASP.NET MVC“ haben Sie die **wichtigsten Grundlagen von ASP.NET** sowie einige fortgeschrittene Themen kennengelernt. Mit Hilfe des Erlernten ist es Ihnen möglich, **professionelle und komplexe Webauftritte** mittels C# (oder einer anderen .NET Programmiersprache) zu erstellen und dabei die **Klassen des .NET Frameworks zu nutzen**.

Wie auch bei den anderen Kapiteln können wir hier nicht alle Funktionen, Eigenschaften, Interfaces und Klassen von ASP.NET behandeln, sondern haben uns auf die gängigsten und wichtigsten beschränkt. Eine **Dokumentation über den kompletten Namensraum** von ASP.NET sowie dessen Klassen, Interfaces, Eigenschaften und Funktionen finden Sie auf der [ASP.NET-Referenz von Microsoft](#).

Für serverseitige Programmierung gibt es neben ASP.NET auch noch [PHP](#) und [Perl](#). Ein weiterer Konkurrent zu ASP.NET ist [Java EE](#), wozu die Technologien Java Server Pages, Java Server Faces und Servlets gehören. Zu den genannten Programmiersprachen und Technologien bieten wir hier ebenfalls Tutorials an.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislungen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

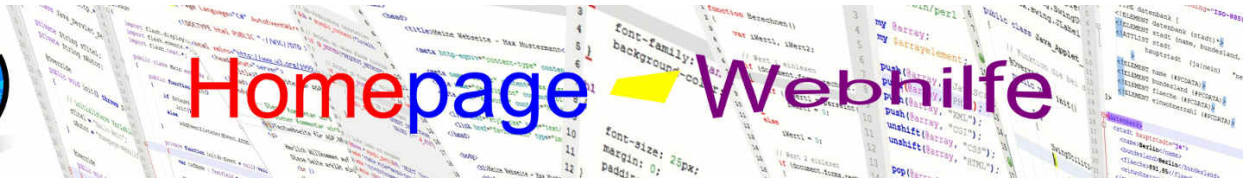
Java EE

XML

# E-Book

## Java EE





Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Einführung](#)

## Einführung



Die **Enterprise Edition der Java Platform** (kurz Java EE) ist eine Spezifikation bzw. ein Teil der Java-Plattform. Der größte Konkurrent der Java-EE-Spezifikation ist dabei Microsofts .NET-Plattform. Ein wichtiger Bestandteil von Java EE sind unter anderem die Webanwendungen, mit welchen wir uns in diesem Thema beschäftigen möchten.

Die Java EE Plattform umfasst in Bezug auf Webanwendungen einige Technologien, welche innerhalb dieses Themas noch genauer erläutert werden. Zur Ausführung solcher Anwendungen ist ein **Applikationsserver** notwendig. Ein Applikationsserver unterteilt sich in mehrere Systeme, welche Container genannt werden. Für Webanwendungen reicht auch ein **Web-Container** aus. Mehr dazu später.

Java EE wird auf einem relativ geringen Anteil (ca. 3%) an Webseiten für die **serverseitige Programmierung** eingesetzt. Dabei wird Java EE hauptsächlich auf „großen“ bzw. „komplexen“ Websites verwendet. Private Websites mit Java EE sind kaum zu finden.

Wer bereits in Java programmiert hat, der hat mit Java-Webanwendungen den **Vorteil**, dass keine neue Sprache erlernt werden muss und die ganzen Klassen der Java Plattform eingesetzt werden können. Auf Grund der geringen **Nachfrage** gibt es jedoch nur sehr wenig Anbieter für Java-Hosting.

Bitte beachten Sie, dass für dieses Kapitel **Kenntnisse in der Programmiersprache Java** erforderlich sind. Bei Bedarf können Sie sich auch unseren [Crashkurs](#) anschauen.

### Inhalt dieser Seite:

1. Geschichte
2. Entwicklung und Webserver
3. Funktionsweise und Technologien

## Geschichte

Die **erste Version** von Java EE (damals noch als Java 2 Platform Enterprise Edition bzw. kurz Java2EE bezeichnet) erschien Ende des Jahres 1999. Bis zum Jahre 2003 folgten weitere Versionen bis zur Version 1.4.

Anschließend wurde der **Name der Plattform** in die heute bekannte Bezeichnung Java Platform Enterprise Edition (kurz Java EE) geändert. Die „erste“ Version von Java EE erschien im Mai 2006 und trug die Versionsnummer 5. 2009 folgte dann bereits die Version 6. Im Jahr 2013 ist die Version 7 veröffentlicht worden.

## Entwicklung und Webserver

Zur Entwicklung von Java EE bzw. Java-Webanwendungen werden in der Regel **Entwicklungsumgebungen** (kurz IDE für *Integrated Development Environment*) verwendet, welche einen Quellcode-Editor, eine Schnittstelle für einen Compiler und einen Debugger enthält. Ein solche IDE ist z. B. NetBeans oder Eclipse. Wir haben für die Erstellung der in diesem Thema enthaltenen Projekte NetBeans verwendet.

Wie bereits oben erwähnt, ist für die Ausführung von Java-EE-Anwendungen ein sogenannter **Applikationsserver** notwendig. Hier sind z. B. die Open-Source-Server Apache Geronimo und GlassFish sowie die kommerziellen Server Oracle Application Server und IBM WebSphere Application Server zu nennen. Natürlich gibt es noch einige weitere.

Als Alternative zu einem „vollständigen“ Applikationsserver können auch Programme verwendet werden, welche lediglich einen **Web-Container** (oft auch als Servlet- oder JSP-Container bezeichnet) implementieren. Ein solcher Web-Container ist für die Ausführung von Servlets und JavaServer Pages ausreichend. Der bekannteste Web-Container ist Apache Tomcat, welcher unter anderem im Software-Paket XAMPP enthalten ist.

**Wichtig:** Bei der Verwendung von JSP ist ein JDK (Java Development Kit) auf dem Webserver notwendig.

## Funktionsweise und Technologien

Der Java-Anwendungsserver stellt die Laufzeitumgebung für die (Web-)Anwendungen dar, wobei der Server zudem diverse Funktionen wie z. B. Sicherheit und Installations-Hilfen (Deployment) enthält. Die Ausführung Ihrer Anwendung erfolgt dabei durch den Anwendungsserver oder Container, wovon die Zugriffe vom Server gekapselt und auch gesteuert werden können.

Java EE enthält diverse Technologien, welche auch als APIs bezeichnet werden. Für Webanwendungen sind dabei hauptsächlich die Technologien [Servlets](#) und [JavaServer Pages \(JSP\)](#) interessant, welche wir in den Unterkapiteln dieses Kapitels auch noch genauer erläutern werden. Weitere APIs sind z. B. Enterprise JavaBeans (EJB), JavaMail und JavaServer Faces (JSF).

Wenn Sie noch keine Erfahrungen mit Java Webanwendungen haben, empfehlen wir Ihnen dringend zu allererst das Kapitel Servlet zu bearbeiten, bevor Sie sich bei Interesse noch mit JSP beschäftigen.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

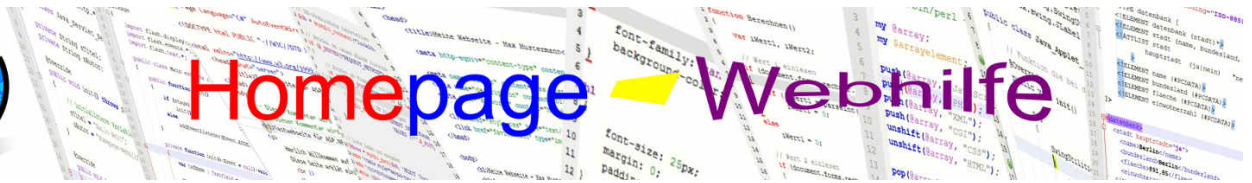
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Grundlagen](#)

## Grundlagen



Servlets sind Java-Klassen, welche in einer Java-Webanwendung zum Einsatz kommen. Dabei empfängt ein Servlet bestimmte Anfragen von Clients (abhängig von der URL) und kann diese beantworten. Der Inhalt kann dabei dynamisch generiert werden.

Java-Webanwendungen benötigen einen speziellen **Webserver, die einen Web- bzw. Servlet-Container enthalten**. Der bekannteste Webserver hierfür ist wohl Apache Tomcat. Für einige Webserver (z. B. den Microsoft-Webserver IIS) können Module verwendet werden, um auch dort Java-Webanwendungen ausführen zu können.

Bei dem Begriff Servlet handelt es sich um ein sogenanntes Kofferwort, welches sich aus den Begriffen Server und Applet (clientseitige Java-Anwendungen innerhalb einer Webseite) zusammensetzt. Servlets sind also wie Applets „**kleine**“ **Anwendungen, die in einem Container laufen**. Servlets laufen dabei in einem Container (um genauer zu sein dem Web- bzw. Servlet-Container) auf dem Webserver.

Servlets bauen aus technischer Sicht auf der **CGI-Schnittstelle** auf. Im Gegensatz zu anderen serverseitigen Technologien, wie z. B. PHP oder ASP.NET, werden bei Servlets HTML- und Programmcode nicht gemischt, vielmehr wird die komplette Ausgabe (also z. B. der HTML-Code) vom Programm (in diesem Fall also vom Java-Programm) erzeugt.

### Inhalt dieser Seite:

1. Schnittstellen
2. Konfiguration
3. Erstes Servlet

## Schnittstellen

Eine Servlet-Klasse wird von der Klasse `HttpServlet` (Package `javax.servlet.http`) abgeleitet. Diese Klasse implementiert unter anderem das Interface `Servlet` (Package `javax.servlet`).

Um auf eingehende Anfragen zu reagieren, müssen zu allererst die Funktionen `doGet()` und `doPost()` sowie ggf. auch `doPut()` und `doDelete()` überschrieben werden. Die oben genannten Funktionen werden immer dann aufgerufen, wenn eine Anfrage zum Servlet mit der jeweiligen HTTP-Methode gesendet wird, d. h. sendet jemand eine GET-Anfrage an ein Servlet, so wird dessen Funktion `doGet()` aufgerufen. Möchten Sie für alle HTTP-Methoden die gleiche Funktion verwenden, so können Sie auch die Methode `service()` überschreiben. Alle diese Funktionen geben keinen Wert zurück und besitzen als Übergabeparameter ein Objekt der Interfaces `HttpServletRequest` und `HttpServletResponse`. Des Weiteren können diese Methoden Ausnahmen der Klassen `ServletException` und `IOException` werfen.

Das Interface `HttpServletRequest` ermöglicht den Zugriff auf einige Anfrage-Informationen, wie z. B. die HTTP-Methode, die Anfragezeile oder die URL. Zudem können auch Formulardaten (GET/URL-Parameter sowie POST-Parameter), die Session des Besuchers und Cookies abgerufen werden. Durch das Interface `HttpServletResponse` ist es möglich, Informationen über die HTTP-Antwort abzurufen, aber auch zu ändern. Des Weiteren ist es auch möglich, Weiterleitungen oder das Setzen von Cookies mit Hilfe dieser Interfaces zu realisieren. Beide Interfaces werden wir jedoch im nächsten Thema noch genauer besprechen.

Wie viele andere Klassen besitzt auch die `HttpServlet`-Klasse die Funktionen `init()` und `destroy()`. Diese können überschrieben werden, um Ressourcen für das Servlet zu reservieren und am Ende wieder freizugeben. Bei dem Aufruf eines Servlets wird vom Servlet-Container als erstes eine Instanz des Objekts erstellt und dann die `init()`-Methode aufgerufen. Anschließend wird eine der `doX()`-Methoden bzw. die `service()`-Methode aufgerufen. Am Ende wird die `destroy()`-Funktion aufgerufen und letztendlich die Objektinstanz wieder gelöscht.

## Konfiguration

Servlets bzw. Java-Webanwendungen müssen grundsätzlich konfiguriert werden. Dafür gibt es die Datei `web.xml`, welche auch als **Deployment Descriptor** bezeichnet wird. Diese Datei (im Falle einer Web-Anwendung) besitzt das Wurzelement `web-app`. In dieser Datei wird üblicherweise das Attribut `version` sowie einige XML-spezifische Attribute (`xmlns` u. a.) angegeben. Dies sieht dann wie folgt aus:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
4 </web-app>

```

Jedes Servlet, welches nachher von außen zugänglich sein soll, muss in der `web.xml`-Datei erstmal registriert werden. Dazu wird das Element `servlet` angegeben. In diesem befinden sich wiederum die Elemente `servlet-name` und `servlet-class`. Der Wert innerhalb von `servlet-name` legt den internen (für die XML-Datei geltenden) Namen des Servlets fest. Im Element `servlet-class` wird die Klasse des Servlets inkl. dem Package angegeben.

```

1 <servlet>
2   <servlet-name>HelloServlet</servlet-name>
3   <servlet-class>de.hwh.bsp.hallowelt.HelloServlet</servlet-class>
4 </servlet>

```

Um nun eine gewisse URL mit einem Servlet zu verbinden, benötigen wir das Element `servlet-mapping`, welchem die Elemente `servlet-name` (dort wird der interne Name des Servlets „von oben“ angegeben) und `url-pattern` untergeordnet werden. Als Wert des Elements `url-pattern` kann ein **URL-Muster** angegeben werden. Dabei ist es auch möglich, mit dem Wildcard-Zeichen `*` zu arbeiten. Mit dem Muster `/Download/*.zip` könnte man bspw. alle Anfragen auf eine ZIP-Datei aus dem Ordner `/Download/` auf ein gewisses Servlet lenken.

```

1 <servlet-mapping>
2   <servlet-name>HelloServlet</servlet-name>
3   <url-pattern>/</url-pattern>
4 </servlet-mapping>

```

Neben den genannten Elementen gibt es noch viele weitere, die dazu genutzt werden können, eine Web-Anwendung zu konfigurieren. Hierzu zählt z. B. das automatische Timeout von Sessions.

## Erstes Servlet

Nachdem wir uns nun mit der grundlegenden Theorie in Bezug auf Servlets beschäftigt haben, wollen wir nun unser erstes Servlet erstellen. Alle in diesem Tutorial

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

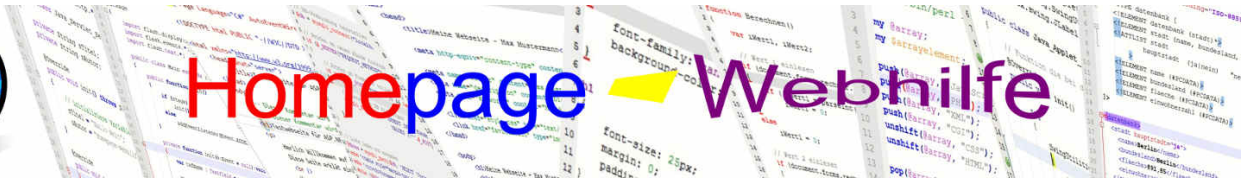
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Grundlagen](#)

enthaltenen Beispiele wurden mit der **Entwicklungsumgebung NetBeans** erstellt und mit dem **JDK 8 (mit der Java EE Version 6)** kompiliert.

Um eine Java-Webanwendung zu erstellen, müssen wir als erstes **ein neues Projekt anlegen**. Bei der Projekt-Erstellung müssen wir unter „Categories“ „Java Web“ auswählen. In der Auswahlbox „Project“ wählen wir „Web Application“.

Als nächstes werden Name und Speicherort des Projekts festgelegt. Anschließend werden die Servereinstellungen festgelegt. Dort muss der Server, die Java-EE-Version und der Pfad ausgewählt bzw. eingegeben werden. Wurde noch kein Server angelegt, was bei der ersten Erstellung des ersten Projekts der Fall ist, so müssen Sie diesen zuerst erstellen. Hier können Sie, sofern Sie mit XAMPP entwickeln, NetBeans mit **dem in XAMPP enthaltenen Apache Tomcat Webserver** verbinden. Dazu wählen Sie als erstes „Apache Tomcat or TomEE“ aus und geben anschließend den Pfad des Server-Verzeichnisses sowie einen Benutzernamen und ein Passwort ein.

Im letzten Schritt können noch Frameworks ausgewählt werden, welche eingebunden werden sollen. Für unsere Servlet-Beispiele benötigen wir jedoch keine Frameworks.

Von NetBeans wird bei den oben genannten Schritten eine JSP-Seite (Datei `index.jsp`) erzeugt. Diese können Sie löschen, da wir diese zurzeit nicht benötigen. Unter dem Projektdrorder „Source Packages“ können Sie bei Bedarf ein Package und anschließend **ein Servlet anlegen**. Nach dem Festlegen des Namens und des Speicherorts können Sie das Servlet dem Deployment Descriptor hinzufügen. Auch hier ist uns der Assistent von NetBeans eine große Hilfe. Dadurch werden die Einstellungen automatisch in die Datei `web.xml` hinzugefügt. Beim Erstellen des ersten Servlets wird die Datei `web.xml` angelegt.

Das Erstellen eines neuen Servlets mit Hilfe des Assistenten erzeugt eine Java-Quelldatei, in welcher die Klasse des Servlets definiert ist. In der Klasse werden die Methoden `doGet()`, `doPost()` und `getServletInfo()` überschrieben. `doGet()` und `doPost()` rufen die lokale Funktion `processRequest()` auf.

Das Projekt kann direkt von der IDE heraus **in den Webserver geladen und das Ergebnis betrachtet** werden. Auch das **Debuggen von Java Web-Anwendungen ist möglich**.

Im folgenden Beispiel wurde der Quellcode etwas gekürzt und der Inhalt der Funktion `processRequest()` verändert. Bei diesem und den folgenden Beispielen haben Sie die Möglichkeit, sich das Ergebnis anzuschauen, indem Sie auf die Lupe klicken. Ein Klick auf den Pfeil nach unten startet den Download des NetBeans-Projekts.

#### Servlet (HelloServlet.java):

```

1 package de.hwh.bsp.hallowelt;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 import java.util.Date;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10
11 public class HelloServlet extends HttpServlet
12 {
13     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
14         throws ServletException, IOException
15     {
16         // Ausgabe als Text-Datei
17         response.setContentType("text/plain;charset=UTF-8");
18
19         // Ausgabe durchführen
20         PrintWriter out = response.getWriter();
21         out.println("Hallo Welt!");
22         out.println();
23         out.println("Aktuelles Datum und Uhrzeit: " + (new Date()).toString());
24         out.close();
25     }
26
27     @Override
28     protected void doGet(HttpServletRequest request, HttpServletResponse response)
29         throws ServletException, IOException
30     {
31         processRequest(request, response);
32     }
33
34     @Override
35     protected void doPost(HttpServletRequest request, HttpServletResponse response)
36         throws ServletException, IOException
37     {
38         processRequest(request, response);
39     }
40 }

```

#### Konfiguration (web.xml):

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
4     <servlet>
5         <servlet-name>HelloServlet</servlet-name>
6         <servlet-class>de.hwh.bsp.hallowelt.HelloServlet</servlet-class>
7     </servlet>
8     <servlet-mapping>
9         <servlet-name>HelloServlet</servlet-name>

```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Grundlagen](#)

```
9      <url-pattern>/</url-pattern>
10     </servlet-mapping>
11     <session-config>
12       <session-timeout>
13         30
14       </session-timeout>
15     </session-config>
16 </web-app>
```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

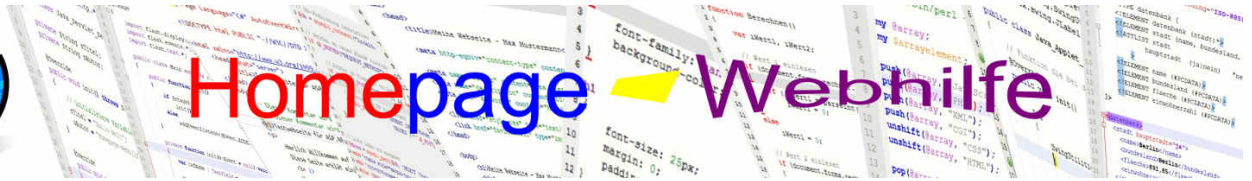
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Anfrage und Antwort](#)

## Anfrage und Antwort

### Inhalt dieser Seite:

1. Anfrage-Informationen
2. Antwort-Informationen

In Servlets werden uns durch die Funktionen `doX()` und `service()` Objekte der Interfaces `HttpServletRequest` und `HttpServletResponse` bereitgestellt (Übergabe als Methodenparameter). Beide Interfaces enthalten einige Funktionen zum Abrufen und sogar zum Ändern (nur bei dem Interface `HttpServletResponse`) von Informationen. Dadurch kann z. B. der Inhaltstyp (Header `Content-Type`) dynamisch gesetzt werden oder eine Umleitung durchgeführt werden. In diesem Thema wollen wir uns mit diesen beiden Interfaces genauer beschäftigen.

### Anfrage-Informationen

Ein Objekt von `HttpServletRequest` ermöglicht das **Abrufen verschiedener Informationen, welche zur HTTP-Anfrage gehören**. Der Abruf erfolgt über die sogenannten Zugriffsfunktionen (`getX()`). Des Weiteren werden einige Funktionen vom Interface `ServletRequest` geerbt. Dort existieren auch einige solcher Zugriffsfunktionen. Die folgende Tabelle zeigt die verschiedenen Funktionen zum Abrufen von Anfrage-Informationen:

<code>getCharacterEncoding()</code>	Gibt die Zeichenkodierung zurück.
<code>getContentLength()</code>	Gibt die Länge des Inhalts zurück.
<code>getContentType()</code>	Gibt den Inhaltstyp zurück.
<code>getHeader()</code>	Gibt den Wert des angegebenen Header-Felds zurück.
<code>getMethod()</code>	Gibt die HTTP-Methode zurück.
<code>getLocalAddr()</code>	Gibt die IP-Adresse des lokalen Geräts zurück.
<code>getLocalName()</code>	Gibt den Hostnamen des lokalen Geräts zurück.
<code>getLocalPort()</code>	Gibt den Port des lokalen Geräts zurück.
<code>getProtocol()</code>	Gibt das angefragte Protokoll zurück.
<code>getQueryString()</code>	Gibt die Anfrage-Zeichenkette zurück.
<code>getRemoteAddr()</code>	Gibt die IP-Adresse des entfernten Geräts zurück.
<code>getRemoteHost()</code>	Gibt den Hostnamen des entfernten Geräts zurück.
<code>getRemotePort()</code>	Gibt den Port des entfernten Geräts zurück.
<code>getRequestURI()</code>	Gibt die angefragte URI zurück.
<code>getServerName()</code>	Gibt den Hostnamen des Servers zurück.
<code>getServerPort()</code>	Gibt den Port des Servers zurück.

Hier nun ein Beispiel, in welchem einige Informationen abgerufen und ausgegeben werden:

```

1  package de.hwh.bsp.anfrageinfo;
2
3  import java.io.IOException;
4  import java.io.PrintWriter;
5  import javax.servlet.ServletException;
6  import javax.servlet.http.HttpServlet;
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9
10 public class RequestServlet extends HttpServlet
11 {
12     @Override
13     protected void service(HttpServletRequest request, HttpServletResponse response)
14         throws ServletException, IOException
15     {
16         // Ausgabe als Text-Datei
17         response.setContentType("text/plain;charset=UTF-8");
18
19         // Ausgabe durchführen
20         PrintWriter out = response.getWriter();
21         out.println("lokale IP-Adresse: " + request.getLocalAddr());
22         out.println("lokaler Hostname: " + request.getLocalName());
23         out.println("lokaler Port: " + request.getLocalPort());
24         out.println("HTTP-Methode: " + request.getMethod());
25         out.println("Protokoll: " + request.getProtocol());
26         out.println("Query String: " + request.getQueryString());
27         out.println("IP-Adresse: " + request.getRemoteAddr());
28         out.println("Hostname: " + request.getRemoteHost());
29         out.println("Port: " + request.getRemotePort());
30         out.println("URI: " + request.getRequestURI());
31         out.println("Servername: " + request.getServerName());
32         out.println("Serverport: " + request.getServerPort());

```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Anfrage und Antwort](#)

```
33 |         out.close();
34 |     }
35 | }
```



Übrigens: Die meisten Funktionen geben `null` zurück, wenn die angefragte Information (bzw. das angefragte Header-Feld) nicht vorhanden ist.

## Antwort-Informationen

Bei den Antwort-Informationen (Interface `HttpServletResponse`) gibt es unterschiedliche Zugriffsfunktionen, wovon die eine Gruppe die **Informationen abrufen** (`getX()`) und die andere Gruppe die **Informationen ändern bzw. setzen** (`setX()`). Dies kommt daher, da eine **HTTP-Antwort** beeinflusst werden kann, wovon die HTTP-Anfrage vom Client kommt und nicht verändert werden kann. Auch hier haben wir eine Tabelle mit den unterschiedlichen Funktionen zum Abrufen und Setzen von Informationen zusammengestellt:

<code>getCharacterEncoding()</code>	Gibt die Zeichenkodierung zurück bzw. setzt diese.
<code>setCharacterEncoding()</code>	
<code>setContentLength()</code>	Setzt die Länge des Inhalts.
<code>getContentType()</code>	Gibt den Inhaltstyp zurück bzw. setzt diesen.
<code>setContentType()</code>	
<code>getHeader()</code>	Gibt den Wert des angegebenen Header-Felds zurück bzw. setzt diesen.
<code>setHeader()</code>	
<code>getStatus()</code>	Gibt den Statuscode zurück bzw. setzt diesen.
<code>setStatus()</code>	
<code>getWriter()</code>	Gibt den Stream für das Senden des Antwort-Inhalts zurück.
<code>sendError()</code>	Sendet eine Fehlermeldung (HTTP-Fehler) an den Client zurück.
<code>sendRedirect()</code>	Sendet eine Umleitung an den Client zurück.

In diesem und in den 2 vorherigen Beispielen haben wir ja bereits die Funktion `setContentType()` zum Setzen des „Inhaltstyps“ und `getWriter()` zum Abruf des ausgehenden Streams benutzt. Diese Funktionen werden wir auch in diesem und in den folgenden Beispielen verwenden. Zusätzlich rufen wir in diesem Beispiel einige Informationen aus der `HttpServletResponse` ab und geben diese aus:

```
1 | package de.hwh.bsp.antwortinfo;
2 |
3 | import java.io.IOException;
4 | import java.io.PrintWriter;
5 | import javax.servlet.ServletException;
6 | import javax.servlet.http.HttpServlet;
7 | import javax.servlet.http.HttpServletRequest;
8 | import javax.servlet.http.HttpServletResponse;
9 |
10 | public class ResponseServlet extends HttpServlet
11 | {
12 |     @Override
13 |     protected void service(HttpServletRequest request, HttpServletResponse response)
14 |         throws ServletException, IOException
15 |     {
16 |         // Ausgabe als Text-Datei
17 |         response.setContentType("text/plain;charset=UTF-8");
18 |
19 |         // Ausgabe durchführen
20 |         PrintWriter out = response.getWriter();
21 |         out.println("Zeichenkodierung: " + response.getCharacterEncoding());
22 |         out.println("Inhalts-Typ: " + response.getContentType());
23 |         out.println("Puffer-Größe: " + response.getBufferSize());
24 |         out.println("HTTP-Status-Code: " + response.getStatus());
25 |         out.close();
26 |     }
27 | }
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Formulardaten](#)

## Formulardaten

In Servlets können Werte von Parametern (unabhängig ob es sich um GET- oder POST-Parameter handelt) über die Methode `getParameter()` eines `HttpServletRequest`-Objekts abgerufen werden. Der Funktion wird dabei als Übergabeparameter der **Name des abzurufenden Parameters** übergeben. Als Rückgabe erhalten Sie den **Wert des Parameters** in Form des Datentyps `String`. Existiert der Parameter nicht, so wird `null` zurückgegeben.

Im folgenden Beispiel wird über die `doGet()`-Methode ein Formular (mit der HTTP-Methode POST) ausgegeben. Die Parameterwerte werden in der `doPost()`-Methode ausgegeben.

```

1  package de.hwh.bsp.formular;
2
3  import java.io.IOException;
4  import java.io.PrintWriter;
5  import javax.servlet.ServletException;
6  import javax.servlet.http.HttpServlet;
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9
10 public class FormServlet extends HttpServlet
11 {
12     @Override
13     protected void doGet(HttpServletRequest request, HttpServletResponse response)
14         throws ServletException, IOException
15     {
16         // Ausgabe als HTML-Seite
17         response.setContentType("text/html;charset=UTF-8");
18
19         // Ausgabe durchführen
20         PrintWriter out = response.getWriter();
21         out.println("<!DOCTYPE html>");
22         out.println("<html>");
23         out.println("    <head>");
24         out.println("        <title>Formulardaten - Java EE Servlet Code-Beispiel</title>");
25         out.println();
26         out.println("        <meta charset=\"utf-8\" />");
27         out.println();
28         out.println("        <meta name=\"robots\" content=\"noindex,nofollow\" />");
29         out.println("        <meta name=\"publisher\" content=\"Homepage-Webhilfe\" />");
30         out.println("    </head>");
31         out.println();
32         out.println("    <body style=\"line-height: 1.5em;\">");
33         out.println("        <form method=\"post\">");
34         out.println("            <b>Bitte füllen Sie das Formular aus:</b><br />");
35         out.println("            Vorname: <input type=\"text\" name=\"vorname\" /><br />");
36         out.println("            Nachname: <input type=\"text\" name=\"nachname\" /><br />");
37         out.println("            <input type=\"submit\" value=\"Absenden\" />");
38         out.println("        </form>");
39         out.println("    </body>");
40         out.println("</html>");
41         out.close();
42     }
43
44     @Override
45     protected void doPost(HttpServletRequest request, HttpServletResponse response)
46         throws ServletException, IOException
47     {
48         // Zeichenkodierung auf UTF-8 setzen (für Formulardaten)
49         request.setCharacterEncoding("UTF-8");
50
51         // Ausgabe als HTML-Seite
52         response.setContentType("text/html;charset=UTF-8");
53
54         // Ausgabe durchführen
55         PrintWriter out = response.getWriter();
56         out.println("<!DOCTYPE html>");
57         out.println("<html>");
58         out.println("    <head>");
59         out.println("        <title>Formulardaten - Java EE Servlet Code-Beispiel</title>");
60         out.println();
61         out.println("        <meta charset=\"utf-8\" />");
62         out.println();
63         out.println("        <meta name=\"robots\" content=\"noindex,nofollow\" />");
64         out.println("        <meta name=\"publisher\" content=\"Homepage-Webhilfe\" />");
65         out.println("    </head>");
66         out.println();
67         out.println("    <body style=\"line-height: 1.5em;\">");
68         out.println("        <b>Ihre Eingaben im Formular waren:</b><br />");
69         out.println("        Vorname: " + request.getParameter("vorname") + "<br />");

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Formular Daten](#)

```
70 |         out.println("        Nachname: " + request.getParameter("nachname") + "<br />");
71 |         out.println("        </body>");
72 |         out.println("</html>");
73 |         out.close();
74 |     }
75 | }
```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

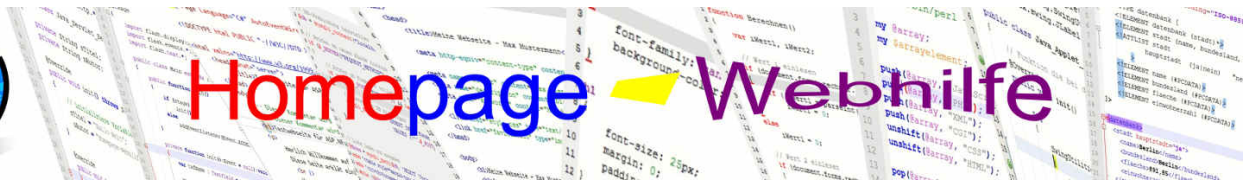
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Session und Cookies](#)

## Session und Cookies

Mit Hilfe von Sessions und Cookies ist es möglich, **anfrage- aber auch seitenübergreifend Daten zu speichern**. Diese Daten werden dabei je nach verwendetem Typ client- oder serverseitig gespeichert. Eine solche Technologie kann z. B. für Warenkörbe oder Merklisten in Online-Shops, die „Angemeldet bleiben“-Funktion in Foren und vielen mehr eingesetzt werden.

### Inhalt dieser Seite:

1. Session
2. Cookies

### Session



Sessions (zu Deutsch Sitzung) erlauben das anfrage- und seitenübergreifende Speichern von Informationen. In einer Session können mehrere sogenannte **Session-Variablen** (im Java-Umfeld auch als Session-Attribute bezeichnet) abgelegt werden.

Die Speicherung der Werte von Session-Variablen erfolgt auf dem Webserver, d. h. diese **können vom Besucher nicht manipuliert werden**. Technisch **realisiert wird eine Session mit Hilfe eines Cookies** (Cookie-Name `JSESSIONID`). Dieses Cookie hat eine Lebensdauer von 0, d. h. das Cookie gilt so lange wie der Browser geöffnet bleibt. Der Server speichert die Session auch nur für einen **begrenzten Zeitraum**. Dieser Zeitraum wird in der Datei `web.xml` (XML-Element `session-timeout`) festgelegt.

Eine Session wird in Java-EE-Anwendungen durch das Interface `HttpSession` (Package `javax.servlet.http`) repräsentiert. Um ein Session-Objekt abzurufen, wird die Funktion `getSession()` des Interfaces `HttpServletRequest` genutzt. Der Funktion kann ein Wert vom Typ `boolean` übergeben werden, welcher angibt, ob die Session, falls diese nicht existiert, erstellt werden soll. `false` gibt an, dass die Session nicht erstellt werden soll und somit gibt die Funktion `getSession()`, wenn bisher keine Session existiert, `null` zurück. Wird der Parameter weggelassen oder wird der Wert `true` übergeben, so wird eine **Session erstellt**, falls noch keine existiert. Bei der „ersten“ Erstellung der Session wird das Cookie `JSESSIONID` angelegt und an den Client zurückgesendet.

Das Interface `HttpSession` enthält drei wichtige Funktionen: `getAttribute()`, `setAttribute()`, `removeAttribute()`. Mit der Funktion `getAttribute()` kann ein **Session-Attribut abgerufen** werden. Existiert das Attribut nicht, so gibt die Funktion `null` zurück. Mit `setAttribute()` kann der **Wert eines Session-Attributs gesetzt** bzw., sofern das Attribut noch nicht existiert, das Session-Attribut angelegt werden. `removeAttribute()` erlaubt das **Entfernen eines Attributs** aus einer Session. Die Werte von Session-Attributen sind vom Typ `Object`. Dadurch ist es möglich, **komplette Objekte einer (eigenen) Klasse zu speichern**.

Im folgenden Beispiel haben Sie zwei Eingabefelder. Die dort eingegebenen Werte werden in der Session gespeichert. Wenn Sie die Beispiel-Ansicht öffnen, dort Werte eingeben, auf „Speichern“ klicken, den Tab schließen und anschließend das Beispiel erneut öffnen, werden Sie feststellen, dass die eingegebenen Werte (Vorname und Nachname) „erhalten geblieben“ sind.

```

1 package de.hwh.bsp.session;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 import javax.servlet.ServletException;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9 import javax.servlet.http.HttpSession;
10
11 public class SessionServlet extends HttpServlet
12 {
13     @Override
14     protected void service(HttpServletRequest request, HttpServletResponse response)
15         throws ServletException, IOException
16     {
17         HttpSession oSession;
18         String sVorname, sNachname;
19
20         // Zeichenkodierung auf UTF-8 setzen (für Formulardaten)
21         request.setCharacterEncoding("UTF-8");
22
23         // Session abrufen oder erstellen (falls diese noch nicht existiert)
24         oSession = request.getSession(true);
25
26         // Prüfen ob die Anfrage mittels der POST-Methode durchgeführt wurde
27         if (request.getMethod() == "POST")
28         {
29             // Werte aus dem Formular abrufen
30             sVorname = request.getParameter("vorname") != null ? request.getParameter("vorname") : "";
31             sNachname = request.getParameter("nachname") != null ? request.getParameter("nachname") : "";
32
33             // Werte in der Session speichern
34             oSession.setAttribute("vorname", sVorname);
35             oSession.setAttribute("nachname", sNachname);
36         }
37         else
38         {
39             // Werte aus der Session abrufen
40             sVorname = oSession.getAttribute("vorname") != null ? oSession.getAttribute("vorname").toString() : "";
41             sNachname = oSession.getAttribute("nachname") != null ? oSession.getAttribute("nachname").toString() : "";
42         }
43
44         // Ausgabe als HTML-Seite
45         response.setContentType("text/html;charset=UTF-8");

```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Session und Cookies](#)

```

46
47 // Ausgabe durchführen
48 PrintWriter out = response.getWriter();
49 out.println("<!DOCTYPE html>");
50 out.println("<html>");
51 out.println("    <head>");
52 out.println("        <title>Session - Java EE Servlet Code-Beispiel</title>");
53 out.println();
54 out.println("        <meta charset=\"utf-8\" />");
55 out.println();
56 out.println("        <meta name=\"robots\" content=\"noindex,nofollow\" />");
57 out.println("        <meta name=\"publisher\" content=\"Homepage-Webhilfe\" />");
58 out.println("    </head>");
59 out.println();
60 out.println("    <body style=\"line-height: 1.5em;>");
61 out.println("        <form method=\"post\">");
62 out.println("            Vorname: <input type=\"text\" name=\"vorname\" value=\"\" + sVorname + "\" /><br />");
63 out.println("            Nachname: <input type=\"text\" name=\"nachname\" value=\"\" + sNachname + "\" /><br />");
64 out.println("            <input type=\"submit\" value=\"Speichern\" />");
65 out.println("        </form>");
66 out.println("    </body>");
67 out.println("</html>");
68 out.close();
69 }
70 }

```



## Cookies

Mit Cookies ist es, anders als bei einer Session, möglich, **Daten über eine Sitzung hinaus zu speichern**. Durch eine solche Funktionalität ist es z. B. möglich, dass ein Benutzer über mehrere Browser-Sitzungen in einem Forum angemeldet bleiben kann. In einem Cookie kann jedoch, anders als in einer Session, nur ein Wert gespeichert werden. Sollen mehrere Werte gespeichert werden, so sind **mehrere Cookies** notwendig.

Die technische Realisierung von Cookies erfolgt über das HTTP-Protokoll mittels der Headerfelder `Cookie` und `Set-Cookie`. Die **Speicherung des Cookies selbst erfolgt clientseitig** (im Cache des Browsers). Wird ein Cookie vom Server „geschrieben“, so sendet der Webserver das Headerfeld `Set-Cookie`. Dies geschieht einmalig. Bei jeder Anfrage an den Server sendet der Client (Browser) das Headerfeld `Cookie` zurück. Dadurch kennt der Server den Wert des Cookies und kann diesen in der Anwendung verarbeiten.

Cookies werden in Java-Webanwendungen durch die Klasse `Cookie` (Package `javax.servlet.http`) repräsentiert. Eine Liste aller erhaltenen Cookies (diese wurden vom Client durch das Headerfeld `Cookie` an den Server gesendet) kann über die Methode `getCookies()` des `HttpServletRequest`-Interfaces abgerufen werden. Um der HTTP-Antwort ein Cookie anzufügen (dies wird vom Server durch das Headerfeld `Set-Cookie` an den Client gesendet), können Sie die Funktion `addCookie()` des `HttpServletResponse`-Interfaces nutzen. Dieser Funktion wird ein Objekt der `Cookie`-Klasse übergeben, welche nach Belieben erzeugt werden kann.

Dem Konstruktor der `Cookie`-Klasse werden zwei Zeichenketten übergeben: **Cookie-Name** und **Cookie-Wert**. Die folgende Tabelle zeigt eine Auflistung der wichtigsten Funktionen zum Festlegen und Abrufen von verschiedenen Eigenschaften:

<code>getName()</code>	Gibt den Namen des Cookies zurück oder legt diesen fest.
<code>setName()</code>	Gibt den Namen des Cookies zurück oder legt diesen fest.
<code>getValue()</code>	Gibt den Wert des Cookies zurück oder legt diesen fest.
<code>setValue()</code>	Gibt den Wert des Cookies zurück oder legt diesen fest.
<code>getMaxAge()</code>	Gibt die Lebensdauer des Cookies zurück oder legt diese fest.
<code>setMaxAge()</code>	Gibt die Lebensdauer des Cookies zurück oder legt diese fest.
<code>getDomain()</code>	Gibt die Domain, in welchem das Cookie gültig ist, zurück oder legt diese fest.
<code>setDomain()</code>	Gibt die Domain, in welchem das Cookie gültig ist, zurück oder legt diese fest.
<code>getPath()</code>	Gibt den Pfad, in welchem das Cookies gültig ist, zurück oder legt diesen fest.
<code>setPath()</code>	Gibt den Pfad, in welchem das Cookies gültig ist, zurück oder legt diesen fest.

Hierzu nun ein Beispiel, welches, wie das Beispiel zu Sessions, ein Feld für einen Vor- und Nachnamen besitzt. Diese Werte werden über zwei Cookies gespeichert. Diese Cookies besitzen dabei eine Lebensdauer von 1 Stunde.

```

1 package de.hwh.bsp.cookies;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 import javax.servlet.ServletException;
6 import javax.servlet.http.Cookie;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Session und Cookies](#)

```

11 public class CookieServlet extends HttpServlet
12 {
13     @Override
14     protected void service(HttpServletRequest request, HttpServletResponse response)
15         throws ServletException, IOException
16     {
17         Cookie oVornameCookie = null, oNachnameCookie = null;
18         String sVorname, sNachname;
19
20         // Zeichenkodierung auf UTF-8 setzen (für Formulare Daten)
21         request.setCharacterEncoding("UTF-8");
22
23         // Cookies abrufen
24         if (request.getCookies() != null)
25         {
26             for (Cookie oCookie : request.getCookies())
27             {
28                 if (oCookie.getName().equals("vorname"))
29                     oVornameCookie = oCookie;
30                 else if (oCookie.getName().equals("nachname"))
31                     oNachnameCookie = oCookie;
32             }
33         }
34
35         // Prüfen ob die Anfrage mittels der POST-Methode durchgeführt wurde
36         if (request.getMethod() == "POST")
37         {
38             // Werte aus dem Formular abrufen
39             sVorname = request.getParameter("vorname") != null ? request.getParameter("vorname") : "";
40             sNachname = request.getParameter("nachname") != null ? request.getParameter("nachname") : "";
41
42             // Cookies speichern, Werte darin speichern und diese der Antwort hinzufügen
43             oVornameCookie = new Cookie("vorname", sVorname);
44             oVornameCookie.setMaxAge(3600);
45             response.addCookie(oVornameCookie);
46             oNachnameCookie = new Cookie("nachname", sNachname);
47             oNachnameCookie.setMaxAge(3600);
48             response.addCookie(oNachnameCookie);
49         }
50         else
51         {
52             // Werte aus den Cookies abrufen
53             sVorname = oVornameCookie != null ? oVornameCookie.getValue() : "";
54             sNachname = oNachnameCookie != null ? oNachnameCookie.getValue() : "";
55         }
56
57         // Ausgabe als HTML-Seite
58         response.setContentType("text/html;charset=UTF-8");
59
60         // Ausgabe durchführen
61         PrintWriter out = response.getWriter();
62         out.println("<!DOCTYPE html>");
63         out.println("<html>");
64         out.println("    <head>");
65         out.println("        <title>Cookies - Java EE Servlet Code-Beispiel</title>");
66         out.println("    ");
67         out.println("        <meta charset=\"utf-8\" />");
68         out.println("    ");
69         out.println("        <meta name=\"robots\" content=\"noindex,nofollow\" />");
70         out.println("        <meta name=\"publisher\" content=\"Homepage-Webhilfe\" />");
71         out.println("    </head>");
72         out.println("    <body style=\"line-height: 1.5em;\">");
73         out.println("        <form method=\"post\">");
74         out.println("            Vorname: <input type=\"text\" name=\"vorname\" value=\"\" + sVorname + \"\" /><br />");
75         out.println("            Nachname: <input type=\"text\" name=\"nachname\" value=\"\" + sNachname + \"\" /><br />");
76         out.println("            <input type=\"submit\" value=\"Speichern\" />");
77         out.println("        </form>");
78         out.println("    </body>");
79         out.println("</html>");
80         out.close();
81     }
82 }
83 }

```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Filter](#)

## Filter

Filter sind eine spezielle Technologie für Java-Webanwendungen, welche es erlauben, **vor oder nach dem Ausführen des eigentlichen Servlets, bestimmte Anweisungen auszuführen** bzw. das Ausführen zu unterbinden. Typische Beispiele zur Verwendung von Filtern sind Authentifizierung (z. B. für einen geschützten Bereich auf der Website), Protokollierung (z. B. das Protokollieren von Zugriffen auf bestimmte Ressourcen), Komprimierung und Verschlüsselung.

Filter sind ganz normale Java-Klassen, die das Interface `Filter` (Package `javax.servlet`) implementieren. Das `Filter`-Interface bietet 3 Funktionen, welche überschrieben werden können: `init()`, `doFilter()` und `destroy()`. Die Funktionen `init()` und `destroy()` werden auch hier üblicherweise zum **Allokieren und Freigeben von Ressourcen** verwendet. Der `init()`-Methode wird ein Objekt des Interfaces `FilterConfig` übergeben, welches die **Konfiguration des Filters** enthält. Eine solche Konfiguration kann über die `web.xml`-Datei festgelegt werden. Ein typisches Beispiel für die Verwendung einer solchen Konfiguration wäre ein Wartungs-Flag: Ist das Flag gesetzt, so wird auf eine Wartungsseite umgeleitet, andernfalls wird die angefragte Ressource abgerufen. Die Methode `doFilter()` ist die Funktion, welche vom Servlet-Container aufgerufen wird. Als Übergabeparameter erhalten Sie ein Objekt des `ServletRequest`- und `ServletResponse`-Interfaces sowie des `FilterChain`-Interfaces. Um das eigentliche Servlet bzw. die weiteren Filter (wenn es mehrere Filter gibt, die auf die angefragte URL zutreffen) aufzurufen, wird die Methode `doFilter()` des `FilterChain`-Objekts aufgerufen.



Filter müssen im **Deployment Descriptor** (Datei `web.xml`) konfiguriert werden. Dafür müssen wir als erstes den Filter „bekanntmachen“ und diesem einen lokalen Namen geben. Hier kommen, ähnlich wie bei Servlets, die XML-Elemente `filter`, `filter-name` und `filter-class` zum Einsatz.

```
1 <filter>
2   <filter-name>LogFilter</filter-name>
3   <filter-class>de.hwh.bsp.filter.LogFilter</filter-class>
4 </filter>
```

Als nächstes müssen wir dem Filter ein **URL-Muster** zuordnen. Dieses wird im XML-Element `url-pattern` angegeben. So wie bei Servlets auch, kann hier ein Wildcard-Zeichen `*` verwendet werden (siehe Beispiel).

```
1 <filter-mapping>
2   <filter-name>LogFilter</filter-name>
3   <url-pattern>*/</url-pattern>
4 </filter-mapping>
```

Beim Erstellen eines Filters in NetBeans werden die lokalen Methoden `doBeforeProcessing()` und `doAfterProcessing()` in der Java-Klasse angelegt, wovon die eine vor dem und die andere nach dem das eigentliche Servlet ausgeführt wird, aufgerufen wird. Der Aufruf der Funktion `doFilter()`, welche die **Filter-Kette** (engl. *filter chain*) aufruft, steht dabei zwischen dem Aufruf von `doBeforeProcessing()` und `doAfterProcessing()`. Der Assistent zum Anlegen eines Filters in NetBeans kann auch die Konfiguration im Deployment Descriptor für Sie übernehmen.

Hier nun ein Beispielprojekt, bei welchem die Remote-IP-Adresse und der Inhaltstyp in die Konsole geloggt werden.

**Filter (LogFilter.java):**

```
1 package de.hwh.bsp.filter;
2
3 import java.io.IOException;
4 import java.util.Date;
5 import javax.servlet.Filter;
6 import javax.servlet.FilterChain;
7 import javax.servlet.FilterConfig;
8 import javax.servlet.ServletException;
9 import javax.servlet.ServletRequest;
10 import javax.servlet.ServletResponse;
11
12 public class LogFilter implements Filter
13 {
14     private void doBeforeProcessing(ServletRequest request, ServletResponse response)
15     {
16         System.out.println((new Date()).toString() + ":");
17         System.out.println(" Zugriff von " + request.getRemoteAddr());
18     }
19
20     private void doAfterProcessing(ServletRequest request, ServletResponse response)
21     {
22         System.out.println(" Rückgabe als " + response.getContentType());
23     }
24
25     @Override
26     public void init(FilterConfig filterConfig)
27     {
28         // Ressourcen allokkieren und Initialisierung ausführen
29     }
30
31     @Override
32     public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
33     {
34         // Filter-Funktion bevor die Anfrage bearbeitet wurde ausführen
35         doBeforeProcessing(request, response);
36
37         // Anfrage versuchen zu bearbeiten
38         try
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Filter](#)

```

39         {
40             chain.doFilter(request, response);
41         }
42         catch (IOException eIO)
43         {
44             System.out.println(eIO.toString());
45         }
46         catch (ServletException eIO)
47         {
48             System.out.println(eIO.toString());
49         }
50     }
51     // Filter-Funktion nachdem die Anfrage bearbeitet wurde ausführen
52     doAfterProcessing(request, response);
53 }
54
55 @Override
56 public void destroy()
57 {
58     // Ressourcen freigeben
59 }
60 }

```

## Servlet (LogServlet.java):

```

1 package de.hwh.bsp.filter;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 import javax.servlet.ServletException;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10 public class LogServlet extends HttpServlet
11 {
12     @Override
13     protected void service(HttpServletRequest request, HttpServletResponse response)
14         throws ServletException, IOException
15     {
16         // Ausgabe als Text-Datei
17         response.setContentType("text/plain;charset=UTF-8");
18
19         // Ausgabe durchführen
20         PrintWriter out = response.getWriter();
21         out.println("Ihre IP-Adresse ist: " + request.getRemoteAddr());
22         out.println("Die Antwort wird gesendet als: " + response.getContentType());
23         out.close();
24     }
25 }

```

## Konfiguration (web.xml):

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
4     <filter>
5         <filter-name>LogFilter</filter-name>
6         <filter-class>de.hwh.bsp.filter.LogFilter</filter-class>
7     </filter>
8     <filter-mapping>
9         <filter-name>LogFilter</filter-name>
10        <url-pattern>*/</url-pattern>
11    </filter-mapping>
12    <servlet>
13        <servlet-name>LogServlet</servlet-name>
14        <servlet-class>de.hwh.bsp.filter.LogServlet</servlet-class>
15    </servlet>
16    <servlet-mapping>
17        <servlet-name>LogServlet</servlet-name>
18        <url-pattern></url-pattern>
19    </servlet-mapping>
20    <session-config>
21        <session-timeout>
22            30
23        </session-timeout>
24    </session-config>
25 </web-app>

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

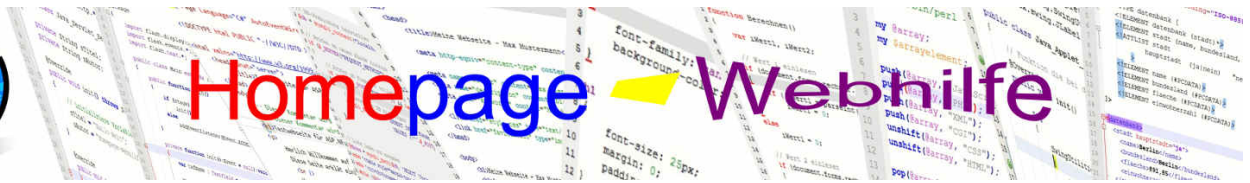
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » Fehlerbehandlung

## Fehlerbehandlung

### Inhalt dieser Seite:

1. HTTP-Fehler
2. Ausnahmefehler

In diesem Thema wollen wir uns damit beschäftigen, wie Sie in Java-Webanwendungen **HTTP- und Ausnahme-Fehler abfangen** können, um somit z. B. Ihren Besuchern beim Auftreten eines Fehlers, eine **eigene Fehlerseite anzeigen** zu können (und nicht die Standardseiten von Ihrem Webserver). Natürlich können Sie diese Möglichkeit auch dazu verwenden, **Fehler zu protokollieren**.

### HTTP-Fehler

Im Protokoll HTTP (und auch bei HTTPS) sind sogenannte **Statuscodes** implementiert. Bei jeder HTTP-Antwort sendet der Server an den Client einen solchen Statuscode mit. In den meisten Fällen wird der Code **200 OK** zurückgeschickt, welcher signalisiert, dass die angefragte Ressource gefunden wurde und der Inhalt zurückgeschickt wird. Die Liste der verfügbaren Statuscodes ist lang, weshalb wir uns hier vorerst mal nur mit den Statuscodes **403 Forbidden** und **404 Not Found** auseinandersetzen wollen. Der Code **403** weist darauf hin, dass der Zugriff auf die Ressource auf Grund einer **fehlenden Berechtigung** nicht möglich ist. Einen solchen Fehlercode könnten Sie bspw. in einem Filter zurückgeben, wenn ein Zugriff auf den Admin-Bereich vorgenommen wird, jedoch nicht der Administrator angemeldet ist. Den Statuscode **404** hat vermutlich schon jeder gesehen und besagt einfach nur, dass die angefragte **Ressource nicht (mehr) verfügbar** ist. Im Internet sind u. a. oftmals Links zu finden, welche auf eine Seite verweisen, welche nicht mehr existiert. So ein Link wird dann auch als toter Link bezeichnet.

Zurück zu den Java-Webanwendungen: Wenn Sie beim Auftreten eines bestimmten Statuscodes auf eine bestimmte Seite (dabei kann es sich auch um ein Servlet handeln) verweisen wollen, dann müssen Sie lediglich eine **Konfiguration im Deployment Descriptor** durchführen. Dafür benötigen Sie das XML-Element `error-page`, welchem Sie die XML-Elemente `error-code` und `location` unterordnen. Dies sieht dann z. B. so aus:

```
1 <error-page>
2   <error-code>404</error-code>
3   <location>/ErrorServlet</location>
4 </error-page>
```

Innerhalb eines Servlets können Sie mit der Funktion `getAttribute()` des `HttpServletRequest`-Objekts den „fehlerhaften“ **Statuscode** abrufen. Als Attribut-Name müssen Sie `javax.servlet.error.status_code` übergeben.

Beim Aufrufen des folgenden Beispiels werden Sie als erstes auf das Servlet `ErrorServlet` umgeleitet, da es kein Servlet für das Wurzelverzeichnis gibt. Es handelt sich daher also um einen **404-Fehler**. Beim Versuch des Aufrufs von `AdminServlet` werden Sie ebenfalls wieder auf `ErrorServlet` umgeleitet. Dieses Mal wird Ihnen jedoch der Fehlercode **403** angezeigt.

#### Servlet (ErrorServlet.java):

```
1 package de.hwh.bsp.httpfehler;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 import javax.servlet.ServletException;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10 public class ErrorServlet extends HttpServlet
11 {
12     @Override
13     protected void service(HttpServletRequest request, HttpServletResponse response)
14         throws ServletException, IOException
15     {
16         // Ausgabe als Text-Datei
17         response.setContentType("text/plain;charset=UTF-8");
18
19         // Ausgabe durchführen
20         PrintWriter out = response.getWriter();
21         out.println("HTTP-Fehler-Code: " + request.getAttribute("javax.servlet.error.status_code"));
22         out.close();
23     }
24 }
```

#### Servlet (AdminServlet.java):

```
1 package de.hwh.bsp.httpfehler;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8
9 public class AdminServlet extends HttpServlet
10 {
11     @Override
12     protected void service(HttpServletRequest request, HttpServletResponse response)
13         throws ServletException, IOException
14     {
15         response.sendError(403);
16     }
17 }
```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » [Fehlerbehandlung](#)

## Konfiguration (web.xml):

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
4      <servlet>
5          <servlet-name>ErrorServlet</servlet-name>
6          <servlet-class>de.hwh.bsp.httpfehler.ErrorServlet</servlet-class>
7      </servlet>
8      <servlet>
9          <servlet-name>AdminServlet</servlet-name>
10         <servlet-class>de.hwh.bsp.httpfehler.AdminServlet</servlet-class>
11     </servlet>
12     <servlet-mapping>
13         <servlet-name>ErrorServlet</servlet-name>
14         <url-pattern>/ErrorServlet</url-pattern>
15     </servlet-mapping>
16     <error-page>
17         <error-code>403</error-code>
18         <location>/ErrorServlet</location>
19     </error-page>
20     <error-page>
21         <error-code>404</error-code>
22         <location>/ErrorServlet</location>
23     </error-page>
24     <servlet-mapping>
25         <servlet-name>AdminServlet</servlet-name>
26         <url-pattern>/AdminServlet</url-pattern>
27     </servlet-mapping>
28     <session-config>
29         <session-timeout>
30             30
31         </session-timeout>
32     </session-config>
33 </web-app>

```



Übrigens: Ein weiterer gängiger Statuscode ist `500 Internal Server Error`, welcher in den meisten Fällen auf Grund einer fehlerhaften Konfiguration des Webservers erscheint.

## Ausnahmefehler

In Java sollten Sie, so wie in anderen Programmiersprachen auch, versuchen, Fehler (vor allem Ausnahmefehler) im Programm **so gut wie möglich abzufangen**. Trotzdem kommt es immer wieder vor, dass Sie vergessen, einen Ausnahmefehler (engl. *exception*) abzufangen. Dies ist auch nicht weiter schlimm, da **vom Servlet-Container alle Ausnahmen abgefangen werden**. Da in der Regel in so einem Fall jedoch eine Art von Debug-Informationen angezeigt werden, welche die Besucher nicht sehen sollten, sollten diese Art von Fehlern ebenfalls abgefangen werden. Auch hier kann dazu der Deployment Descriptor verwendet werden.

Im **Deployment Descriptor** wird auch hier das XML-Element `error-page` verwendet. Diesem wird nun das `exception-type`- und das bereits bekannte `location`-Element untergeordnet.

```

1  <error-page>
2      <exception-type>java.io.IOException</exception-type>
3      <location>/ErrorServlet</location>
4  </error-page>

```

Im „Fehler-Servlet“ können wir mit der Funktion `getAttribute()` und dem Attributnamen `javax.servlet.error.exception_type` den **Typ der Ausnahme** abfragen. Mit dem Attribut `javax.servlet.error.message` kann die übergebene **Meldung** abgefragt werden.

Im folgenden Beispiel wird auf der Startseite (`StartServlet.java`) manuell (und lediglich zu Demonstrationszwecken) eine IO-Ausnahme (`IOException`) geworfen. Dadurch leitet uns der Servlet-Container auf das Servlet `ErrorServlet` um, in welchem die Informationen zur Ausnahme angezeigt werden.

## Servlet (ErrorServlet.java):

```

1  package de.hwh.bsp.ausnahmefehler;
2
3  import java.io.IOException;
4  import java.io.PrintWriter;
5  import javax.servlet.ServletException;
6  import javax.servlet.http.HttpServlet;
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9
10 public class ErrorServlet extends HttpServlet
11 {
12     @Override
13     protected void service(HttpServletRequest request, HttpServletResponse response)
14         throws ServletException, IOException
15     {
16         // Ausgabe als Text-Datei
17         response.setContentType("text/plain;charset=UTF-8");

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Servlet](#) » Fehlerbehandlung

```

18
19 // Ausgabe durchführen
20 PrintWriter out = response.getWriter();
21 out.println("Exception-Typ: " + request.getAttribute("javax.servlet.error.exception_type"));
22 out.println("Meldung: " + request.getAttribute("javax.servlet.error.message"));
23 out.close();
24 }
25 }

```

Servlet (StartServlet.java):

```

1 package de.hwh.bsp.ausnahmefehler;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8
9 public class StartServlet extends HttpServlet
10 {
11     @Override
12     protected void service(HttpServletRequest request, HttpServletResponse response)
13         throws ServletException, IOException
14     {
15         throw new IOException("Fehler beim Zugriff auf die Ressource");
16     }
17 }

```

Konfiguration (web.xml):

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
4   <servlet>
5     <servlet-name>ErrorServlet</servlet-name>
6     <servlet-class>de.hwh.bsp.ausnahmefehler.ErrorServlet</servlet-class>
7   </servlet>
8   <servlet>
9     <servlet-name>StartServlet</servlet-name>
10    <servlet-class>de.hwh.bsp.ausnahmefehler.StartServlet</servlet-class>
11  </servlet>
12  <servlet-mapping>
13    <servlet-name>ErrorServlet</servlet-name>
14    <url-pattern>/ErrorServlet</url-pattern>
15  </servlet-mapping>
16  <servlet-mapping>
17    <servlet-name>StartServlet</servlet-name>
18    <url-pattern>/</url-pattern>
19  </servlet-mapping>
20  <error-page>
21    <exception-type>java.io.IOException</exception-type>
22    <location>/ErrorServlet</location>
23  </error-page>
24  <session-config>
25    <session-timeout>
26      30
27    </session-timeout>
28  </session-config>
</web-app>

```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

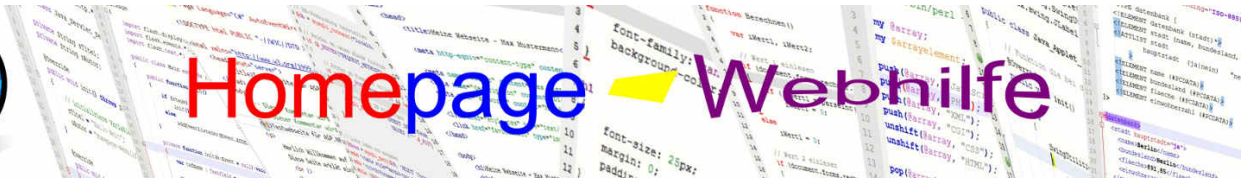
## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [JavaServer Pages](#) » [Grundlagen](#)

## Grundlagen

Die JavaServer Pages Technologie (kurz JSP) ermöglicht uns das **einfache Generieren einer dynamischen HTML-Seite** (oder auch XML-Seite). Hierbei wird **HTML- und Java-Code direkt gemischt** und ist somit mit PHP vergleichbar.

JavaServer Pages haben in der Regel die Dateiendung `.jsp` und enthalten ganz normale HTML-Code sowie einige **Code-Blöcke**, welche Java-Code enthalten. Durch die Code-Blöcke können **dynamische Ausgaben**, Abfragen und vieles mehr durchgeführt werden. Das hieraus resultierende Konzept ermöglicht also das Mischen von HTML- und Java-Code. Dies ist auf Grund des MVC-Konzepts nicht immer erwünscht, stellt jedoch eine leichte Methode zur Erstellung einer Seite dar.

Innerhalb einer JavaServer Page können prinzipiell alle Klassen der Java-Packages verwendet werden. Zudem können über sogenannte Standardvariablen Objekte verschiedener Servlet-Klassen und -Interfaces abgerufen werden.

Die Technologie JavaServer Pages wurde bereits Ende der 90er-Jahre eingesetzt und gehört somit zu den älteren Technologien für serverseitige Programmierung. Auf Grund der immer größer werdenden **Ablösung durch andere Technologien** und der Verwendung von Frameworks (wie z. B. JavaServer Faces) wird JSP immer seltener eingesetzt und gilt seit Java EE Version 6 als veraltet (engl. *deprecated*). Daher ist es **nicht zu empfehlen, JSP für neue Projekte zu verwenden**. Wir bieten auf dieser Webseite jedoch auch Tutorials zu vielen anderen serverseitigen Technologien an.

## Erstes Beispiel

Um eine bessere Vorstellung von JSP zu bekommen, erstellen wir als erstes ein einfaches JSP-Beispiel. Wir haben, wie auch im Servlet-Tutorial, die **Entwicklungsumgebung NetBeans** sowie den **Apache Tomcat Webserver**, der Teil des XAMPP-Softwarepakets ist, verwendet, um die Beispiele dieses Tutorials zu erstellen. Falls Sie bisher noch keine IDE und keinen Webserver installiert haben, würden wir Ihnen empfehlen, NetBeans zu verwenden, da Sie dadurch den nachfolgenden Schritten genauer folgen können.

Nach dem Start der Entwicklungsumgebung NetBeans müssen wir ein neues Projekt anlegen. Im Dialog zur Projektstellung wählen wir die Kategorie „Java Web“ und den Projekttyp „Web Application“. Als nächstes müssen wir noch den Projektnamen und -speicherort sowie anschließend den Server (in diesem Fall den Apache Tomcat) und die Version (i. d. R. Java EE 6) auswählen. Bei der Einbindung von Frameworks müssen wir nichts auswählen.

Das nun erstellte Projekt enthält bereits eine JSP-Datei: `index.jsp`. Diese Datei wurde automatisch generiert und liegt im Wurzelverzeichnis des Web-Projekts. Die generierte JavaServer Page enthält zwar bereits ein HTML-Grundgerüst und eine `page`-Direktive (dazu mehr im [Thema Direktiven](#)), jedoch noch keinen Code-Block. Nun können Sie die Seite anpassen und folgenden Java-Code einfügen: `<%= (new java.util.Date()).toString() %>`. Wenn Sie das Projekt nun ausführen, werden Sie feststellen, dass auf der Seite das aktuelle Datum und die aktuelle Uhrzeit angezeigt wird. Wenn Sie die Seite im Browser erneut laden, aktualisiert sich auch das Datum bzw. die Uhrzeit. Sie haben hiermit nun Ihre **erste dynamische Webseite** mittels JSP erstellt.

```

1  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2
3  <!DOCTYPE html>
4  <html>
5      <head>
6          <title>Erste Seite - JavaServer Pages Code-Beispiel</title>
7
8          <meta charset="utf-8" />
9
10         <meta name="robots" content="noindex,nofollow" />
11         <meta name="publisher" content="Homepage-Webhilfe" />
12     </head>
13
14     <body>
15         Serverzeit: <%= (new java.util.Date()).toString() %>
16     </body>
17 </html>
    
```



## Syntax

Prinzipiell enthalten JavaServer Pages **ganz normalen HTML- oder XML-Code**. Um dort nun **dynamische Inhalte** sowie Java-Code zu platzieren, werden spezielle Tags oder XML-Elemente verwendet. Hierbei kann eine grobe Unterteilung zwischen Skriptelementen (oft auch als Skripttag oder Code-Block bezeichnet), Direktiven und Aktionen getroffen werden. Auf die [Direktiven](#) und [Aktionen](#) gehen wir in den nächsten Themen genauer ein.

In Java gibt es drei unterschiedliche **Skriptelemente**, welche sich im öffnenden Tag unterscheiden. Der schließende Tag der Elemente ist immer `>`. Das einfachste Skriptelement beginnt mit dem Skripttag `<%`. Zwischen `<%` und `>` kann Java-Code angegeben werden. Dieser Code wird an der Stelle ausgeführt, wo Sie ihn platzieren. Dieser Syntax wird oftmals auch als **Skriptlet** bezeichnet. Hierzu ein kurzes Beispiel:

```

1  <%
2      for (int i = 0; i < 10; i++)
3          out.println(i + "<br />");
4  %>
    
```

Wenn Sie **globale Variablen oder Methoden definieren** möchten, so müssen Sie den Starttag durch `<%!` ersetzen. Dies könnte wie folgt aussehen:

```

1  <%!
2      private double berechneKreisUmfang(double dRadius)
3      {
4          return 2 * Math.PI * dRadius;
5      }
6  %>
    
```

Um einen Wert (Konstante, Variable oder Rückgabewert einer Funktion) auszugeben, könnten wir die Funktion `out.print()` aufrufen und dieser den

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

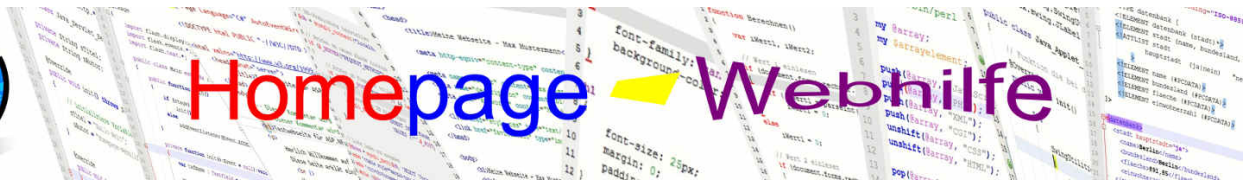
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [JavaServer Pages](#) » [Grundlagen](#)

**auszugebenen Wert** übergeben. Zu bevorzugen ist jedoch der Skripttag `<%=`. Letztendlich wird hier ebenfalls die `print()`-Funktion aufgerufen, jedoch ist dadurch unser Code kürzer und leichter zu lesen.

```
1 | <%= (new java.util.Date()).toString() %>
```

Wenn Sie innerhalb Ihrer JSP einen **Kommentar** platzieren möchten, so können Sie die Tags `<!--` und `-->` verwenden. Der Unterschied des JSP-Kommentars im Vergleich zu einem einfachen HTML-Code ist, dass der JSP-Kommentar nur serverseitig existiert und somit nicht an den Browser geschickt wird. Je nach Verwendungszweck ist daher entweder der JSP- oder der HTML-Kommentar zu bevorzugen.

```
1 | <!-- Dies ist serverseitiger Kommentar! -->
```

## Standardvariablen

Wie bereits weiter oben erwähnt können wir innerhalb einer JSP auf Standardvariablen in Form von Objekten zugreifen, die es uns ermöglichen, Informationen abzufragen oder die „Antwort“ anzupassen. Einige dieser Objekte kommen Ihnen vielleicht vom [Servlet-Tutorial](#) bekannt vor:

Variable	Klasse / Interface	Beschreibung
request	HttpServletRequest	Objekt zum Abrufen von Anfrage-Informationen
response	HttpServletResponse	Objekt zum Anpassen der Antwort
out	JspWriter	Objekt für den Ausgabe-Stream
session	HttpSession	Objekt der HTTP-Session
page	Object	Objekt der Seite selbst
pageContext	PageContext	Objekt für erweiterte Zugriffe auf die Seite bzw. dessen Daten
exception	Throwable	Objekt für Ausnahmefehler-Informationen (für Fehlerseiten)
application	ServletContext	Objekt der Webserver-Anwendung
config	ServletConfig	Objekt zur Konfiguration

## Funktionsweise

Die Funktionsweise von JavaServer Pages lässt sich am besten mit einer Schritt-Für-Schritt-Folge erklären:

1. Als erstes schickt der Webbrowser des Besuchers eine Anfrage an den Webserver.
2. Der Webserver empfängt die Anfrage (engl. *request*) und speichert die Anfrage-Informationen.
3. Nun stellt der Webserver fest, dass es sich bei der angefragten Ressource um eine JavaServer Page handelt (dies wird mittels der Dateiendung festgestellt).
4. Da es sich um eine JavaServer Page handelt, muss die Seite durch einen **JSP-Compiler** übersetzt werden. Dabei wird die JSP lediglich in ein ganz normales HTTP-Servlet umgewandelt.
5. Im nächsten Schritt wird das HTTP-Servlet durch den **Java-Compiler** übersetzt und somit in eine `.class`-Datei umgewandelt. Diese Datei wird für eine evtl. erneute Nutzung gespeichert (dazu gleich mehr).
6. Der generierte Java-Bytecode, welcher das Servlet repräsentiert, wird nun vom Servlet-Container ausgeführt.
7. Durch das Servlet, welches vom Servlet-Container ausgeführt wurde, wird als nächstes eine HTTP-Antwort (engl. *response*) generiert. Diese enthält i. d. R. einen HTTP-Header sowie eine HTML-Seite, welche für den Browser statisch erscheint.
8. Die erzeugte Antwort wird zurück an den Browser geschickt.

Der Java-Bytecode des vom JSP-Compiler erzeugten Servlets wird gespeichert. Dadurch muss es bei einer erneuten Anfrage **nicht erneut kompiliert werden**, sondern kann direkt ausgeführt werden. Um zu erkennen, ob sich die JSP-Datei in der Zwischenzeit geändert hat, wird das **Änderungsdatum der Datei geprüft**. Wurde eine Änderung erkannt, so wird wieder eine Kompilierung ausgeführt.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [JavaServer Pages](#) » [Direktiven](#)

## Direktiven

Direktiven erlauben es, **Informationen der Seite festzulegen** bzw. diese zu konfigurieren. Eine Seite darf dabei mehrere Direktiven enthalten. Eine Direktive beginnt immer mit dem XML-Tag `<%@` und endet mit dem XML-Tag `%>`. Eine Direktivendeklaration setzt sich aus den genannten XML-Tags, einem Schlüsselwort und Attributen zusammen. **Attribute** werden wie in XML angegeben: `name="wert"`.

### Inhalt dieser Seite:

1. Seitenkonfiguration
2. Einbindung externer Dateien

## Seitenkonfiguration

Direktiven zur Konfiguration der Seite beginnen mit dem Schlüsselwort `page`. `page`-Direktiven werden vor dem eigentlichen Inhalt der Seite angegeben. Bei diesen Direktiven können nun verschiedene Attribute angegeben werden. Hier sehen Sie eine Übersicht der wichtigsten Attribute:

<b>contentType</b>	Legt den Inhaltstyp der Seite (im Regelfall text/html) fest.
<b>pageEncoding</b>	Legt die Zeichenkodierung der Seite fest.
<b>import</b>	Importiert ein oder mehrere Java-Package(s).
<b>errorPage</b>	Legt die Seite für Fehlerfälle fest.
<b>isErrorPage</b>	Gibt an, ob es sich um eine Fehlerseite handelt.
<b>isThreadSafe</b>	Gibt an, ob die Seite threadsicher ist.
<b>info</b>	Legt die Beschreibung der Seite fest (kann per <code>getServletInfo()</code> abgerufen werden).

Hierzu nun folgendes Beispiel:

```

1  <%@page import="java.util.Date"%>
2  <%@page contentType="text/html" pageEncoding="UTF-8"%>
3
4  <!DOCTYPE html>
5  <html>
6    <head>
7      <title>Seitenkonfiguration - JavaServer Pages Code-Beispiel</title>
8
9      <meta charset="utf-8" />
10
11     <meta name="robots" content="noindex,nofollow" />
12     <meta name="publisher" content="Homepage-Webhilfe" />
13   </head>
14
15   <body>
16     Serverzeit: <%= (new Date()).toString() %>
17   </body>
18 </html>

```



## Einbindung externer Dateien

Die `include`-Direktive erlaubt es, den **vollständigen Inhalt einer Datei in die JavaServer Page einzubetten**. Hierbei ist jedoch zu beachten, dass dieser Vorgang bei der Kompilierung durch den JSP-Compiler erfolgt, d. h. der **Inhalt wird statisch eingebettet**. Wird die eingebettete Datei geändert, so wird die Datei erst dann neu eingebunden, wenn auch die JSP-Datei neu kompiliert wird (z. B. durch die Änderung der JSP-Datei). Die URL der einzubettenden Datei wird über das Attribut `file` spezifiziert.

**JSP-Dokument (index.jsp):**

```

1  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2
3  <!DOCTYPE html>
4  <html>
5    <head>
6      <title>Einbindung externer Dateien - JavaServer Pages Code-Beispiel</title>
7
8      <meta charset="utf-8" />
9
10     <meta name="robots" content="noindex,nofollow" />
11     <meta name="publisher" content="Homepage-Webhilfe" />
12   </head>
13
14   <body>
15     <%@include file="startseite.html" %>
16   </body>
17 </html>

```

**HTML-Dokument (startseite.html):**

```

1 | <h1>Beispielseite</h1>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [JavaServer Pages](#) » [Direktiven](#)

2 | [<p>...</p>](#)



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [JavaServer Pages](#) » [Aktionen](#)

## Aktionen

Aktionen werden als XML-Elemente angegeben und ermöglichen es auf einfache Art und Weise, **bestimmte Webserververfunktionen auszuführen**. Die Namen der Elemente einer Aktion beginnen immer mit dem Präfix `jsp` gefolgt von dem eigentlichen Aktionsnamen. Verschiedene Eigenschaften der Aktion werden über Attribute festgelegt. Die Aktionselemente sind in der Regel einteilig bzw. leer.

### Inhalt dieser Seite:

1. Einbindung anderer Seiten
2. Weiterleitung

## Einbindung anderer Seiten

Die `include`-Aktion erlaubt es den Inhalt einer anderen Seite (dies kann eine HTML-Seite, eine JSP oder ein anderes Dokument sein) **dynamisch einzubetten**. Anders als bei der `include`-Direktive wird also bei der `include`-Aktion der Inhalt der **referenzierten Seite bei jedem Aufruf der Seite neu eingebunden**. Die URL der Seite wird über das `page`-Attribut angegeben.

### JSP-Dokument (index.jsp):

```

1 <%@page contentType="text/html" pageEncoding="UTF-8"%>
2
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <title>Einbindung anderer Seiten - JavaServer Pages Code-Beispiel</title>
7
8     <meta charset="utf-8" />
9
10    <meta name="robots" content="noindex,nofollow" />
11    <meta name="publisher" content="Homepage-Webhilfe" />
12  </head>
13
14  <body>
15    <jsp:include page="time.jsp" />
16  </body>
17 </html>

```

### JSP-Dokument (time.jsp):

```

1 <%@page contentType="text/html" pageEncoding="UTF-8"%>
2
3 Serverzeit: <%= (new java.util.Date()).toString() %>

```



## Weiterleitung

Mit der Aktion `forward` ist es möglich, eine **Weiterleitung auf eine andere Seite durchzuführen**. Der Pfad, zu welcher Seite weitergeleitet werden soll, wird im `page`-Attribut angegeben.

### JSP-Dokument (index.jsp):

```

1 <%@page contentType="text/html" pageEncoding="UTF-8"%>
2
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <title>Weiterleitung - JavaServer Pages Code-Beispiel</title>
7
8     <meta charset="utf-8" />
9
10    <meta name="robots" content="noindex,nofollow" />
11    <meta name="publisher" content="Homepage-Webhilfe" />
12  </head>
13
14  <body>
15    <jsp:forward page="time.jsp" />
16  </body>
17 </html>

```

### JSP-Dokument (time.jsp):

```

1 <%@page contentType="text/html" pageEncoding="UTF-8"%>
2
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <title>Weiterleitung - JavaServer Pages Code-Beispiel</title>
7
8     <meta charset="utf-8" />
9
10    <meta name="robots" content="noindex,nofollow" />
11    <meta name="publisher" content="Homepage-Webhilfe" />
12  </head>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

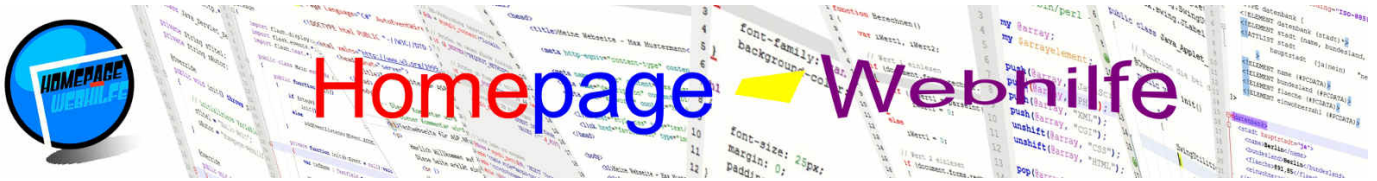
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [JavaServer Pages](#) » [Aktionen](#)

```
13 |
14 |     <body>
15 |         Serverzeit: <%= (new java.util.Date()).toString() %>
16 |     </body>
17 | </html>
```



#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

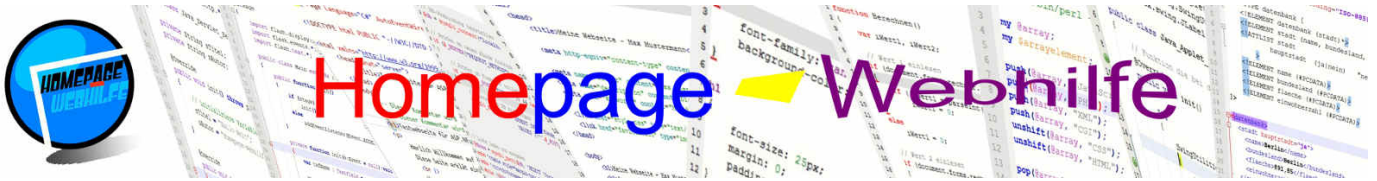
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Java EE](#) » [Abschluss](#)

## Abschluss

Nachdem Sie nun die Möglichkeiten und Technologien von Java EE in Bezug auf Webanwendungen kennengelernt haben, können Sie nun selbst **entscheiden, ob Sie Java EE produktiv einsetzen möchten** oder nicht. Dabei sind natürlich Geschmack und technische Möglichkeiten oder Notwendigkeiten von großer Bewandnis.

Wir haben in den beiden Unterkapiteln natürlich wieder versucht, die meisten bzw. wichtigsten Funktionen, Klassen und Interfaces zu erläutern, jedoch behandeln wir auf Grund der Komplexität und Übersichtlichkeit halber **nur die wichtigsten Komponenten**. Die vollständige [Referenz für die Java EE API](#) finden Sie auf der offiziellen Seite von Oracle.

Für die serverseitige Programmierung wird Java EE nur selten eingesetzt. Häufiger kommen [PHP](#) oder aber auch [ASP.NET](#) zum Einsatz. Für diese Technologien bieten wir hier ebenfalls Tutorials an.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislungen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

Java EE

XML

# E-Book

## XML & Co.





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [Einführung](#)

## Einführung



Zu XML gehört eine **Vielzahl an unterschiedlichen Sprachen**. Dabei wird in der Regel von der **XML-Familie** gesprochen. Zur XML-Familie gehören neben der Grundsprache XML selbst, auch Bestandteile der Sprache XML (z. B. DTD), sowie Sprachen, die auf der Grundsprache XML aufbauen (z. B. XSLT).

Dieses Thema teilt sich in mehrere Kapitel auf, in welchen Sprachen der XML-Familie sowie die Grundsprache XML behandelt werden. Auf dieser Seite wollen wir Ihnen jedoch schon mal einen **groben Überblick** über die XML-Familie geben.

Die Spezifikation der Grundsprache XML sowie der meisten XML basierenden Sprachen wurden vom W3C (World Wide Web Consortium) veröffentlicht. Die erste Version von XML erschien 1998. XML wurde **von der Sprache SGML erweitert**. Deshalb haben XML und SGML sehr viele Gemeinsamkeiten, weshalb bei HTML oft von einer XML basierenden Sprache gesprochen wird. Dies trifft jedoch nur auf XHTML zu. Alle anderen HTML-Versionen basieren auf SGML.

### Inhalt dieser Seite:

1. Grundsprache
2. Adressierung
3. Transformation
4. Definition
5. Weitere Sprachen

## Grundsprache

Bei der „Grundsprache“ **XML** (*eXtensible Markup Language*, zu Deutsch *erweiterbare Auszeichnungssprache*) handelt es sich um eine **Auszeichnungssprache**, die es ermöglichen soll, **Daten strukturieren** zu können und diese in einer Textdatei so zu speichern, dass diese sowohl **von Menschen als auch von Maschinen leicht gelesen** werden kann.

Die Verwendung von XML nimmt stark zu und findet vor allem in Netzwerken (inkl. dem Internet) statt. Dabei wird XML zum **Übermitteln von Daten zwischen Computersystemen** (Endgeräten und Servern) verwendet. Auf Grund des **einfachen Aufbaus** sowie der **Plattform- und Implementationsunabhängigkeit** ist auch die Integrierung von XML-Verarbeitungen in eigene Anwendungen ziemlich einfach möglich. Dies liegt daran, dass XML-Dokumente mittels diverser Programmiersprachen wie z. B. JavaScript und PHP verarbeitet werden können.

## Adressierung

Immer wieder wird es nötig sein, innerhalb eines XML-Dokuments zu **navigieren und bestimmte Knoten zu adressieren**. Für diesen Zweck kommt die Abfragesprache **XPath** zum Einsatz. XPath kann dabei sowohl in Sprachen wie z. B. JavaScript bei der Verarbeitung eines XML-Dokuments verwendet werden, aber auch in XSL-Sprachen (XSLT und XSL-FO), welche zur Transformation von XML-Dokumenten verwendet werden.

## Transformation

Ein Teil der XML-Familie sind Transformationsprachen, welche zur **Definierung von Designs und Layouts** verwendet werden. Diese Sprachen lassen sich in die Gruppe XSL (*eXtensible Stylesheet Language*, zu Deutsch *erweiterbare Stylesheet-Sprache*) einordnen.

Zur Gruppe XSL gehören 3 Sprachen: XSLT, XSL-FO und XPath. Mittels **XSLT** (*XSL Transformation*) können XML-Dokumente **in andere Dokumente transformiert** werden. In der Regel wird XSLT dazu eingesetzt, HTML-Seiten zu generieren. **XSL-FO** (*XSL Formatting Objects*) ist eine Sprache, die **einem XML-Dokument Stilregeln zuweist** (so wie CSS einer HTML-Seite Stilregeln zuweist). Durch ein Formatierungsprogramm (den sogenannten FO-Prozessor) kann dann aus einer XML- und einer XSL-FO-Datei eine andere Datei (oftmals eine PDF-Datei) generiert werden.

## Definition

Prinzipiell gibt es für die Namen von Elementen, Attributen und Attributwerten sowie für den Aufbau des Dokumentenbaums in XML-Dokumenten keine Einschränkungen, d. h. die Namen können beliebig gewählt und die Elemente beliebig geschachtelt werden. Um dies zu ändern bzw. um **Regeln und Definitionen** (auch als Grammatik bezeichnet) für das XML-Dokument aufzustellen, gibt es 2 Sprachen: **DTD** (*Document Type Definition*, zu Deutsch *Dokumenttypdefinition*) und **XSD** (*XML Schema Definition*, zu Deutsch *XML Schema*).

## Weitere Sprachen

Neben den bisher genannten Sprachen der XML-Familie gibt es natürlich noch viele mehr. Die wohl bekannteste auf XML basierende Sprache ist XHTML. Weitere Sprachen sind z. B. SVG, GPX, MusicXML, RSS, Atom und XAML. Zu den Sprachen **RSS** und **SVG** bieten wir hier ebenfalls Tutorials an.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

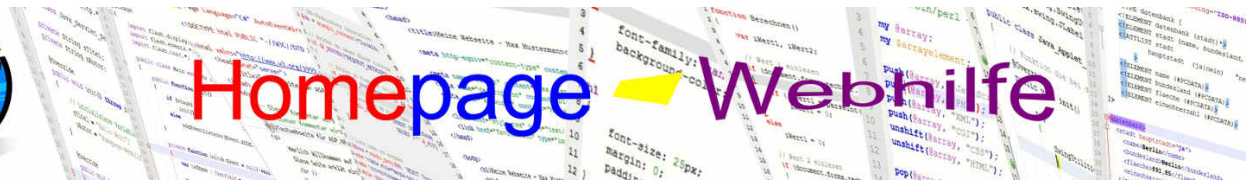
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XML](#) » [Grundlagen](#)

## Grundlagen

XML (*eXtensible Markup Language*, zu Deutsch *erweiterbare Auszeichnungssprache*) ist eine Auszeichnungssprache, welche als **Basis für viele weitere Sprachen** verwendet wird und vom World Wide Web Consortium (kurz W3C) spezifiziert wurde. Die Sprache XML kommt immer dann zum Einsatz, wenn **Daten strukturiert in einer Textdatei abgelegt** werden sollen.

Ein XML-Code wird in der Regel in einer Datei mit der Endung `.xml` abgelegt. Der MIME-Typ von XML-Dateien (auch als XML-Dokumente bezeichnet) ist `application/xml`. Der MIME-Typ `text/xml` war ebenfalls für XML-Dokumente gedacht, gilt aber in der Zwischenzeit als veraltet (engl. *deprecated*).

XML-Dateien sind i. d. R. UTF-8-Dateien und können somit alle **darstellbare Zeichen** (Textzeichen) des Zeichenvorrats von UTF-8 enthalten. Das Speichern von **Binärdaten** innerhalb einer XML-Datei ist nur nach einer Umkodierung in Textzeichen (z. B. mittels Base64) möglich.

## Struktur

Der Aufbau bzw. die Struktur eines XML-Dokuments ist relativ simpel: Es gibt Elemente, Attribute und Texte. **Elemente** können ineinander beliebig **geschachtelt** werden. Ein Element kann dabei über keine, ein oder mehrere **Attribute** verfügen. Letztendlich kann ein Element noch einen **Text** enthalten.

Um sich diese Struktur leichter vorstellen zu können, möchten wir an dieser Stelle einen **Vergleich mit einem Baum** ziehen. Ein Baum verfügt ebenfalls über mehrere **Äste** (dies sind die XML-Elemente). Die Äste können sich dabei **beliebig und unterschiedlich verzweigen**. Des Weiteren kann jeder Ast **Blätter** besitzen (dies sind die XML-Attribute). Dabei ist zu beachten, dass natürlich nicht jeder Ast über Blätter verfügen muss und die Anzahl der Blätter sich unterscheiden kann. Schlussendlich können an einem Ast neben den Blättern noch Früchte hängen. Diese **Früchte** werden „zusammengefasst“ und stellen in unserem XML-Dokument den Text dar.

Eine weitere Gemeinsamkeit, welche sich bei einem XML-Dokument und einem Baum finden lässt, ist, dass beide über exakt einen **Ursprung** verfügen. Ein Baum besitzt einen Stamm und ein XML-Dokument enthält ein Wurzelement. Aus diesem entspringen dann weitere Äste bzw. Elemente.



Bildquelle: [Vektor-Grafik von Freepik](#)

## Syntax

Elemente werden in XML mit Hilfe von einem Tag-Paar notiert. Tag-Paar bedeutet, dass es **einen Start-Tag und einen End-Tag** gibt. Sowohl im Start-Tag als auch im End-Tag wird der **Name des Elements** in spitzen Klammern (`<` und `>`) angegeben. Der End-Tag verfügt zwischen dem `<`-Zeichen und dem Elementnamen noch zusätzlich über einen Schrägstrich.

```
1 <kontakte>
2
3 </kontakte>
```

Innerhalb eines solchen Elements können nun entweder weitere Elemente, Text oder auch nichts Weiteres mehr angegeben werden. Die **Verschachtelung** ist dabei beliebig. Es gilt nur eine Regel: Elemente müssen in der umgekehrten Reihenfolge geschlossen werden wie diese geöffnet wurden.

```
1 <kontakte>
2   <person>
3
4   </person>
5   <person>
6
7     <adresse>
8
9     </adresse>
10  </person>
11  <person>
12
13    <adresse>
14
15    </adresse>
16  </person>
17 </kontakte>
```

Wenn Sie einem Element keine weiteren Elemente oder Text unterordnen möchten, so können Sie auch auf ein „**leeres**“ Element zurückgreifen. Hierfür notieren wir einen **Leer-Tag**, welcher sich von dem Start-Tag nur so unterscheidet, dass ein Schrägstrich zwischen Elementname und dem `>`-Zeichen notiert wird. In der Regel wird vor dem Schrägstrich noch ein Leerzeichen notiert. Dies dient jedoch nur der besseren Lesbarkeit.

```
1 | <absatz />
```

Als nächstes wollen wir uns die **Notation von Attributen** anschauen. Ein Attribut besteht immer aus drei Teilen: dem Attributnamen, einem Gleichheitszeichen und dem Attributwert. Der Attributwert muss dabei in **doppelten Anführungszeichen** angegeben werden.

```
1 | <person geschlecht="m">
2
3 | </person>
```

Der erste Teil jedes XML-Dokuments ist i. d. R. eine **XML-Deklaration**, welche in Form einer Verarbeitungsanweisung notiert wird. Eine **Verarbeitungsanweisung** beginnt mit dem Spezial-Tag `<?xml` und endet mit dem Spezial-Tag `?>`. Zwischen diesen Tags können nun Attribute angegeben werden: `version`, `encoding` und `standalone`. Das Attribut `version` muss angegeben werden und besitzt im Regelfall den Wert `1.0`. Zwar gibt es bereits eine Version 1.1 der XML-

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XML](#) » [Grundlagen](#)

Spezifikation, jedoch ist diese kaum verbreitet, weshalb empfohlen wird, die Version 1.0 zu nutzen, sofern die Funktionen von Version 1.1 nicht erforderlich sind. Mit dem Attribut `encoding` kann die **Zeichenkodierung** des Dokuments festgelegt werden. Dieses Attribut ist jedoch nicht erforderlich. Wird es weggelassen, so wird von der UTF-8-Kodierung ausgegangen. Das optionale Attribut `standalone` gibt an, ob sich eine Dokumenttypdefinition (DTD) im gleichen Dokument (Wert `yes`) oder in einer externen Datei (Wert `no`) befindet. Auch wenn die Attribute `encoding` und `standalone` optional sind, muss die Reihenfolge (`version - encoding - standalone`) stets eingehalten werden.

```
1 | <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
```

So wie in fast jeder anderen Sprache auch, können in XML **Kommentare** notiert werden. Kommentare enthalten einen beliebigen Text und werden hauptsächlich zu **Dokumentationszwecken** verwendet. Sie werden von XML-Parsen (also Programmen, die den XML-Code verarbeiten) ignoriert. Die Angabe von XML-Kommentaren erfolgt so wie in HTML auch: `<!-- Text -->`

```
1 | <!-- Dieser Inhalt wird nicht interpretiert! -->
```

**Wichtig:** Die Namen von Elementen und Attributen sowie die Attributwerte sind frei wählbar. Auch die Struktur ist beliebig. Dies gilt natürlich nur, sofern das XML-Dokument nicht durch eine DTD oder XSD definiert ist.

## Sonderzeichen

Prinzipiell sind in XML-Dokumenten alle Zeichen des jeweiligen Zeichensatzes erlaubt. Jedoch gibt es ein paar Sonderzeichen in XML, die nicht direkt verwendet werden dürfen. Die Angabe dieser Zeichen erfolgt dann mittels einer speziellen Notation (siehe Tabelle).

Zeichen	Code (Name)	Code (Dez.)	Code (Hex.)
<	&lt;	&#60;	&#x3C;
>	&gt;	&#62;	&#x3E;
&	&amp;	&#38;	&#x26;
"	&quot;	&#34;	&#x22;
'	&apos;	&#39;	&#x27;

Hierzu ein kurzes Beispiel:

```
1 | <firma>Max Mustermann GmbH & Co. KG</firma>
```

Kommen innerhalb eines Textabschnitts mehrere dieser Sonderzeichen vor, so ist es „umständlich“, alle diese Sonderzeichen über den entsprechenden Code anzugeben. Für solche Fälle gibt es sogenannte **CDATA-Abschnitte**. Innerhalb dieser Abschnitte dürfen alle Zeichen notiert werden, ohne diese speziell umformatieren zu können. Ein solcher CDATA-Abschnitt sieht wie folgt aus:

```
1 | <![CDATA[
2 |     Dieser Abschnitt darf auch die Zeichen <, >, &, " und ' enthalten.
3 | ]]>
```

**Übrigens:** Abschnitte, bei denen der Zeicheninhalt direkt angegeben wird, werden als **PCDATA-Abschnitte** bezeichnet.

## Beispiele

Im folgenden XML-Dokument gibt es das Wurzelement mit dem Namen `sprachen`. Diesem sind nun Elemente mit dem Namen `sprache` untergeordnet. Zur Gruppierung wurden die `sprache`-Elemente den Elementen `adressierung`, `transformation` und `definition` untergeordnet.

```
1 | <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 |
3 | <sprachen>
4 |   <sprache>XML</sprache>
5 |   <adressierung>
6 |     <sprache>XPath</sprache>
7 |   </adressierung>
8 |   <transformation>
9 |     <sprache>XSLT</sprache>
10 |    <sprache>XSL-F0</sprache>
11 |   </transformation>
12 |   <definition>
13 |     <sprache>DTD</sprache>
14 |     <sprache>XSD</sprache>
15 |   </definition>
16 | </sprachen>
```

Das folgende XML-Dokument enthält im Vergleich zum obigen Beispiel noch eine Erweiterung: Die `sprache`-Elemente verfügen über ein Attribut mit dem Namen `basierend`. Dieses Attribut gibt an, ob es sich bei der Sprache um eine XML basierte Sprache handelt (Wert `ja`) oder nicht (Wert `nein`).

```
1 | <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 |
3 | <sprachen>
4 |   <sprache>XML</sprache>
5 |   <adressierung>
6 |     <sprache basierend="nein">XPath</sprache>
7 |   </adressierung>
8 |   <transformation>
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XML](#) » [Grundlagen](#)

```
9      <sprache basierend="ja">XSLT</sprache>
10     <sprache basierend="ja">XSL-FO</sprache>
11   </transformation>
12   <definition>
13     <sprache basierend="nein">DTD</sprache>
14     <sprache basierend="ja">XSD</sprache>
15   </definition>
16 </sprachen>
```

**Übrigens:** XML-Dokumente können vom Browser direkt angezeigt werden. Der Browser zeigt dabei lediglich den Quellcode an (da keine Style-Informationen verfügbar sind).

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislöcher



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XML](#) » [Namensräume](#)

## Namensräume

In XML ist es möglich, für Elemente einen Namensraum festzulegen. Namensräume sind eigentlich dazu gedacht, **Namenskonflikte zu vermeiden**. Zu so einem Konflikt kann es immer dann kommen, wenn in einem XML-Dokument unterschiedliche **XML-Varianten** gemischt werden.

Ein **Beispiel**: Sie haben eine eigene XML-Datei geschrieben, in welcher es das Element `article` gibt. Ein bestimmter Teil Ihrer XML-Datei soll nun aber HTML-Code enthalten. In HTML gibt es aber ebenfalls das Element `article`. Sie haben einen Namenskonflikt.

Genau für solche Zwecke bzw. grundsätzlich, wenn Sie Elemente aus einer anderen XML-Variante (wie z. B. von HTML) verwenden, sollten Sie einen **Namensraum angeben**. Die Definition des Namensraums erfolgt mittels des Attributs `xmlns`, einem Präfix und einer URI (meist in Form der URL). Hinter der angegebenen URL muss in der Realität nichts existieren, sie muss sich lediglich unterscheiden. Es empfiehlt sich jedoch, hinter der URL eine Spezifikation zu hinterlegen (bzw. auf die Spezifikation z. B. vom W3C zu verlinken). Die vollständige Angabe eines Namensraums lautet `xmlns:Präfix="URI"`. Alle Elementnamen, welche diesem Namensraum angehören, müssen nun mit dem Präfix erweitert werden. Aus einem Element mit dem Namen `article` und dem Präfix mit dem Namen `h` würde also `h:article` werden. Die Angabe des Namensraums kann entweder im Wurzelement oder im ersten Element, in welchem der Namensraum benötigt wird, erfolgen.

Im folgenden Beispielcode ist das oben beschriebene Szenario mittels Namensraumangabe gelöst:

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3  <article>
4      <info>
5          <autor>Max Mustermann</autor>
6          <datum>15.01.2017</datum>
7      </info>
8      <inhalt>
9          <h:article xmlns:h="http://www.w3.org/TR/html5/">
10             <h:h1>XML Namensräume</h:h1>
11             <h:p>Dies ist ein Newsletter zum Thema Namensräume
12                 in der Auszeichnungssprache XML.</h:p>
13         </h:article>
14     </inhalt>
15 </article>

```

**Übrigens:** Wenn alle Elemente, welche einem bestimmten Element untergeordnet sind, einem anderen Namensraum angehören, so kann auf den Präfix verzichtet werden. Das Attribut lautet dann lediglich `xmlns`. Im Beispiel oben wäre dies möglich.

**Wichtig:** Alle Elemente, bei denen kein Namensraum explizit angegeben wurde, gehören dem Standardnamensraum an.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XML](#) » Wohlgeformtheit und Gültigkeit

## Wohlgeformtheit und Gültigkeit

Ein XML-Dokument kann bzw. sollte einige **Regeln** einhalten, andernfalls besteht die Gefahr, dass das Dokument nicht richtig verarbeitet / interpretiert werden kann. Alle Regeln lassen sich grundsätzlich in zwei Gruppen einteilen: Wohlgeformtheit und Gültigkeit. Diese beiden Begriffe sowie deren Regeln werden in den folgenden Abschnitten genauer erklärt.

### Inhalt dieser Seite:

1. Wohlgeformtheit
2. Gültigkeit

### Wohlgeformtheit

Ein XML-Dokument gilt als wohlgeformt (engl. *well-formed*), **wenn alle XML-Regeln eingehalten sind**. Hier eine Auflistung der wichtigsten Regeln:

- Die XML-Deklaration ist optional, darf jedoch nicht mehrmals vorkommen.
- Der Verarbeitungshinweis enthält das `version`-Attribut.
- Die Reihenfolge der Attribute im Verarbeitungshinweis wurde eingehalten: `version` - `encoding` - `standalone`.
- Das XML-Dokument verfügt über exakt ein Wurzelement.
- Die XML-Elemente werden in der umgekehrten Reihenfolge geschlossen, wie diese geöffnet wurden.
- Zu jedem Start-Tag existiert exakt ein End-Tag (Groß- und Kleinschreibung beachten).
- Leer-Tags sind entsprechend mit dem Schrägstrich gekennzeichnet.
- Innerhalb eines Elements darf jedes Attribut nur einmal vorkommen.
- Der Wert eines Attributs muss in doppelten Anführungszeichen angegeben werden.
- Sonderzeichen müssen entsprechend angegeben werden (z. B. `&lt;` statt `<`) sofern diese nicht in einem CDATA-Abschnitt vorkommen.

### Gültigkeit

Die Wohlgeformtheit eines XML-Dokuments ist i. d. R. erforderlich, um das Dokument überhaupt richtig verarbeiten zu können. Die Gültigkeit (auch als Validität bezeichnet) eines XML-Dokuments ist jedoch erst dann erfüllt, wenn das Dokument wohlgeformt ist, über eine Definition oder Schema (mittels DTD oder XSD) verfügt und **die in der Definition oder in dem Schema definierten Regeln eingehalten sind**. XML-Dokumente ohne DTD oder XSD sind daher u. U. wohlgeformt, können jedoch niemals gültig bzw. valide sein.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

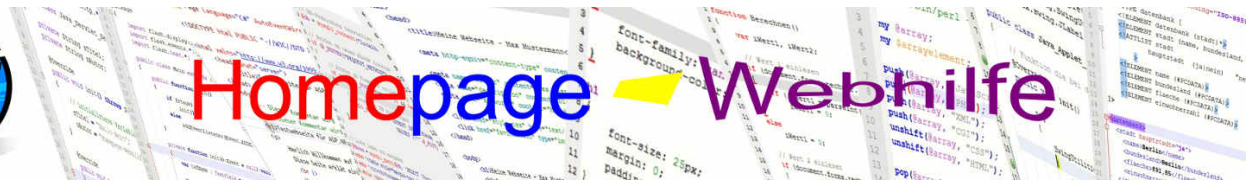
#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XPath](#) » [Grundlagen](#)

## Grundlagen

Bei XPath (*XML Path Language*, zu Deutsch *XML Pfadsprache*) handelt es sich um eine **Abfragesprache**, die dazu verwendet wird, **Knoten innerhalb eines XML-Dokuments zu adressieren**. Verwendung findet XPath dabei vor allem in der Transformationssprache XSLT, kann jedoch auch in anderen Programmiersprachen (z. B. JavaScript oder C#) für die Adressierung von Bestandteilen des XML-Baums verwendet werden.

Die mit XPath definierten Ausdrücke werden nicht in extra Dateien abgespeichert, sondern in anderen Dokumenten bzw. in Skripten eingebettet. Die Sprache XPath wurde vom W3C entworfen und dient zudem als Grundlage für weitere Sprachen.

In diesem Thema werden wir verschiedene Fachbegriffe (wie z. B. Knoten und Achse) sowie die Schlüsselwörter und den Syntax von XPath erläutern.

### Inhalt dieser Seite:

1. Knoten
2. Achsen
3. Prädikate
4. Pfade
5. Beispiele

## Knoten

In Bezug auf XML und XPath werden Sie oft den Begriff Knoten (engl. *node*) hören. Doch was genau ist ein Knoten? Unter einem Knoten versteht man **einen Teil eines XML-Dokuments** oder auch das ganze XML-Dokument. Dabei stehen Knoten in enger Verbindung mit der Baumstruktur von XML. Bei Knoten wird zwischen ein paar unterschiedlichen Typen unterschieden:

- **Wurzelknoten** (engl. *root node*, auch Dokumentknoten genannt): Dabei handelt es sich um den abstrakten Ursprung des XML-Dokuments. Der Wurzelknoten repräsentiert das ganze Dokument und darf nicht mit dem Wurzelement verwechselt werden.
- **Elementknoten** (engl. *element node*): ein beliebiges Element.
- **Textknoten** (engl. *text node*): ein Text, welcher einem Element untergeordnet ist.
- **Attributknoten** (engl. *attribute node*): ein beliebiges Attribut eines Elements.
- **Kommentarknoten** (engl. *comment node*): ein beliebiger Kommentar.
- **Namensraumknoten** (engl. *namespace node*): eine beliebige Namensraumangabe eines Elements oder Attributs.
- **Verarbeitungsanweisungsknoten** (engl. *processing instruction node*): ein XML-Verarbeitungshinweis (die XML-Deklaration zählt dabei nicht dazu).

## Achsen

Für die Pfade, welche nachher genutzt werden, um im Dokument zu navigieren bzw. Knoten zu adressieren, werden sogenannte Achsen (engl. *axis*) benötigt. Eine Achse spezifiziert dabei die **Beziehung des gewünschten Knotens** (also dem Knoten, welchen Sie mittels XPath adressieren möchten) **zum aktuellen Knoten** (auch als Kontextknoten bezeichnet). Einige Achsen adressieren nur einzelne Knoten, wohingegen mit anderen Achsen auch mehrere Knoten adressiert werden können. Die folgende Tabelle zeigt die verschiedenen Achsen (Name und Kurz-Notation) sowie die Knoten, die damit selektiert werden:

Name	Kurz-Notation	selektierte Knoten
/	/	Wurzelknoten
child	(nicht notwendig)	direkt untergeordnete Knoten (Kindknoten)
self	.	aktuelle Knoten (Kontextknoten)
parent	..	direkt übergeordneter Knoten (Elternknoten)
descendant	..//	alle untergeordnete Knoten
descendant-or-self		alle untergeordnete Knoten sowie der aktuelle Knoten
ancestor		alle übergeordnete Knoten
ancestor-or-self		alle übergeordneten Knoten sowie der aktuelle Knoten
following		alle nachfolgende Knoten (ohne Kindknoten)
following-sibling		alle nachfolgende Knoten (ohne Kindknoten), die den gleichen Elternknoten haben
preceding		alle vorangehende Knoten (ohne alle Elternknoten)
preceding-sibling		alle vorangehende Knoten (ohne alle Elternknoten), die den gleichen Elternknoten haben
attribute	@	Attributknoten
namespace		Namensraumknoten

## Prädikate

Ein Prädikat (engl. *predicate*) wird in XPath dazu verwendet, mit Hilfe von Ausdrücken (i. d. R. handelt es sich dabei um Bedingungen) die **Selektierung von Knoten weiter einzuschränken**. Ein typisches Beispiel: Sie möchten alle Elemente mit dem Namen `x` selektieren, aber nur, sofern diese über das Attribut `y` verfügen. Die Notation von Prädikaten erfolgt in den eckigen Klammern `[ ]` und `]`.

Innerhalb der Prädikate können Sie verschiedene Operatoren verwenden. Dazu zählen die mathematischen Operatoren `+` (Addition), `-` (Subtraktion), `*` (Multiplikation), `div` (Division) und `mod` (Modulo, Divisionsrest), aber auch die logischen Operatoren `and` (und) und `or` (oder) sowie die Vergleichsoperatoren `=` (gleich), `!=` (ungleich), `<` (kleiner als), `<=` (kleiner-gleich), `>` (größer als) und `>=` (größer-gleich).

Eine weitere beliebte Verwendung von Prädikaten ist die **Adressierung eines einzelnen Knotens** mit Hilfe des Indexes. Bei einem **Index** handelt es sich um eine laufende Nummer. Bei der Notation wird der Index direkt als Zahl innerhalb der eckigen Klammern angegeben: `1` = erster Knoten, `2` = zweiter Knoten, etc..

Oft werden Prädikate aber auch **in Verbindung mit XPath-Funktionen** (dazu mehr im [nächsten Thema](#)) verwendet. Dadurch ist z. B. die Adressierung des letzten Knotens, das Modifizieren von Zahlen und Zeichenketten und vieles mehr möglich.

## Pfade

Ein XPath-Ausdruck (also der Pfad, welcher zur Adressierung von Knoten verwendet wird) setzt sich aus einem oder mehreren Lokalisierungsschritt(en) zusammen.

<b>Über uns</b> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<b>Community</b> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<b>Nachschlagewerk</b> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	Benjamin Jung Krummstraße 9/3 73054 Eislingen
			Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a>



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XPath](#) » [Grundlagen](#)

Ein **Lokalisierungsschritt** besteht wiederum aus weiteren 3 Teilen: der Achse, einem Knotentest und bei Bedarf aus einem oder mehreren Prädikat(en). Wird bei der Angabe der Achse der Achsenname verwendet, so muss der Achsenname vom Knotentest mit Hilfe der Zeichen `:` (Doppel-Doppelpunkt) getrennt werden. Wird die **verkürzte Notation** verwendet, so fällt der Doppel-Doppelpunkt weg. Der Knotentest ist letztendlich nichts anderes als der Name des Knotens. Alternativ kann hier auch mit dem sogenannten **Wildcard-Zeichen** `*` gearbeitet werden. Das Wildcard-Zeichen dient dabei als Platzhalter für den Namen. Mit `*` können somit also alle Knoten (ausgehend von dem aktuellen Knoten und abhängig von der Achse) unabhängig vom Namen selektiert werden. Lokalisierungsschritte werden mit Hilfe des Schrägstrichs `/` voneinander getrennt.

## Beispiele

Wie Sie sicherlich bemerkt haben, war dieses Thema sehr theorielastig. Daher wollen wir uns das theoretische Wissen nun noch an ein paar Praxisbeispielen verdeutlichen. In den Beispielen nutzen wir folgendes XML-Dokument:

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3  <adressbuch>
4    <kontakt geschlecht="m" alter="17">
5      <name>Max Mustermann</name>
6      <strasse>Musterstraße 123</strasse>
7      <ort>12345 Musterstadt</ort>
8    </kontakt>
9    <kontakt geschlecht="w" alter="24">
10     <name>Maria Musterfrau</name>
11     <strasse>Musterstraße 456</strasse>
12     <ort>67890 Musterhafen</ort>
13   </kontakt>
14   <kontakt geschlecht="m" alter="53">
15     <name>Peter Müller</name>
16     <strasse>Hauptstraße 29</strasse>
17     <ort>27946 Programmierstadt</ort>
18   </kontakt>
19 </adressbuch>

```

Eine kurze Erklärung dazu: `adressbuch` ist das **Wurzelement** (nicht der Wurzelknoten), welchem 3 **Kindelemente** (child) mit dem Namen `kontakt` untergeordnet sind. Das **Elternelement** (parent) der `kontakt`-Elemente ist daher das Element `adressbuch`. Jedes `kontakt`-Element besitzt das Attribut `geschlecht`, welches angibt, ob es sich um eine männliche oder weibliche Person handelt, und das Attribut `alter`, welche das Alter der Person spezifiziert. Genauere Angaben zur Person werden über die Elemente `name`, `strasse` (eigentlich `straße`, jedoch sollten Sonderzeichen in Knotennamen vermieden werden) und `ort` festgelegt. Zu den **übergeordneten Knoten** (ancestor) des Elements `ort`, gehört daher `kontakt` und `adressbuch`. **Untergeordnete Knoten** (descendant) aus Sicht des Elements `adressbuch` sind die `kontakt`-Elemente sowie deren ungeordneten Elemente `name`, `strasse` und `ort`.

Nun wollen wir uns jedoch ein paar XPath-Ausdrücken widmen. Folgender Ausdruck selektiert alle `kontakt`-Elemente:

```
1 | /child::adressbuch/child::kontakt
```

Die weiter oben angesprochene **verkürzte Notation** würde dazu wie folgt aussehen:

```
1 | /adressbuch/kontakt
```

Möchten wir z. B. nur das erste `kontakt`-Element selektieren, so notieren wir den folgenden Ausdruck:

```
1 | /child::adressbuch/child::kontakt[1]
```

In der Regel wird die ausführliche Notation nur selten verwendet. In den nächsten Beispielen entspricht der erste Ausdruck immer der ausführlichen Notation und der zweite Ausdruck der verkürzten Notation. Hier die verkürzte Notation zum Ausdruck von oben:

```
1 | /adressbuch/kontakt[1]
```

Wollen wir alle `kontakt`-Elemente selektieren, die über das Attribut `geschlecht` verfügen (unabhängig vom Wert), dann könnten wir folgenden Ausdruck verwenden:

```
1 | /child::adressbuch/child::kontakt[attribute::geschlecht]
1 | /adressbuch/kontakt[@geschlecht]
```

Es ist aber auch möglich, mittels Prädikat mehrere Attribute „auf einmal“ abzufragen. Im folgenden Beispiel werden alle `kontakt`-Elemente selektiert, die über das Attribut `geschlecht` und `alter` verfügen:

```
1 | /child::adressbuch/child::kontakt[attribute::geschlecht and attribute::alter]
1 | /adressbuch/kontakt[@geschlecht and @alter]
```

Stellen Sie sich vor, Sie möchten als nächstes alle `kontakt`-Elemente selektieren, aber nur dann, wenn der Wert des Attributs `geschlecht` `m` ist:

```
1 | /child::adressbuch/child::kontakt[attribute::geschlecht='m']
1 | /adressbuch/kontakt[@geschlecht='m']
```

Mit dem Ausdruck von oben erhalten Sie nun die kompletten `kontakt`-Knoten aller männlichen Kontakte Ihres Adressbuchs. Es ist aber auch möglich, nur die Namen dieser Personen herauszufinden, indem wir das `name`-Element selektieren:

```
1 | /child::adressbuch/child::kontakt[attribute::geschlecht='m']/child::name
1 | /adressbuch/kontakt[@geschlecht='m']/name
```

Bei einer Selektierung mit dem Ausdruck von oben werden Sie feststellen (sofern Sie den Ausdruck getestet haben), dass das Ergebnis nicht `Max Mustermann` und `Peter Müller` ist, sondern `<name>Max Mustermann</name>` und `<name>Peter Müller</name>`. In einigen Fällen kann es jedoch notwendig sein,

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

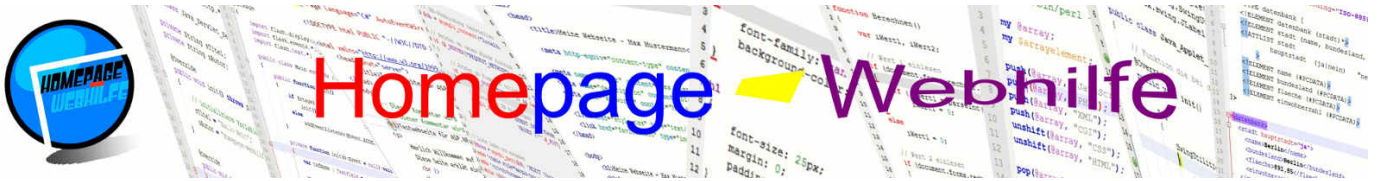
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XPath](#) » [Grundlagen](#)

nicht (wie hier) das `name`-Element zu selektieren, sondern dessen Textinhalt (also den Textknoten). Für diesen Fall wird als Knotentest `text()` angegeben. Dadurch kann dann ein Textknoten (der nicht mittels eines Namens selektiert werden kann) adressiert werden. Dies sieht dann bspw. so aus:

```
1 | /child::adressbuch/child::kontakt[attribute::geschlecht='m']/child::name/child::text()
1 | /adressbuch/kontakt[@geschlecht='m']/name/text()
```

Attribute, deren Inhalt ein Zahlenwert ist, können auch mit dem Kleiner-Als- und Größer-Als-Operator verglichen werden. Im folgenden Beispiel werden nur Kontakte selektiert die mindestens 18 Jahre alt sind:

```
1 | /child::adressbuch/child::kontakt[attribute::alter >= 18]
1 | /adressbuch/kontakt[@alter >= 18]
```

Zum Schluss noch ein Beispiel mit dem Wildcard-Zeichen: Mit dem folgenden Ausdruck selektieren wir alle Kindknoten (unabhängig von deren Namen) der `kontakt`-Elemente:

```
1 | /child::adressbuch/child::kontakt/child::*
1 | /adressbuch/kontakt/*
```

**Wichtig:** In diesen Beispielen wurde immer mit **absoluter Adressierung** gearbeitet. In Sprachen wie XSLT wird dies nur selten verwendet, da man sich hier Stück für Stück durch das Dokument durcharbeitet. Für die Beispiele hier hätten wir jedoch dann immer dazu schreiben müssen, welcher Knoten der Kontextknoten, also der „aktuelle“ Knoten ist, um das Beispiel überhaupt nachvollziehen zu können.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

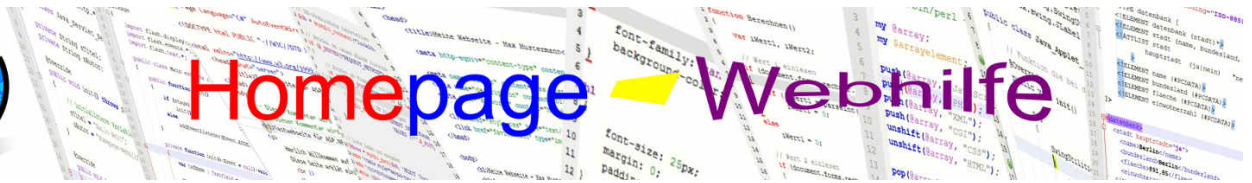
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XPath](#) » Funktionen

## Funktionen

In XPath gibt es, wie den meisten vor allem aus Programmier- und Skriptsprachen bekannt sein wird, Funktionen. Diese Funktionen können in Prädikaten oder auch direkt in XSLT verwendet werden. Funktionen werden dabei mit dem Funktionsnamen, gefolgt von einem runden Klammernpaar, angegeben. Innerhalb des Klammernpaars können sogenannte Parameter übergeben werden. Die **Parameter** (auch Argumente genannt) werden dabei durch ein Komma getrennt. Die Anzahl der Parameter und deren Bedeutung unterscheiden sich von Funktion zu Funktion. XPath-Funktionen geben grundsätzlich einen Wert zurück. Die Funktionen von XPath lassen sich grob in 4 Gruppen teilen: Positions-Funktionen, Logik-Funktionen, Zahlen-Funktionen und Zeichenketten-Funktionen.

### Inhalt dieser Seite:

1. Position
2. Logik
3. Zahlen
4. Zeichenketten

## Position

Positions-Funktionen erlauben es, mit Hilfe von Positionsangaben (also dem Index) zu arbeiten. Die Funktion `position()` gibt die Positionsnummer des **aktuellen Knotens** zurück. Vorher hatten wir z. B. folgendes Beispiel:

```
1 | /adressbuch/kontakt[1]
```

In der Tat ist das Prädikat `[1]` eigentlich nur eine verkürzte Schreibweise. Die ausführliche Notation sieht wie folgt aus:

```
1 | /adressbuch/kontakt[position() = 1]
```

Die Funktion `last()` erlaubt es, die Positionsnummer des **letzten Knotens** zu ermitteln. Die Selektierung des letzten `kontakt`-Elements wäre mittels des folgenden Syntax möglich:

```
1 | /adressbuch/kontakt[position() = last()]
```

Auch dies lässt sich wieder verkürzt darstellen:

```
1 | /adressbuch/kontakt[last()]
```

Mit der Funktion `count()` können Sie die **Anzahl der enthaltenen Knoten** zu einem Knotentest bestimmen. Als Parameter übergeben Sie den Knotentest. Der folgende Ausdruck zählt die Anzahl der `kontakt`-Elemente:

```
1 | count(/adressbuch/kontakt)
```

## Logik

In der Logik gibt es den Datentyp `bool` (auch `boolean` genannt), welcher einen **Wahrheitswert** speichern kann: `true` (wahr) oder `false` (unwahr). Mit den Funktionen `true()` und `false()` ist es möglich, einen solchen booleschen Wert zu erzeugen. Ein boolescher Wert kann mit der Funktion `not()` negiert (also umgekehrt) werden. Aus `true` wird dann `false` und aus `false` wird `true`. Als Parameter wird der zu negierende Wert übergeben. Eine weitere Funktion ist `boolean()`. Die Funktion `boolean()` dient dazu, einen Ausdruck mit `true` oder `false` zu bewerten. Dieser Ausdruck muss als Parameter übergeben werden. Hier kann z. B. auch einfach nur ein Knotenname angegeben werden, um zu prüfen, ob dieser existiert:

```
1 | boolean(/adressbuch/kontakt[1]/name)
```

## Zahlen

Die Funktion `number()` erlaubt es, einen Wert (in der Regel eine Zeichenkette) **in eine Zahl umzuwandeln**. Scheitert dieser Vorgang, so wird `NaN` (*Not A Number*) zurückgegeben. Der „umzuwandelnde“ Wert kann dabei als Parameter übergeben werden. Wird dieser weggelassen, so wird der Inhalt des aktuellen Knotens verwendet. Nun gibt es noch drei weitere Funktionen, die in Verbindung mit **Gleitkommazahlen** verwendet werden: `ceiling()` um die Zahl auf die nächstgrößere Ganzzahl aufzurunden, `floor()` um die Zahl auf die nächstkleinere Ganzzahl abzurunden und `round()` um eine kaufmännische Rundung durchzuführen. Eine weitere interessante Funktion ist `sum()`. Hier wird ein ganz normaler XPath-Ausdruck übergeben und die **Summe aller Knoten-Werte** zurückgegeben:

```
1 | sum(/adressbuch/kontakt/@alter)
```

## Zeichenketten

Als Zeichenkette bezeichnet man eine Folge von Zeichen. Eine Zeichenkette wird (wenn Sie direkt in XSL-Dokumenten verwendet werden) innerhalb einfacher Anführungszeichen notiert und kann dabei aus 0, 1 oder mehreren Zeichen bestehen. Wie es für Zahlen die Funktion `number()` gibt, um Ausdrücke in Zahlen umzuwandeln, gibt es für Zeichenketten die Funktion `string()`, um einen Ausdruck (i. d. R. eine Zahl) **in eine Zeichenkette umzuwandeln**. Die Funktion `string-length()` erlaubt es, die **Länge einer Zeichenkette** zu ermitteln. Die Zeichenkette, von welcher die Länge ermittelt werden soll, kann als Parameter übergeben werden. Wird der Parameter weggelassen, so wird der Inhalt des aktuellen Knotens verwendet.

Die Funktion `concat()` ermöglicht es, eine beliebige Anzahl an **Zeichenketten aneinander zu reihen**. Dabei wird jede Zeichenkette als Parameter übergeben. Dies könnte z. B. wie folgt aussehen:

```
1 | concat(/adressbuch/kontakt[0]/nachname, ', ', /adressbuch/kontakt[0]/vorname)
```

Die Funktion `contains()` prüft, ob eine bestimmte Zeichenkette (1. Parameter) eine bestimmte Teilzeichenkette (2. Parameter) enthält. Die Funktion `starts-with()` prüft hingegen, ob eine Zeichenkette mit der angegebenen Teilzeichenkette beginnt.

Nun gibt es noch drei weitere Funktionen, mit welchen aus einer Zeichenkette **Teilzeichenketten extrahiert werden**: `substring()`, `substring-after()` und `substring-before()`. Alle Funktionen werden dabei als erster Parameter die (Quell-)Zeichenkette übergeben. Der Funktion `substring()` wird des Weiteren die Startposition sowie optional die Anzahl an Zeichen übergeben. Die Funktionen `substring-after()` und `substring-before()` extrahiert die Zeichenkette nach bzw. vor einer angegebenen Zeichenkette. Sinnvoll ist dies z. B., um den Namen oder den Wert bei einem INI-Datei ähnlichen Format (`Name=Wert`) zu extrahieren. Im folgenden Beispiel wird aus einem Namensfeld der Vor- und Nachname extrahiert:

```
1 | substring-after(/adressbuch/kontakt[1]/name, ' ')
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislinsen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XPath](#) » Funktionen

1 | `substring-before(/adressbuch/kontakt[1]/name, ' ')`

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » [Grundlagen](#)

## Grundlagen

XSLT (XSL Transformation) ist eine Programmiersprache, die dazu verwendet wird, eine **XML-Datei in ein anderes Dokument zu transformieren**. XSLT ist dabei selbst eine Sprache, die auf XML basiert und ein Teil von XSL (eXtensible Stylesheet Language), den **Stylesheet-Sprachen der XML-Familie**.

Da XSLT **XML basierend** ist, ist Ihnen der allgemeine Syntax bereits von XML bekannt. In diesem Thema lernen Sie daher nur die **Elemente und Attribute von XSLT** kennen, für was diese verwendet werden und wie diese funktionieren. Für die **Adressierung von Knoten** wird die Sprache XPath eingesetzt. Die **Abfragesprache XPath** haben Sie bereits im [vorherigen Kapitel](#) kennengelernt.

Für den Transformationsvorgang werden Programme, die sogenannten XSLT-Prozessoren, benötigt. **XSLT-Prozessoren** können aus dem Internet bezogen werden. Zudem sind die meisten **Browser in der Lage, eine XSL-Transformation durchzuführen**.

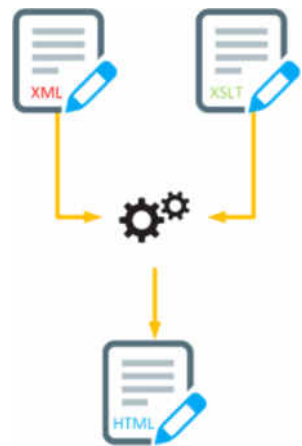
**Inhalt dieser Seite:**

1. Funktionsweise
2. Grundaufbau
3. Werte ausgeben
4. Templates
5. Variablen

## Funktionsweise

Eine XSLT-Datei enthält ein Skript, welches angibt, wie ein bestimmter Baum in einen anderen Baum transformiert (also „übersetzt“) werden soll. Typischerweise wird dabei ein **XML-Dokument in ein HTML-Dokument** (meistens also zur direkten Darstellung im Browser) übersetzt. Es ist jedoch auch möglich, ein XML-Dokument in ein anderes XML-Dokument (z. B. zur Umsetzung in eine andere Baumstruktur oder zum Filtern von Inhalten) oder ein XML-Dokument in ein anderes XML basierendes Dokument zu transformieren.

Ein XSLT-Skript kann für mehrere XML-Dokumente eingesetzt werden (so wie auch eine CSS-Datei in mehreren HTML-Dokumenten eingesetzt werden kann). Um eine **Verknüpfung** herzustellen, muss im XML-Dokument ein **Link zum Stylesheet-Dokument** angegeben werden. Die Notation erfolgt dabei als **Verarbeitungshinweis** mit dem Zusatz `-stylesheet`. Als Attribute werden dort `href` (Pfad zur Datei) und `type` (MIME-Typ des Stylesheets, für XSLT `text/xsl` oder auch `application/xslt+xml`) angegeben.



```
1 | <?xml-stylesheet href="stylesheet.xsl" type="text/xsl" ?>
```

XSLT-Skripte werden in Dateien mit der Erweiterung `.xsl` oder `.xslt` gespeichert. Den Aufbau solcher Skripte beschreiben wir im [Abschnitt Grundaufbau](#) genauer.

Um mit Hilfe einer XML- und XSLT-Datei eine weitere Datei zu erzeugen, wird ein Programm (das sogenannte Transformationsprogramm bzw. meistens als XSLT-Prozessor bezeichnet) benötigt. Dieses **Programm übersetzt die XML-Datei in eine andere Datei**, wobei die XSLT-Datei als Vorlage für den Transformationsvorgang verwendet wird. Zu den bekanntesten **XSLT-Prozessoren** gehören unter anderem Saxon und Xalan. Des Weiteren enthalten die meisten gängigen Browser einen XSLT-Prozessor. Dieser ist jedoch im Regelfall **auf die Transformation in HTML begrenzt**.

Der **XSLT-Prozessor Xalan** wird von der [Apache Software Foundation](#) entwickelt und ist in zwei unterschiedlichen Varianten verfügbar: Xalan-C (für C++) und Xalan-J (für Java). Von beiden Varianten sind sowohl Quellcode als auch vorkompilierte Versionen verfügbar. Für die Beispiele in diesem Kapitel gibt es immer zwei Ansichten: **eine XML- und eine HTML-Ansicht**. Die XML-Ansicht zeigt, sofern Ihr Browser einen XSLT-Prozessor enthält, die vom Browser transformierte Ausgabe, andernfalls wird Ihnen der XML-Baum (Quellcode) angezeigt. Die HTML-Ansicht zeigt eine HTML-Datei. Diese Datei wurde mit dem XSLT-Prozessor Xalan-J erstellt. **Xalan-J** wird über die Kommandozeile aufgerufen. Dabei übergeben Sie i. d. R. folgende Parameter: `-IN` (eingehende XML-Datei), `-XSL` (XSL-Datei) und `-OUT` (ausgehende HTML-/XML-Datei). Der folgende Code zeigt, mit welchem Kommando die HTML-Datei des ersten Beispiels (Werte ausgeben) mittels Xalan-J erstellt wurde:

```
1 | java -jar xalan.jar -IN werte-ausgeben.xml -XSL werte-ausgeben.xsl -OUT werte-ausgeben.html
```

## Grundaufbau

Jede XSLT-Datei beginnt in der Regel mit einer XML-Deklaration. Diese ist jedoch, wie bei anderen XML-Dokumenten auch, optional. Das **Wurzelement** bei XSLT ist `stylesheet`. Innerhalb dieses Elements wird nun das `version`-Attribut mit dem Wert `1.0` angegeben. Zwar gibt es schon eine Version 2.0, jedoch wird diese kaum verwendet. Dies liegt dabei vor allem an der mangelnden Unterstützung. Als weiteres Attribut im `stylesheet`-Element wird `xmlns` notiert, um einen **Namensraum zu spezifizieren**. Hier hat es sich eingebürgert, als Präfix `xsl` zu verwenden. Als URI für den Namensraum von XSLT wird `http://www.w3.org/1999/XSL/Transform` verwendet. Die Elemente die für das resultierende Dokument verwendet werden, werden i. d. R. ohne Namensraum angegeben. Innerhalb des Wurzelements werden ein oder mehrere Templates notiert. Diese stellen den Hauptbestandteil des Dokuments dar. Die Notation und Funktionsweise von Templates erklären wir jedoch später.

```
1 | <?xml version="1.0" encoding="UTF-8" ?>
2 |
3 | <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4 |
5 | </xsl:stylesheet>
```

## Werte ausgeben

Nun wollen wir uns aber auch mal dem ersten Beispiel widmen: Wie bereits oben erwähnt, besteht eine XSLT-Datei im Wesentlichen aus Templates, also Schablonen bzw. Vorlagen. Wie diese genau funktionieren bzw. wie man diese verwendet, möchten wir im nächsten Punkt genauer erklären. Als erstes wollen wir

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » [Grundlagen](#)

Ihnen jedoch zeigen, wie Sie einen **Wert aus einer XML-Datei ausgeben**. Um einen Wert im Ausgabebaum zu platzieren, benötigen wir das Element `value-of`. Ein wichtiges Attribut des `value-of`-Elements ist `select`. Mit dem `select`-Attribut wird der **auszugebende Wert mit Hilfe eines XPath-Ausdrucks selektiert**.

Im folgenden Beispiel werden die Elemente `titel` und `text` aus dem XML-Dokument als Inhalte der HTML-Elemente `h1` (Überschrift der 1. Ebene) und `p` (Absatz) platziert und somit für den Benutzer dargestellt. Das in diesem Beispiel verwendete XSLT-Element `template` wird als Vorlage benötigt und ist erforderlich. Dessen Bedeutung erklären wir jedoch erst im [nächsten Abschnitt](#).

**XML-Code:**

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="werte-ausgeben.xsl" type="text/xsl" ?>
3
4 <artikel>
5   <titel>Grundlagen von XSLT</titel>
6   <text>Hier lernen Sie die Grundlagen von XSLT ...</text>
7 </artikel>
    
```

**XSLT-Code:**

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>
6       <head>
7         <title>Werte ausgeben - XSLT Code-Beispiel</title>
8
9         <meta charset="utf-8" />
10
11        <meta name="robots" content="noindex,nofollow" />
12        <meta name="publisher" content="Homepage-Webhilfe" />
13      </head>
14
15      <body>
16        <artikel>
17          <h1><xsl:value-of select="/artikel/titel" /></h1>
18          <p><xsl:value-of select="/artikel/text" /></p>
19        </artikel>
20      </body>
21    </html>
22  </xsl:template>
23 </xsl:stylesheet>
    
```

XML:   
 HTML:

**Übrigens:** Es ist natürlich auch möglich, innerhalb des `select`-Attributs eine XPath-Funktion aufzurufen, um somit z. B. nur einen Teil einer Zeichenkette auszugeben.

**Templates**

In XSLT gibt es ein sogenanntes Template-System, d. h. es gibt **Transformationsregeln**, welche zu einem Regelsatz (dem Template oder auch als Schablone oder Vorlage bezeichnet) zusammengefasst werden. Von solchen Regelsätzen gibt es innerhalb eines XSLT-Dokuments im Normalfall mehrere. In dem Beispiel von oben (Werte ausgeben) gab es nur ein Template. Regelsätze können auf zwei verschiedene Arten genutzt werden: Anwendung und Aufruf. Bei einer **Template-Anwendung** wird ein Template immer dann angewendet, wenn ein bestimmter XPath-Ausdruck zutrifft. Bei **Template-Aufrufen** muss das gewünschte Template explizit aufgerufen werden.

Eine Schablone zeichnet sich durch das Element `template` aus. Im Element muss nun das Attribut `match` (für Template-Anwendungen) oder `name` (für Template-Aufrufe) angegeben werden. Als erstes wollen wir uns der Template-Anwendung widmen, d. h. wir benötigen das Attribut `match`. Im Attribut `match` wird ein **Knotentest** (im XPath-Syntax) angegeben, um somit das **Template auf bestimmte Knoten oder einen bestimmten Pfad zu begrenzen**. In der Regel besitzt jedes XSLT-Stylesheet über ein Template, welches sich auf Wurzelknoten bezieht (`match="/"`). Bei HTML-Dokumenten wird dort dann meist der Grundaufbau einer HTML-Seite notiert. Um nun innerhalb eines Templates **weitere Templates anzuwenden**, benötigen wir das leere Element `apply-templates`. Optional kann das Attribut `select` angegeben werden, um nur Templates, welche auf einen bestimmten Knotentest bzw. Pfad zutreffen, anzuwenden. Mit Hilfe dieser Template-Technik können Sie sich **Schritt für Schritt durch das Dokument durcharbeiten**, um somit die Transformation durchzuführen.

Im folgenden Beispiel verwenden wir die gleiche XML-Struktur wie beim vorherigen Beispiel, jedoch verfügt unser Stylesheet nun über 3 weitere Templates. Innerhalb des „Wurzel-Templates“ (`match="/"`) haben wir den HTML-Grundaufbau notiert. Im `body`-Element wird nun über `apply-templates` ein weiteres Template angewendet. In diesem Fall handelt es sich um das Template, welches das Wurzelement selektiert (`match="artikel"`). Innerhalb dieses Templates wird das HTML-Element `article` notiert. Des Weiteren wird hier erneut das `apply-templates`-Element verwendet, wodurch die beiden anderen Templates (`match="titel"` und `match="text"`) angewendet werden (in der Reihenfolge, wie diese im XML-Dokument angegeben sind).

**XML-Code:**

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="templates-anwenden.xsl" type="text/xsl" ?>
3
4 <artikel>
5   <titel>Grundlagen von XSLT</titel>
6   <text>Hier lernen Sie die Grundlagen von XSLT ...</text>
7 </artikel>
    
```

**XSLT-Code:**

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung              Krummstraße 9/3              73054 Eisingen </p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a>              E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » XML & Co. » XSLT » Grundlagen

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4    <xsl:template match="/">
5      <html>
6        <head>
7          <title>Templates anwenden - XSLT Code-Beispiel</title>
8
9          <meta charset="utf-8" />
10
11         <meta name="robots" content="noindex,nofollow" />
12         <meta name="publisher" content="Homepage-Webhilfe" />
13       </head>
14
15       <body>
16         <xsl:apply-templates />
17       </body>
18     </html>
19   </xsl:template>
20
21   <xsl:template match="artikel">
22     <article>
23       <xsl:apply-templates />
24     </article>
25   </xsl:template>
26
27   <xsl:template match="titel">
28     <h1><xsl:value-of select="." /></h1>
29   </xsl:template>
30
31   <xsl:template match="text">
32     <p><xsl:value-of select="." /></p>
33   </xsl:template>
34 </xsl:stylesheet>

```

XML:   
HTML:

Wie bereits oben erwähnt, ist es auch möglich, ein Template zu definieren, um dies später einfach nur aufzurufen. Diese Möglichkeit wird jedoch normalerweise nicht für die „normale“ Abarbeitung eines XML-Dokuments verwendet, sondern lediglich, um bestimmte „Standard-Schablonen“ zu definieren, die innerhalb des Ergebnisbaums, unabhängig von der XML-Struktur und u. U. auch mehrmals, benötigt werden. Im Beispiel unten wird diese „Regel“ gebrochen, jedoch bleibt dadurch das Beispiel leicht nachvollziehbar. Ein Template, welches für den Aufruf gedacht ist, muss über das Attribut `name` verfügen. Hier können Sie einen beliebigen, jedoch eindeutigen Namen angeben. Mit dem Element `call-template` können Sie nun ein solches Template aufrufen. Hierfür muss im `name`-Attribut der Name des Templates angegeben werden.

**XML-Code:**

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2  <?xml-stylesheet href="templates-aufrufen.xsl" type="text/xsl" ?>
3
4  <artikel>
5    <titel>Grundlagen von XSLT</titel>
6    <text>Hier lernen Sie die Grundlagen von XSLT ...</text>
7  </artikel>

```

**XSLT-Code:**

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4    <xsl:template match="/">
5      <html>
6        <head>
7          <title>Templates aufrufen - XSLT Code-Beispiel</title>
8
9          <meta charset="utf-8" />
10
11         <meta name="robots" content="noindex,nofollow" />
12         <meta name="publisher" content="Homepage-Webhilfe" />
13       </head>
14
15       <body>
16         <xsl:call-template name="Artikel" />
17       </body>
18     </html>
19   </xsl:template>
20
21   <xsl:template name="Artikel">
22     <article>
23       <h1><xsl:value-of select="/artikel/titel" /></h1>

```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen </p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



Sie befinden sich hier: Homepage-Webhilfe » XML & Co. » XSLT » Grundlagen

```

24 <p><xsl:value-of select="/artikel/text" /></p>
25 </artikel>
26 </xsl:template>
27 </xsl:stylesheet>
    
```

XML:   
 HTML: 

Übrigens: Alle 3 Beispiele (Werte ausgeben, Templates anwenden und Templates aufrufen) erzeugen die gleiche Ausgabe.

## Variablen

In XSLT haben Sie die Möglichkeit, in einer Variablen einen Wert (z. B. einen Text) zu speichern. Verwendet werden Variablen im Allgemeinen immer dann, **wenn ein Wert mehrmals benötigt wird** (um somit den mehrmaligen Zugriff auf XML-Datei zu vermeiden). Eine Variable wird über das XSLT-Element `variable` definiert. Das Attribut `name` legt dabei einen eindeutigen Namen für die Variable fest. Wird die Variable innerhalb eines `template`-Elements notiert, so handelt es sich um eine **lokale Variable** und ist auch nur innerhalb der Schablone verfügbar. Variablen die außerhalb von `template`-Elementen notiert sind, gelten als **globale Variablen** und sind im kompletten Stylesheet verfügbar. Als Wert des Attributs `select` können Sie einen XPath-Ausdruck angeben, um somit der Variable einen Wert aus der XML-Datei zuzuweisen. Wird das Attribut `select` weggelassen, so müssen Sie das Element zweiteilig (und nicht wie sonst einteilig) angeben und zwischen den Tags den Wert der Variable notieren. Es kann sich dabei um eine Konstante (also einen „festen“ Wert) oder einen variablen Wert resultierend aus einer Bedingung (dazu [später mehr](#)) handeln. Um innerhalb eines `select`-Attributs (also dem Attribut, welches Sie kennengelernt haben, um Knoten zu selektieren) oder anderen dafür vorgesehenen Attributen auf die Variable zuzugreifen, notieren Sie das Dollar-Zeichen (\$) gefolgt von dem definierten Variablennamen.

### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="variablen.xsl" type="text/xsl" ?>
3
4 <text>
5     Hier steht ein Text, welcher jedoch nur als Platzhalter fungiert.
6 </text>
    
```

### XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4     <xsl:template match="/">
5         <html>
6             <head>
7                 <title>Variablen - XSLT Code-Beispiel</title>
8
9                 <meta charset="utf-8" />
10
11                 <meta name="robots" content="noindex,nofollow" />
12                 <meta name="publisher" content="Homepage-Webhilfe" />
13             </head>
14
15             <body>
16                 <xsl:variable name="inhalt" select="/text" />
17                 <p><xsl:value-of select="$inhalt" /></p>
18             </body>
19         </html>
20     </xsl:template>
21 </xsl:stylesheet>
    
```

XML:   
 HTML: 

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
 Krummstraße 9/3  
 73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
 E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » [Bedingungen](#)

## Bedingungen

### Inhalt dieser Seite:

1. Einfache Verzweigung
2. Mehrfache Verzweigung

In XSLT gibt es, wie in anderen Programmiersprachen auch, die Möglichkeit, Bedingungen aufzustellen, um somit eine sogenannte **Fallunterscheidung durchzuführen**. Bei der Fallunterscheidung (auch als Abfrage bezeichnet) wird mit Hilfe einer Bedingung (z. B. einem Vergleich) geprüft, ob ein bestimmter Fall (also die Bedingung) zutrifft. Trifft die Bedingung zu, so kann ein bestimmter Programmteil (also ein Code) ausgeführt werden. Trifft die Bedingung nicht zu, so wird der Programmteil nicht ausgeführt. Um eine Bedingung aufzustellen, benötigen wir bestimmte Operatoren. Hier kommen die **logischen Operatoren** von XPath ins Spiel: `and` (und) und `or` (oder). Mit diesen Operatoren ist es möglich, komplexere Bedingungen bestehend aus mehreren **Teilbedingungen** aufzustellen. Zudem werden sogenannte **Vergleichsoperatoren** eingesetzt, um einen Vergleich durchführen zu können: `=` (gleich), `!=` (ungleich), `<` (kleiner als), `<=` (kleiner-gleich), `>` (größer als) und `>=` (größer-gleich). Innerhalb einer Bedingung können auch Variablen, Pfade im XPath-Syntax sowie XPath-Funktionen verwendet werden.

**Wichtig:** Die Zeichen `<` und `>` dürfen in XSLT nicht direkt verwendet werden und müssen maskiert werden: `&lt;` für `<` und `&gt;` für `>`.

## Einfache Verzweigung

Eine **Wenn-Bedingung** kann mit Hilfe des XSLT-Elements `if` aufgestellt werden. Man spricht hier auch von einer einfachen Verzweigung, d. h. der Ablauf des Programms / Skripts verzweigt sich. Der Programmteil, der innerhalb dieser Verzweigung notiert wird, wird daher nur dann ausgeführt, **wenn die angegebene Bedingung zutrifft**. Die Bedingung für die einfache Verzweigung wird innerhalb des `test`-Attribut angegeben. Im folgenden Beispiel wird das Attribut `privat` des Wurzelements auf Übereinstimmung mit dem Wert `'ja'` (einfache Anführungszeichen werden benötigt, da es sich bei dem Wert um eine Zeichenkette handelt) geprüft.

### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="einfache-verzweigung.xsl" type="text/xsl" ?>
3
4 <kontakt privat="ja">
5   <name>Max Mustermann</name>
6 </kontakt>

```

### XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>
6       <head>
7         <title>Einfache Verzweigung - XSLT Code-Beispiel</title>
8
9         <meta charset="utf-8" />
10
11        <meta name="robots" content="noindex,nofollow" />
12        <meta name="publisher" content="Homepage-Webhilfe" />
13      </head>
14
15      <body>
16        <xsl:value-of select="/kontakt/name" />
17        <xsl:if test="/kontakt/@privat = 'ja'">
18          <i>(Privat)</i>
19        </xsl:if>
20      </body>
21    </html>
22  </xsl:template>
23 </xsl:stylesheet>

```

XML: 

HTML: 

**Wichtig:** Die hier genannte Verzweigungsart kennen Sie vielleicht aus anderen Programmiersprachen. In XSLT gibt es jedoch keinen `else`-Zweig, um zusätzlich den Fall abzudecken, wenn die Bedingung nicht zutrifft.

## Mehrfache Verzweigung

Eine mehrfache Verzweigung wird i. d. R. immer dann verwendet, wenn Sie einen **Wert mit unterschiedlichen Werten vergleichen** wollen oder wenn **an Hand von mehreren Bedingungen eine Auswahl getroffen werden soll**. Bei mehrfachen Verzweigungen in XSLT ist es auch möglich, anders als in anderen Programmiersprachen, z. B. auf einen Wertebereich zu prüfen oder Teilbedingungen mit `and` und `or` zu verknüpfen (also so wie bei einfachen Verzweigungen auch). Dies liegt daran, da die mehrfache Verzweigung aus lauter einzelnen Bedingungen besteht, die theoretisch völlig unterschiedlich sein können. Der Code, welcher innerhalb der ersten zutreffenden Bedingung notiert wird, wird ausgeführt, weitere Bedingungen werden dann nicht geprüft und dessen Code natürlich auch nicht ausgeführt.

Um eine solche **Abfragereihe** zu definieren, notieren wir zu allererst das Element `choose`. Dieses Element verfügt über keine weiteren Attribute. Innerhalb des `choose`-Elements werden nun ein oder mehrere `when`-Elemente sowie bei Bedarf ein `otherwise`-Element notiert. Die `when`-Elemente stellen die **einzelnen Bedingungen** dar. Die eigentliche Bedingung wird dabei im `test`-Attribut angegeben. Der Code innerhalb des Elements wird nur dann ausgeführt, wenn die Bedingung zutrifft. Mit dem `otherwise`-Element können Sie den Fall abdecken, wenn keine der Bedingungen (definiert innerhalb der `when`-Elemente) zutrifft. Das `otherwise`-Element wird (falls benötigt) immer als letztes notiert und verfügt auch über keine Attribute.

### XML-Code:

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » [Bedingungen](#)

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="mehrfache-verzweigung.xsl" type="text/xsl" ?>
3
4 <kontakt anrede="hr">
5   <name>Max Mustermann</name>
6 </kontakt>

```

## XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>
6       <head>
7         <title>Mehrfache Verzweigung - XSLT Code-Beispiel</title>
8
9         <meta charset="utf-8" />
10
11        <meta name="robots" content="noindex,nofollow" />
12        <meta name="publisher" content="Homepage-Webhilfe" />
13      </head>
14
15      <body>
16        <xsl:choose>
17          <xsl:when test="/kontakt/@anrede = 'hr'">
18            Herr
19          </xsl:when>
20          <xsl:when test="/kontakt/@anrede = 'fr'">
21            Frau
22          </xsl:when>
23          <xsl:when test="/kontakt/@anrede = 'f'">
24            Firma
25          </xsl:when>
26          <xsl:otherwise>
27            ???
28          </xsl:otherwise>
29        </xsl:choose>
30        <xsl:value-of select="/kontakt/name" />
31      </body>
32    </html>
33  </xsl:template>
34 </xsl:stylesheet>

```

XML:   
HTML: 

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » [Schleifen](#)

## Schleifen

Eine Schleife wird in XSLT dazu verwendet, einen bestimmten **Vorgang für alle Knoten innerhalb eines Knotensets** (Reihe von Knoten) zu wiederholen. Stellen Sie sich vor, Sie haben ein XML-Datei mit dem Element `zahlen`, welchem mehrere Elemente mit dem Namen `zahl` untergeordnet sind. Nun wollen Sie diese Zahlenliste in eine HTML-Liste übersetzen. Für die Abarbeitung dieser Zahlenliste könnten Sie nun eine Schleife verwenden, um somit jede Zahl in einem `li`-Element zu platzieren. Eine Schleife zeichnet sich durch das Element `for-each` aus. Mit dem Attribut `select` wird ein Knotenset bzw. ein Pfad spezifiziert. Hier der Code für das vorher erwähnte Beispiel mit der Zahlenliste:

### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="schleife.xml" type="text/xsl" ?>
3
4 <zahlen>
5   <zahl>45</zahl>
6   <zahl>39</zahl>
7   <zahl>22</zahl>
8   <zahl>70</zahl>
9   <zahl>68</zahl>
10  <zahl>12</zahl>
11  <zahl>51</zahl>
12  <zahl>63</zahl>
13  <zahl>27</zahl>
14  <zahl>94</zahl>
15 </zahlen>

```

### XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>
6       <head>
7         <title>Schleife - XSLT Code-Beispiel</title>
8
9         <meta charset="utf-8" />
10
11        <meta name="robots" content="noindex,nofollow" />
12        <meta name="publisher" content="Homepage-Webhilfe" />
13      </head>
14
15      <body>
16        <ul>
17          <xsl:for-each select="/zahlen/zahl">
18            <li><xsl:value-of select="." /></li>
19          </xsl:for-each>
20        </ul>
21      </body>
22    </html>
23  </xsl:template>
24 </xsl:stylesheet>

```

XML:

HTML:

Falls Sie schon etwas mit XSLT experimentiert haben oder Sie sich noch genau an unsere Erklärung zu Templates erinnern, werden Sie bemerken, dass das obige Beispiel sich auch **ohne Schleife und stattdessen mit Templates** realisieren lässt. Der Code würde dann wie folgt aussehen:

### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="schleife-template.xml" type="text/xsl" ?>
3
4 <zahlen>
5   <zahl>45</zahl>
6   <zahl>39</zahl>
7   <zahl>22</zahl>
8   <zahl>70</zahl>
9   <zahl>68</zahl>
10  <zahl>12</zahl>
11  <zahl>51</zahl>
12  <zahl>63</zahl>
13  <zahl>27</zahl>
14  <zahl>94</zahl>
15 </zahlen>

```

### XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2

```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » [Schleifen](#)

```

3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>
6       <head>
7         <title>Schleifen-Alternative - XSLT Code-Beispiel</title>
8
9         <meta charset="utf-8" />
10
11        <meta name="robots" content="noindex,nofollow" />
12        <meta name="publisher" content="Homepage-Webhilfe" />
13      </head>
14
15      <body>
16        <xsl:apply-templates />
17      </body>
18    </html>
19  </xsl:template>
20
21  <xsl:template match="zahlen">
22    <ul>
23      <xsl:apply-templates />
24    </ul>
25  </xsl:template>
26
27  <xsl:template match="zahl">
28    <li><xsl:value-of select="." /></li>
29  </xsl:template>
30 </xsl:stylesheet>

```

XML: 

HTML: 

Benötige ich dann gar keine Schleifen? Prinzipiell gesehen, können Schleifen immer durch die Template-Technik ersetzt werden, in einigen Fällen ist jedoch die Verwendung von Templates wesentlich umständlicher. Vorteile von Templates sind jedoch die **Wiederverwendbarkeit** sowie der **klare Verlauf bei der Abarbeitung der XML-Daten** (von Template, zu Template, zu Template, ...). Schleifen haben hingegen den Vorteil, dass sie in XSLT-Dokumenten vom Programmierer **leichter gelesen** werden können. Dies ist vor allem von Vorteil, wenn Ihr XSLT-Dokument sehr komplex ist. Letzten Endes ist es natürlich auch wieder „Geschmackssache“. Im Allgemeinen ist jedoch die **Verwendung von Templates in XSLT zu bevorzugen**.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » Nummerierung und Sortierung

## Nummerierung und Sortierung

### Inhalt dieser Seite:

1. Nummerierung
2. Sortierung

XSLT bietet zwei Elemente an, um auf einfache Art und Weise Knoten fortlaufend zu nummerieren und /oder zu sortieren. In den folgenden beiden Abschnitten werden wir auf die Nummerierung und Sortierung von Knoten separat eingehen.

### Nummerierung

Das leere XSLT-Element `number` erlaubt es, eine **fortlaufende Nummerierung** zu erzeugen. Das Element kann dabei direkt innerhalb eines Templates oder aber auch innerhalb einer Schleife vorkommen. Das wichtigste Attribut ist `format`, in welchem das **Nummernformat** angegeben wird. Dabei können als Werte die Präfixe `1` (für 1, 2, 3, 4, 5, ...), `i` (für i, ii, iii, iv, v, ...), `I` (für I, II, III, IV, V, ...), `a` (für a, b, c, d, e, ...) und `A` (für A, B, C, D, E, ...) angegeben werden. Möchten Sie z. B. eine **zweistellige Nummerierung**, so können Sie `01` (für 01, 02, 03, 04, 05, ..., 10, 11, 12, 13, 14, 15, ...) angeben. Auch die Eingabe von Leerzeichen oder weiteren Zeichen wie `.` oder `)` sind möglich. Das Attribut `value` erlaubt das Festlegen eines „manuellen“ Werts. Verwendet wird dies immer dann, wenn das `number`-Element nur zur **Formatierung einer einzelnen Zahl** genutzt werden soll. Weitere optionale Attribute sind unter anderem `grouping-separator` (Trennzeichen für die Zifferngruppierung), `grouping-size` (Zifferanzahl für die Zifferngruppierung) und `count` (Knoten, welche nummeriert werden sollen bzw. bei welchen die Nummerierung fortläuft, Angabe als XPath-Ausdruck).

#### XML-Code:


```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="nummerieren.xsl" type="text/xsl" ?>
3
4 <zahlen>
5   <zahl>45</zahl>
6   <zahl>39</zahl>
7   <zahl>22</zahl>
8   <zahl>70</zahl>
9   <zahl>68</zahl>
10  <zahl>12</zahl>
11  <zahl>51</zahl>
12  <zahl>63</zahl>
13  <zahl>27</zahl>
14  <zahl>94</zahl>
15 </zahlen>
    
```

#### XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>
6       <head>
7         <title>Nummerierung - XSLT Code-Beispiel</title>
8
9         <meta charset="utf-8" />
10
11        <meta name="robots" content="noindex,nofollow" />
12        <meta name="publisher" content="Homepage-Webhilfe" />
13      </head>
14
15      <body>
16        <xsl:for-each select="/zahlen/zahl">
17          <span><xsl:number format="1. " /><xsl:value-of select="." /></span><br />
18        </xsl:for-each>
19      </body>
20    </html>
21  </xsl:template>
22 </xsl:stylesheet>
    
```

XML:   
 HTML: 

### Sortierung

Um eine Reihe an Knoten zu sortieren, gibt es das einteilige Element `sort`. Mit Hilfe des Attributs `select` können Sie festlegen, an Hand welcher Knoten die Sortierung erfolgen soll. Wird das Attribut weggelassen, so werden die „aktuellen“ Knoten sortiert. Um festzulegen, ob die Sortierung **auf- oder absteigend** erfolgen soll, können Sie das Attribut `order` mit den Werten `ascending` (aufsteigend, Voreinstellung) oder `descending` (absteigend) verwenden. Das `data-type`-Attribut ermöglicht es, festzulegen, dass die Sortierung an Hand **numerischer Werte** (`number`) und nicht an Hand von zeichenbasierten Werten (`text`, Standardeinstellung) erfolgen soll. Bei der Sortierung von Text kann mit Hilfe des Attributs `case-order` festgelegt werden, ob Großbuchstaben vor Kleinbuchstaben (`upper-first`) oder Kleinbuchstaben vor Großbuchstaben (`lower-first`) kommen. Das `sort`-Element kann innerhalb einer Schleife (`for-each`-Element) oder innerhalb des `apply-templates`-Element (Notation erfolgt dann zweiteilig) vorkommen. Wird das Element innerhalb von `apply-templates` verwendet, so sollte bei diesem das `select`-Attribut angegeben werden.

#### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="sortieren.xsl" type="text/xsl" ?>
    
```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
 Krummstraße 9/3  
 73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
 E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » Nummerierung und Sortierung

```

3
4 <zahlen>
5   <zahl>45</zahl>
6   <zahl>39</zahl>
7   <zahl>22</zahl>
8   <zahl>70</zahl>
9   <zahl>68</zahl>
10  <zahl>12</zahl>
11  <zahl>51</zahl>
12  <zahl>63</zahl>
13  <zahl>27</zahl>
14  <zahl>94</zahl>
15 </zahlen>
    
```

## XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>
6       <head>
7         <title>Sortierung - XSLT Code-Beispiel</title>
8
9         <meta charset="utf-8" />
10
11        <meta name="robots" content="noindex,nofollow" />
12        <meta name="publisher" content="Homepage-Webhilfe" />
13      </head>
14
15      <body>
16        <ul>
17          <xsl:for-each select="/zahlen/zahl">
18            <xsl:sort data-type="number" order="descending" />
19            <li><xsl:value-of select="." /></li>
20          </xsl:for-each>
21        </ul>
22      </body>
23    </html>
24  </xsl:template>
25 </xsl:stylesheet>
    
```

XML:   
HTML: 

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » Ausgabe im Ergebnisbaum

## Ausgabe im Ergebnisbaum

Bisher haben wir die Ausgabe des Ereignisbaums durch die „direkte Notation“ der Elemente, Attribute und Textknoten gesteuert. Es gibt dafür aber auch einen anderen Weg. Auf diesen und auf weitere Möglichkeiten zur Steuerung der Ausgabe wollen wir in diesem Thema genauer eingehen.

### Inhalt dieser Seite:

1. Elemente
2. Attribute
3. Texte
4. Kommentare
5. Baum kopieren
6. Ausgabe steuern

## Elemente

Um in der Ausgabe ein Element zu platzieren, haben wir das Element bisher immer direkt angegeben. Alternativ können wir ein Element jedoch auch über das XSLT-Element `element` erzeugen. Als Attribute stehen uns `name` für den Elementnamen und `namespace` für den Namensraum des Elements zur Verfügung. Ein weiteres Attribut (`use-attribute-sets`) wird in Verbindung mit Attributsätzen verwendet. Dazu jedoch im [nächsten Abschnitt](#) mehr. Verwendet wird diese Art der Element-Erzeugung i. d. R. nur bei **Verwendung von Attributsätzen** oder bei der **Erzeugung einer weiteren XSL-Datei** (mit dem gleichen Namensraumpräfix, wie die eigene Datei).

### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="elemente.xml" type="text/xsl" ?>
3
4 <artikel>
5   <titel>Grundlagen von XSLT</titel>
6   <text>Hier lernen Sie die Grundlagen von XSLT ...</text>
7 </artikel>



```

### XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>
6       <head>
7         <title>Elemente erzeugen - XSLT Code-Beispiel</title>
8
9         <meta charset="utf-8" />
10
11        <meta name="robots" content="noindex,nofollow" />
12        <meta name="publisher" content="Homepage-Webhilfe" />
13      </head>
14
15      <body>
16        <xsl:element name="artikel">
17          <xsl:element name="h1"><xsl:value-of select="/artikel/titel" /></xsl:element>
18          <xsl:element name="p"><xsl:value-of select="/artikel/text" /></xsl:element>
19        </xsl:element>
20      </body>
21    </html>
22  </xsl:template>
23 </xsl:stylesheet>

```

XML:   
HTML: 

## Attribute

Die Erzeugung von Attributen ist mittels des XSLT-Elements `attribute` möglich. Der Vorteil dieser Variante gegenüber der direkten Notation ist, dass die **Notation an einer beliebigen Stelle** innerhalb des Elements erfolgen kann und der **Wert variabel** (also auch aus den XML-Daten stammend) sein kann. Auch das Einschließen des Attribut-Elements in eine Abfrage ist möglich. Wie auch beim `element`-Element gibt es beim `attribute`-Element die Attribute `name` und `namespace`.

### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="attribute.xml" type="text/xsl" ?>
3
4 <artikel>
5   <titel zentriert="ja">Grundlagen von XSLT</titel>
6   <text>
7     ...
8   </text>
9 </artikel>

```

### XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe


Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » Ausgabe im Ergebnisbaum

```

6      <head>
7          <title>Attribute erzeugen - XSLT Code-Beispiel</title>
8
9          <meta charset="utf-8" />
10
11         <meta name="robots" content="noindex,nofollow" />
12         <meta name="publisher" content="Homepage-Webhilfe" />
13     </head>
14
15     <body>
16         <article>
17             <h1>
18                 <xsl:if test="/artikel/titel/@zentriert = 'ja'">
19                     <xsl:attribute name="style">text-align: center;</xsl:attribute>
20                 </xsl:if>
21                 <xsl:value-of select="/artikel/titel" />
22             </h1>
23             <p><xsl:value-of select="/artikel/text" /></p>
24         </article>
25     </body>
26 </html>
27 </xsl:template>
28 </xsl:stylesheet>

```

XML: 

HTML: 

Um mehrere **Attribute zu gruppieren**, können Sie das XSLT-Element `attribute-set` verwenden. Man spricht hier von einem **Attributsatz**. Attributsätze müssen außerhalb von Templates und somit direkt innerhalb des XSLT-Wurzelements notiert werden. Über das Attribut `name` können Sie einen eindeutigen Namen des Attributsatzes festlegen. Der festgelegte Name kann dann innerhalb des Attributs `use-attribute-sets` in den Elementen `element` und `copy` oder einem weiteren `attribute-set`-Element verwendet werden. Sollen mehrere Attributsätze verwendet werden, so sind diese per Leerzeichen zu trennen.

#### XML-Code:

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2  <?xml-stylesheet href="attribut-satz.xsl" type="text/xsl" ?>
3
4  <artikel>
5      <titel>Grundlagen von XSLT</titel>
6      <text>Hier lernen Sie die Grundlagen von XSLT ...</text>
7  </artikel>

```

#### XSLT-Code:

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4      <xsl:attribute-set name="box">
5          <xsl:attribute name="style">margin: 5px; border: 1px dashed red; padding: 5px;</xsl:attribute>
6      </xsl:attribute-set>
7
8      <xsl:template match="/">
9          <html>
10             <head>
11                 <title>Attributsätze - XSLT Code-Beispiel</title>
12
13                 <meta charset="utf-8" />
14
15                 <meta name="robots" content="noindex,nofollow" />
16                 <meta name="publisher" content="Homepage-Webhilfe" />
17             </head>
18
19             <body>
20                 <xsl:element name="article" use-attribute-sets="box">
21                     <h1><xsl:value-of select="/artikel/titel" /></h1>
22                     <p><xsl:value-of select="/artikel/text" /></p>
23                 </xsl:element>
24
25                 <br />
26
27                 <xsl:element name="div" use-attribute-sets="box">
28                     <b>Übrigens:</b> Auf unserer Website gibt es ein ausführliches Tutorial zu XML und XSLT.
29                 </xsl:element>
30             </body>
31         </html>
32     </xsl:template>
33 </xsl:stylesheet>

```

XML: 

HTML: 

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » [Ausgabe im Ergebnisbaum](#)

## Texte

Möchten Sie einen Textknoten im Ergebnisbaum erzeugen, so können Sie das Element `text` verwenden. Bei dieser Variante **bleiben Leerraumzeichen erhalten** und werden „unverändert“ in die Ausgabe übertragen. Bisher haben wir „Text“ immer direkt notiert. Es wäre jedoch keinesfalls falsch, dort das `text`-Element zu verwenden. Verwendet wird das Element in der Realität jedoch meistens nur, um ein einzelnes Leerzeichen in die Ausgabe zu übertragen.

### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="texte.xsl" type="text/xsl" ?>
3
4 <dokument>
5   <titel>Text-Knoten erzeugen</titel>
6 </dokument>


```

### XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>
6       <head>
7         <title>Texte erzeugen - XSLT Code-Beispiel</title>
8
9         <meta charset="utf-8" />
10
11        <meta name="robots" content="noindex,nofollow" />
12        <meta name="publisher" content="Homepage-Webhilfe" />
13      </head>
14
15      <body>
16        <h1><xsl:value-of select="/dokument/titel" /></h1>
17        <i><xsl:text>Dieser Text-Knoten wurde mit xsl:text erzeugt.</xsl:text></i>
18      </body>
19    </html>
20  </xsl:template>
21 </xsl:stylesheet>

```

XML:   
HTML: 

## Kommentare

Wenn Sie innerhalb eines XSLT-Skripts einen Kommentar (also zwischen den Tags `<!--` und `-->`) platzieren, werden Sie feststellen, dass dieser Kommentar nicht im Ausgabebaum erscheint. Dies kann in einigen Situationen erwünscht sein, in anderen hingegen nicht. Möchten Sie einen **Kommentar im Ergebnisbaum** erzeugen, so müssen Sie das XSLT-Element `comment` verwenden.

### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="kommentare.xsl" type="text/xsl" ?>
3
4 <dokument>
5   <titel>Kommentar-Knoten erzeugen</titel>
6 </dokument>

```

### XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>
6       <head>
7         <title>Kommentare erzeugen - XSLT Code-Beispiel</title>
8
9         <meta charset="utf-8" />
10
11        <meta name="robots" content="noindex,nofollow" />
12        <meta name="publisher" content="Homepage-Webhilfe" />
13      </head>
14
15      <body>
16        <h1><xsl:value-of select="/dokument/titel" /></h1>
17        <xsl:comment>Dieser Kommentar-Knoten wurde mit xsl:comment erzeugt.</xsl:comment>
18      </body>
19    </html>
20  </xsl:template>
21 </xsl:stylesheet>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » [Ausgabe im Ergebnisbaum](#)

XML:   
HTML:

## Baum kopieren

Um ein **Element von der XML-Datei in die Ausgabedatei zu kopieren**, können Sie das XSLT-Element `copy` verwenden. Beim Kopiervorgang werden die Tags des Elements sowie der Textinhalt (PCDATA) kopiert. Attribute und untergeordnete Elemente werden nicht kopiert. Innerhalb des `copy`-Elements wird i. d. R. mit dem `value-of`-Element ein Knoten (also der zu kopierende Knoten) selektiert. Des Weiteren können Sie mit Hilfe des Attributs `use-attribute-set` dem Element einen Attributsatz zuweisen.

### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="kopieren.xsl" type="text/xsl" ?>
3
4 <dokument>
5   <h1>Knoten kopieren</h1>
6 </dokument>

```

### XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>
6       <head>
7         <title>Knoten kopieren - XSLT Code-Beispiel</title>
8
9         <meta charset="utf-8" />
10
11        <meta name="robots" content="noindex,nofollow" />
12        <meta name="publisher" content="Homepage-Webhilfe" />
13      </head>
14
15      <body>
16        <xsl:apply-templates />
17      </body>
18    </html>
19  </xsl:template>
20
21  <xsl:template match="/dokument/h1">
22    <xsl:copy><xsl:value-of select="/dokument/h1" /></xsl:copy>
23  </xsl:template>
24 </xsl:stylesheet>

```

XML:   
HTML:

Um einen **ganzen XML-Baum** zu kopieren, können Sie das einteilige XSLT-Element `copy-of` verwenden. Das `copy-of`-Element kopiert das selektierte Element, inkl. den Attributen sowie allen untergeordneten Knoten (Elementknoten, Textknoten etc.). Den zu kopierenden Knoten legen Sie mit dem `select`-Attribut fest.

### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="kopieren-baum.xsl" type="text/xsl" ?>
3
4 <dokument>
5   <article>
6     <h1>Grundlagen von XSLT</h1>
7     <p>Hier lernen Sie die Grundlagen von XSLT ...</p>
8   </article>
9 </dokument>

```

### XSLT-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
5     <html>
6       <head>
7         <title>Baum kopieren - XSLT Code-Beispiel</title>
8
9         <meta charset="utf-8" />
10
11        <meta name="robots" content="noindex,nofollow" />
12        <meta name="publisher" content="Homepage-Webhilfe" />
13      </head>
14
15      <body>

```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » [Ausgabe im Ergebnisbaum](#)

```

16 |         <xsl:copy-of select="/dokument/article" />
17 |     </body>
18 | </html>
19 | </xsl:template>
20 | </xsl:stylesheet>

```

XML:   
HTML: 

## Ausgabe steuern



Um ein paar Einstellungen für die Ausgabe bzw. Transformation festzulegen, können Sie das Element `output` verwenden. Zur Konfiguration stehen einige Attribute zur Verfügung:

<b>method</b>	Legt die Art der Ausgabe (xml, html oder text) fest.
<b>media-type</b>	Legt den MIME-Typ für die Ausgabe fest.
<b>indent</b>	Gibt an, ob untergeordnete Elemente in der Ausgabe eingerückt (yes) werden sollen oder nicht (no).
<b>version</b>	Legt die Version, welche in der XML-Deklaration verwendet wird, fest.
<b>encoding</b>	Legt die Zeichenkodierung für den Ergebnisbaum (wird u. a. für die XML-Deklaration verwendet) fest.
<b>standalone</b>	Legt die Einstellung für den DTD-Bezug (yes oder no) in der XML-Deklaration fest (Vergleich standalone-Attribut in XML).
<b>omit-xml-declaration</b>	Gibt an, ob die XML-Deklaration weggelassen (yes, Standard) werden soll oder nicht (no).

Das `output`-Element bzw. dessen Einstellungen werden in den XSLT-Prozessoren von Browsern **unzureichend unterstützt**. Die meisten XSLT-Prozessoren von Browsern sind nur für die Transformation in HTML geeignet.

### XML-Code:

```

1 | <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 | <?xml-stylesheet href="ausgabe.xsl" type="text/xsl" ?>
3 |
4 | <adressbuch>
5 |   <kontakt geschlecht="m" alter="17">
6 |     <name>Max Mustermann</name>
7 |     <strasse>Musterstraße 123</strasse>
8 |     <ort>12345 Musterstadt</ort>
9 |     <email>max.mustermann@example.com</email>
10 |   </kontakt>
11 |   <kontakt geschlecht="w" alter="24">
12 |     <name>Maria Musterfrau</name>
13 |     <strasse>Musterstraße 456</strasse>
14 |     <ort>67890 Musterhafen</ort>
15 |     <email>m.musterfrau@example.com</email>
16 |   </kontakt>
17 |   <kontakt geschlecht="m" alter="53">
18 |     <name>Peter Müller</name>
19 |     <strasse>Hauptstraße 29</strasse>
20 |     <ort>27946 Programmierstadt</ort>
21 |     <email>petra-mueller@example.com</email>
22 |   </kontakt>
23 | </adressbuch>

```


### XSLT-Code:

```

1 | <?xml version="1.0" encoding="UTF-8" ?>
2 |
3 | <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4 |   <xsl:output method="xml" omit-xml-declaration="no" version="1.0" encoding="UTF-8" standalone="yes" indent="yes" />
5 |
6 |   <xsl:template match="/">
7 |     <mailliste>
8 |       <xsl:apply-templates />
9 |     </mailliste>
10 |   </xsl:template>
11 |
12 |   <xsl:template match="kontakt">
13 |     <adresse><xsl:value-of select="email" /></adresse>
14 |   </xsl:template>
15 | </xsl:stylesheet>

```

XML (Eingabe): 

XML (Ausgabe): 

**Wichtig:** Beachten Sie, dass wenn Sie auf das Vorschau-Icon „XML (Eingabe)“ klicken, Sie vermutlich kein richtiges Ergebnis sehen werden, da Ihr Browser die

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

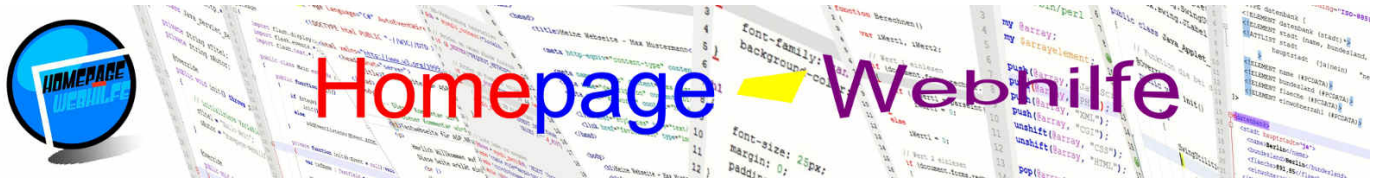
#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » Ausgabe im Ergebnisbaum

Transformation von XML zu XML nicht unterstützt.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

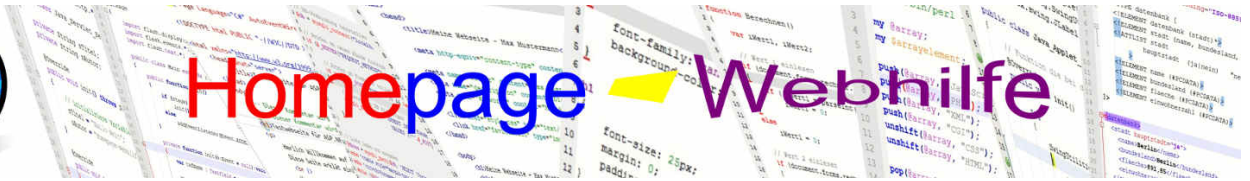
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislungen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » [Stylesheet-Einbindung](#)

## Stylesheet-Einbindung

### Inhalt dieser Seite:

1. Import
2. Inkludierung

Für große XSLT-Skripte oder -Projekte kann es sinnvoll sein, **Templates in weitere Dateien auszulagern**. Dadurch werden die einzelnen Skripte nicht nur kleiner und übersichtlicher, sondern auch wiederverwendbar, d. h. Sie können ein XSLT-Skript von mehreren anderen XSLT-Skripten einbinden. In XSLT gibt es zwei Mechanismen für solche Vorhaben: den Import und die Inkludierung.

### Import

Um eine Datei zu importieren, müssen Sie das XSLT-Element `import` angeben. Dieses muss sich dabei direkt im `stylesheet`-Element befinden. Das leere Element `import` verfügt über das Attribut `href`, mit welchem Sie den Pfad zur importierenden Datei angeben. In der importierten Datei müssen Sie nichts Weiteres angeben. Sie notieren dort ebenfalls ganz normal Templates. Um von der „Hauptdatei“, die in der importierten Datei notierten Templates anzuwenden, verwenden Sie auch ganz normal das `apply-templates`-Element. Wenn Sie in der „Hauptdatei“ und in der importierten Datei ein Template notiert haben, welches über die gleiche Bedingung verfügt, so haben Sie einen **Template-Konflikt**. Dies ist jedoch nicht weiter schlimm. Das Template der Hauptdatei hat prinzipiell immer Vorrang. Möchten Sie explizit das Template von der importierten Datei aufrufen oder möchten Sie aus dem Template in der Hauptdatei noch zusätzlich das Template der importierten Datei anwenden, so können Sie das XSLT-Element `apply-imports` notieren. Das Element ist einteilig und verfügt über keine Attribute.

#### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="import.xml" type="text/xsl" ?>
3
4 <artikel>
5   <titel>Grundlagen von XSLT</titel>
6   <text>Hier lernen Sie die Grundlagen von XSLT ...</text>
7 </artikel>
    
```

#### XSLT-Code (import.xml):

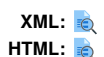
```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:import href="import-erw.xml" />
5
6   <xsl:template match="/">
7     <html>
8       <head>
9         <title>Stylesheet importieren - XSLT Code-Beispiel</title>
10
11        <meta charset="utf-8" />
12
13        <meta name="robots" content="noindex,nofollow" />
14        <meta name="publisher" content="Homepage-Webhilfe" />
15      </head>
16
17      <body>
18        <xsl:apply-imports />
19      </body>
20    </html>
21  </xsl:template>
22 </xsl:stylesheet>
    
```

#### XSLT-Code (import-erw.xml):

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="artikel">
5     <article>
6       <xsl:apply-templates />
7     </article>
8   </xsl:template>
9
10  <xsl:template match="titel">
11    <h1><xsl:value-of select="." /></h1>
12  </xsl:template>
13
14  <xsl:template match="text">
15    <p><xsl:value-of select="." /></p>
16  </xsl:template>
17 </xsl:stylesheet>
    
```



**Übrigens:** Das hier verwendete Verfahren kommt Ihnen vielleicht von der Objektorientierung bekannt vor, wo Methoden überschrieben werden können.

## Inkludierung

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSLT](#) » [Stylesheet-Einbindung](#)

Bei der Inkludierung werden die „Hauptdatei“ und die inkludierte Datei verschmolzen bzw. vereint. Um ein Stylesheet in das Dokument zu inkludieren, verwenden Sie das einteilige Element `include`. Dieses muss ebenfalls direkt im Wurzelement des XSLT-Dokuments notiert werden. Über das `href`-Attribut wird der Pfad zur inkludierten Datei festgelegt. Um die Templates aus der inkludierten Datei anzuwenden, verwenden Sie weiterhin wie bekannt das `apply-templates`-Element.

**XML-Code:**

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="include.xml" type="text/xsl" ?>
3
4 <artikel>
5   <titel>Grundlagen von XSLT</titel>
6   <text>Hier lernen Sie die Grundlagen von XSLT ...</text>
7 </artikel>
    
```

**XSLT-Code (include.xml):**

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:include href="include-erw.xml" />
5
6   <xsl:template match="/">
7     <html>
8       <head>
9         <title>Stylesheet inkludieren - XSLT Code-Beispiel</title>
10
11        <meta charset="utf-8" />
12
13        <meta name="robots" content="noindex,nofollow" />
14        <meta name="publisher" content="Homepage-Webhilfe" />
15      </head>
16
17      <body>
18        <xsl:apply-templates />
19      </body>
20    </html>
21  </xsl:template>
22 </xsl:stylesheet>
    
```

**XSLT-Code (include-erw.xml):**

```

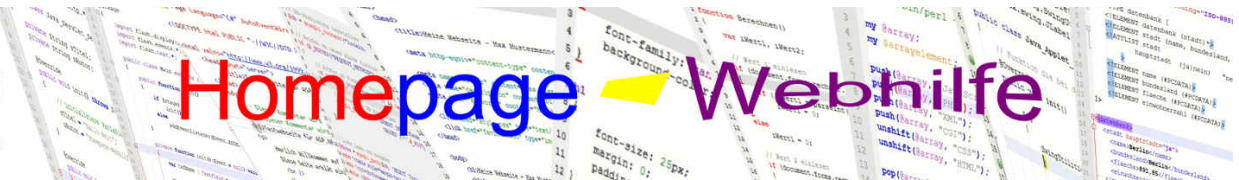
1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="artikel">
5     <article>
6       <xsl:apply-templates />
7     </article>
8   </xsl:template>
9
10  <xsl:template match="titel">
11    <h1><xsl:value-of select="." /></h1>
12  </xsl:template>
13
14  <xsl:template match="text">
15    <p><xsl:value-of select="." /></p>
16  </xsl:template>
17 </xsl:stylesheet>
    
```

XML:

HTML:

**Wichtig:** Wenn bei der Inkludierung ein Konflikt entsteht (also zwei Templates, welche die gleiche Bedingung haben), so kann keine Transformation erfolgen. Beim Import hingegen gibt es keinen Fehler (siehe Beschreibung oben).

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Grundlagen](#)

## Grundlagen

XSL-FO (XSL Formatting Objects) ist eine Auszeichnungssprache, die es ermöglicht, eine **XML-Datei zu formatieren**. Da XSL-FO nicht auf die Daten einer XML-Datei zugreifen kann, wird XSL-FO i. d. R. mit XSLT (und XPath für die Adressierung) gemischt. XSL-FO ist im Gegensatz zu XSLT also weniger zur Transformation, sondern eher zur Formatierung gedacht.

Die Sprache XSL-FO ist XML basierend und wird in den meisten Fällen dazu genutzt, **Ausgaben, die für den Druck bestimmt sind, zu erzeugen**. Wie auch bei XSLT, ist ein sogenannter Prozessor (bei XSL-FO ein (XSL-)FO-Prozessor) erforderlich, um die „Konvertierung“ durchzuführen. Je nach verwendetem Prozessor sind unterschiedliche Ausgabeformate möglich. Geläufig sind unter anderem die **Formate PDF, PostScript und RTF**.

### Inhalt dieser Seite:

1. Funktionsweise
2. Maßeinheiten
3. Grundaufbau
4. Blöcke
5. Erstes Beispiel

## Funktionsweise

Das in einer XSL-FO-Datei notierte Skript enthält **XSL-FO-Elemente, die den Aufbau und Inhalt einer oder mehrerer Seiten beschreibt**. Auf den [Grundaufbau](#) sowie das [Seitenlayout](#) gehen wir später noch genauer ein. XSL-FO-Skripte werden in Dateien mit der Endung `.xsl` oder `.fo` gespeichert. Als MIME-Typ wird i. d. R. `text/xsl` verwendet.

Wie bereits oben erwähnt, wird im Regelfall **XSL-FO in Kombination mit XSLT verwendet**. Es gibt dann also ein XML-Dokument, welches die „Daten“ für die Ausgabe enthält, und ein XSLT-Skript, welches als Ausgabe ein XSL-FO-Skript (inkl. den Daten) erzeugt (und nicht wie üblich ein HTML-Dokument). Das vom XSLT-Prozessor erzeugte XSL-FO-Skript kann dann mit Hilfe eines XSL-FO-Prozessors in eine andere Datei (z. B. in eine PDF-Datei) transformiert werden. Natürlich sollte das XML-Dokument als erstes noch mit dem XSL-Skript per Verarbeitungshinweis verlinkt werden:



```
1 | <?xml-stylesheet href="stylesheet.xsl" type="text/xsl" ?>
```

Es wäre jedoch auch möglich, über einen anderen Weg (z. B. mittels einer serverseitigen Skriptsprache wie PHP) ein XSL-FO-Skript (inkl. dem „Inhalt“) zu erzeugen, um dann lediglich die Formatierung durch einen XSL-FO-Prozessor durchführen zu lassen.

Anders als CSS beschreibt XSL-FO nicht nur den Aufbau und den Stil eines Dokuments, sondern auch zugleich den Inhalt. Man könnte XSL-FO also als Kombination aus HTML und CSS sehen (natürlich mal abgesehen davon, dass XSL-FO für Druckausgaben konzipiert ist). Auf Grund dieser Tatsache ist ein XSL-FO-Skript nur dann wiederverwendbar, wenn es durch einen vorhergehenden Transformationsvorgang (wie z. B. mittels XSLT) erzeugt wurde.

Für den Transformations- bzw. Formatierungsvorgang wird ein sogenannter **XSL-FO-Prozessor oder kurz FO-Prozessor** benötigt. Solche Prozessoren sind in der Regel kostenpflichtig, es gibt jedoch auch ein paar **kostenlose**. Die kostenlosen Prozessoren sind jedoch **teilweise in der Funktionalität und Unterstützung begrenzt**.

Ein kostenloser XSL-FO-Prozessor, welchen auch wir für unsere Beispiele verwendet haben, ist **Apache FOP** von der [Apache Software Foundation](#). FOP verfügt zwar immer noch über ein paar Probleme und unterstützt noch nicht den vollen Funktionsumfang von XSL-FO, ist jedoch alles in allem schon relativ gut und für einfache Zwecke mehr als ausreichend. FOP ist ein Java-Programm und kann entweder **in ein eigenes Java-Projekt integriert werden oder über eine im Download-Paket enthaltene bat-Datei ausgeführt werden**.

Apache FOP enthält neben dem eigentlichen FO-Prozessor auch einen XSLT-Prozessor, wodurch nur ein Transformationsvorgang notwendig ist, wenn aus einer XML- und XSL-Datei eine PDF-Datei (oder ein anderes Dokument) erzeugt werden soll. Dem FOP werden i. d. R. die Parameter `-xml`, `-xsl` und (bei einer Transformation in ein PDF-Dokument) `-pdf` mitgegeben. Das erste Beispiel (siehe [Abschnitt Erstes Beispiel](#)) wurde mit folgendem Kommando erzeugt:

```
1 | fop.bat -xml erstes-beispiel.xml -xsl erstes-beispiel.xsl -pdf erstes-beispiel.pdf
```

Liegt als Grundlage nur ein reines XSL-FO-Skript vor (also ohne XML-Datei und XSLT-Skript), so werden die Parameter `-xml` und `-xsl` weggelassen und stattdessen der Parameter `-fo` verwendet. Ein Aufruf des `fop.bat`-Skripts könnte folgendermaßen aussehen:

```
1 | fop.bat -fo schriftfarbe.xsl -pdf schriftfarbe.pdf
```

## Maßeinheiten

Für die Größenangabe von Blöcken, Grafiken, Schriften etc. stehen verschiedene Maßeinheiten zur Verfügung. Dabei ist es natürlich auch möglich, **verschiedene Maßeinheiten in einem Dokument zu mischen**. Gerade bei Druckmedien werden (z. B. für Größenangaben von Seiten oder Tabellenspalten) vor allem die Maßeinheiten `cm` (Zentimeter), `mm` (Millimeter) und `in` (Inch) von Bewandnis sein. Bei Schriftgrößen oder ähnlichem wird vermutlich aber doch eher auf die Maßeinheiten `pt` (Punkte), `pc` (Picas) und `px` (Pixel) zurückgegriffen. Eine weitere Einheit ist `em`, welche die Höhe der Schrift repräsentiert und somit oft für Abstände verwendet wird.

## Grundaufbau

Das Wurzelement bei XSL-FO ist `root`, in welchem i. d. R. nur ein Attribut für den Namensraum spezifiziert wird. Als Namensraumpräfix wird üblicherweise `fo` verwendet. Da es sich bei einer XSL-FO-Datei auch um eine XML-Datei handelt, wird zudem in den meisten Fällen die XML-Deklaration am Dateianfang notiert.

```

1 | <?xml version="1.0" encoding="UTF-8" ?>
2 |
3 | <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4 |
5 | </fo:root>
  
```

<b>Über uns</b> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<b>Community</b> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<b>Nachschlagewerk</b> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
---	--	---	---



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » XML & Co. » XSL-FO » Grundlagen

## Blöcke

Die wichtigsten Bestandteile eines XSL-FO-Skripts sind die sogenannten Blöcke. Blöcke zeichnen sich durch das Element `block` aus und bilden im Ausgabedokument einen **Absatz**. Innerhalb des `block`-Elements können Sie nun z. B. einen Text notieren. Die direkte Notation von Text oder einigen anderen Inhalten (also ohne das `block`-Element) ist nicht gültig und führt zu einem Transformationsfehler.

```
1 <fo:block>
2   Hier steht Ihr Inhalt!
3 </fo:block>
```

In einem `block`-Element ist die Angabe einer Größe (also einer Breite und / oder Höhe) nicht möglich. Ist dies zwingend erforderlich, so muss das `block-container`-Element verwendet werden. Im `block-container`-Element können `block`-Elemente notiert werden. Das `block-container`-Element verfügt über die Attribute `width` und `height`, um die **Breite und Höhe festzulegen**.

```
1 <fo:block-container width="5cm" height="2cm">
2   <fo:block>
3     Hier steht Ihr Inhalt!
4   </fo:block>
5 </fo:block-container>
```

Ein Block erstreckt sich (sofern nicht das `block-container`-Element mit einer Größenangabe verwendet wird) immer über die komplett verfügbare Breite und über die Höhe des Inhalts. Das `block`-Element kann man also mit dem `div`-Element von HTML vergleichen. Für Inhalte, die sich **nur über die Breite des Inhalts erstrecken sollen** (also wie beim HTML-Element `span`), können Sie in XSL-FO das Element `inline` (zu Deutsch inzeilig) verwenden. Dieses Element muss innerhalb eines `block`-Elements vorkommen und wird im Regelfall zu **Formatierungszwecken** (z. B. zum Hervorheben eines Worts in einem Text) verwendet.

```
1 <fo:block>
2   Hier steht Ihr Inhalt mit <fo:inline font-weight="bold">Hervorhebung</fo:inline>
3 </fo:block>
```

## Erstes Beispiel

Nun ist es aber Zeit für das erste Beispiel: Im folgenden Beispiel verwenden wir ein XSLT-Skript, welches mit Hilfe einer XML-Datei ein XSL-FO-Skript erzeugt (so wie es in der Praxis üblich ist). Im Wurzel-Template (`match="/"`) wird das **Grundgerüst von XSL-FO** erzeugt. Das Element `layout-master-set` und dessen Unterelemente legen die **Vorlagen** für die später verwendeten Seiten fest. Innerhalb der Unterelemente des `page-sequence`-Elements werden die **Folgen von Seiten und auch der eigentliche Inhalt** der Seiten angegeben. Auf diese und weitere Elemente gehen wir jedoch im nächsten Thema noch genauer ein. Innerhalb des `flow`-Elements erkennen Sie ein `block`-Element, in welchem der Inhalt des `text`-Elements aus der XML-Datei mittels des XSLT-Elements `value-of` platziert wird.

### XML-Code:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="erstes-beispiel.xsl" type="text/xsl" ?>
3
4 <text>
5   Dies ist das erste Beispiel zu XSL-FO.
6 </text>
```

### XSL-Code:

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:fo="http://www.w3.org/1999/XSL/Format">
4   <xsl:output method="xml" omit-xml-declaration="no" version="1.0" encoding="UTF-8" indent="yes" />
5
6   <xsl:template match="/">
7     <fo:root>
8       <fo:layout-master-set>
9         <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
10          <fo:region-body region-name="inhalt" margin="2cm" />
11        </fo:simple-page-master>
12      </fo:layout-master-set>
13
14      <fo:page-sequence master-reference="DIN-A4">
15        <fo:flow flow-name="inhalt">
16          <fo:block>
17            <xsl:value-of select="/text" />
18          </fo:block>
19        </fo:flow>
20      </fo:page-sequence>
21    </fo:root>
22  </xsl:template>
23 </xsl:stylesheet>
```

PDF: 

**Wichtig:** Im XSLT-Tutorial haben Sie kennengelernt, dass die meisten Browser über einen integrierten XSLT-Prozessor verfügen, um somit XML-Dokumente in HTML umzuwandeln. Jedoch enthält keiner der gängigen Browser einen XSL-FO-Prozessor, um somit z. B. eine dynamische Erzeugung eines PDF-Dokuments durch den Browser zu ermöglichen. Daher wurden alle unsere Beispiele mit dem Apache FOP von uns bereits in eine PDF-Datei transformiert, die Sie über das Vorschaufenster anschauen können.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Seitenlayout](#)

## Seitenlayout

**Inhalt dieser Seite:**

1. Seitenaufbau
2. Seitenfolgen
3. Seitenumbruch

Im vorherigen Thema haben Sie ja bereits das Wurzelement `root` von XSL-FO kennengelernt. In diesem Thema wollen wir uns nun mit den untergeordneten Elementen und dem Seitenlayout beschäftigen. Dem `root`-Element muss exakt ein `layout-master-set`-Element und ein oder mehrere `page-sequence`-Element(e) untergeordnet werden.

Dem `layout-master-set` werden ein oder mehrere Elemente des Typs `simple-page-master` und / oder `page-sequence-master` untergeordnet. Das `simple-page-master`-Element wird dazu verwendet, das **Format und die Bereiche einer Seite** zu definieren. Mehr dazu im [Abschnitt Seitenaufbau](#). Das `page-sequence-master`-Element wird zur **Definition von Seitenfolgen** (also von sich wiederholenden Seiten) verwendet. Darauf werden wir jedoch im [2. Abschnitt](#) noch genauer eingehen.

Das Element `page-sequence` wird dazu genutzt, die „eigentlichen“ Seiten bzw. Seitensequenzen an Hand der vorliegenden Vorlagen **im Ausgabedokument zu platzieren und mit Inhalt zu füllen**. Um einer Seitensequenz eine Vorlage zuzuweisen, wird das Attribut `master-reference` im `page-sequence`-Element verwendet. Die wichtigsten Unterelemente von `page-sequence` sind `static-content` und `flow`.

```

1 <fo:page-sequence master-reference="DIN-A4">
2
3 </fo:page-sequence>
    
```

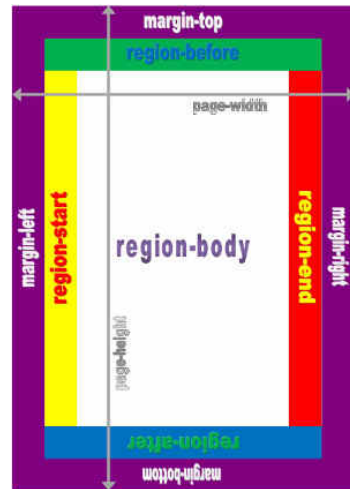
Das `static-content`-Element beinhaltet den **statischen Inhalt einer Region**, d. h. der Inhalt ist fest auf die jeweilige Region begrenzt, wird nicht umgebrochen und wird, sofern sich der fließende Inhalt über mehrere Seiten erstreckt, auf diesen Seiten wiederholt. Ein typisches Beispiel für die Anwendung von `static-content`-Elementen wäre daher die Kopf- und Fußzeile. Im `flow`-Element hingegen wird der **fließende Inhalt einer Region** notiert, d. h. ist der Inhalt (z. B. der Text) zu groß, so wird dieser umgebrochen und auf der nächsten Seite fortgeführt. Jedes `page-sequence`-Element enthält beliebig viele `static-content`-Elemente sowie exakt ein `flow`-Element. Dabei ist auch die genannte Reihenfolge einzuhalten. Um das `static-content`- und `flow`-Element einer definierten Region zuzuordnen, wird das Attribut `flow-name` verwendet.

```

1 <fo:page-sequence master-reference="DIN-A4">
2   <fo:static-content flow-name="kopf">
3     Titel für die Kopfzeile
4   </fo:static-content>
5   <fo:static-content flow-name="inhalt">
6     Fließinhalt der Seitensequenz ...
7   </fo:static-content>
8 </fo:page-sequence>
    
```

## Seitenaufbau

Bevor wir Seiten mit Inhalten befüllen können, müssen wir als erstes den Aufbau definieren. Um einen Seitenaufbau festzulegen, verwenden wir das Element `simple-page-master`. Der Name einer solchen Seitenvorlage wird über das Attribut `master-name` festgelegt. Um die **Größe einer Seite** festzulegen, verwenden wir die Attribute `page-height` (Seitenhöhe) und `page-width` (Seitenbreite). Möchten Sie noch zusätzlich einen **Druckrand** festlegen, so können Sie die Attribute `margin-left` (Druckrand links), `margin-right` (Druckrand rechts), `margin-top` (Druckrand oben) und `margin-bottom` (Druckrand unten) verwenden.



```

1 <fo:simple-page-master master-name="DIN-A4"
2   page-height="297mm" page-width="210mm"
3   margin-left="25mm" margin-right="25mm" margin-top="25mm" margin-bottom="20mm">
4
5 </fo:simple-page-master>
    
```

In XSL-FO gibt es bereits **5 fest definierte (Druck-)Bereiche** (auch Regionen genannt), die wir nutzen können, um eine Seite zu unterteilen: `region-body`, `region-start`, `region-end`, `region-before` und `region-after`. `region-body` repräsentiert den „Hauptbereich“, welcher i. d. R. für den **Fließinhalt** verwendet wird. Die Bedeutung der anderen Elemente unterscheidet sich je nach verwendeter Schreibrichtung. Wenn wir jedoch von der europäischen Schreibrichtung, also von links nach rechts und anschließend von oben nach unten, ausgehen, dann befindet sich `region-start` links, `region-end` rechts, `region-before` oben und `region-after` unten. Ein typisches Beispiel für `region-before` wäre also die Kopfzeile. Die Angaben für die Druckbereiche (abgesehen von `region-body`) sind optional und können, sofern diese nicht benötigt werden, weggelassen werden. Wollen wir den Regionen später einen Inhalt zuweisen, so können wir den Regionen mit dem Attribut `region-name` einen eindeutigen Namen geben, andernfalls wird automatisch der Name des Elements mit dem Präfix `xsl-` verwendet (z. B. bei `region-body` `xsl-region-body`). Alle `region-x`-Elemente sind inhaltsleer und können über Attribute genauer spezifiziert werden (z. B. `background-color` für die Hintergrundfarbe). Eines der wichtigsten Attribute ist `extent`, mit welchem die Breite bzw. Höhe einer

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Seitenlayout](#)

Region festgelegt wird. Dieses Attribut wird bei `region-body` nicht verwendet, da `region-body` automatisch die komplette Seite (abzüglich der Druckränder) füllt. Da i. d. R. der Hauptbereich nicht mit anderen Regionen überlappen sollte, müssen Sie im `region-body`-Element einen Abstand angeben. Hierfür dienen die Attribute `margin-left` (Abstand links), `margin-right` (Abstand rechts), `margin-top` (Abstand oben) und `margin-bottom` (Abstand unten).

```
1 <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm" >
2   <fo:region-body region-name="inhalt" margin-left="25mm" margin-right="25mm" margin-top="25mm" margin-bottom="20mm" />
3   <fo:region-before region-name="kopf" extent="25mm"/>
4 </fo:simple-page-master>
```

Nun wollen wir uns aber auch mal ein **vollständiges Beispiel** anschauen. Im folgenden XSL-FO-Code sind unter der Seitenvorlage `DIN-A4` alle Regionen von XSL-FO definiert und mit unterschiedlichen Hintergrundfarben gekennzeichnet. Als Seitengröße wird, wie der Name schon sagt, DIN-A4 verwendet. Druckränder sind in diesem Beispiel nicht definiert. Im Kopfbereich, Fußbereich und Hauptbereich (oftmals auch als Inhaltsbereich bezeichnet) werden Texte aus der XML-Datei per XSLT platziert.

#### XML-Code:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="seitenaufbau.xsl" type="text/xsl" ?>
3
4 <seite>
5   <titel>Homepage-Webhilfe XSL-FO-Kurs</titel>
6   <copy>Copyright by Homepage-Webhilfe</copy>
7   <inhalt>Hier steht der Inhalt ...</inhalt>
8 </seite>
```

#### XSL-Code:

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:fo="http://www.w3.org/1999/XSL/Format" >
4   <xsl:output method="xml" omit-xml-declaration="no" version="1.0" encoding="UTF-8" indent="yes" />
5
6   <xsl:template match="/">
7     <fo:root>
8       <fo:layout-master-set>
9         <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
10          <fo:region-body region-name="inhalt" margin="30mm" />
11          <fo:region-before region-name="kopf" extent="25mm" background-color="red" />
12          <fo:region-after region-name="fuss" extent="20mm" background-color="yellow" />
13          <fo:region-start extent="25mm" background-color="blue" />
14          <fo:region-end extent="25mm" background-color="lime" />
15        </fo:simple-page-master>
16      </fo:layout-master-set>
17
18      <fo:page-sequence master-reference="DIN-A4">
19        <fo:static-content flow-name="kopf">
20          <fo:block>
21            <xsl:value-of select="/seite/titel" />
22          </fo:block>
23        </fo:static-content>
24        <fo:static-content flow-name="fuss">
25          <fo:block>
26            <xsl:value-of select="/seite/copy" />
27          </fo:block>
28        </fo:static-content>
29        <fo:flow flow-name="inhalt">
30          <fo:block>
31            <xsl:value-of select="/seite/inhalt" />
32          </fo:block>
33        </fo:flow>
34      </fo:page-sequence>
35    </fo:root>
36  </xsl:template>
37 </xsl:stylesheet>
```

PDF:

## Seitenfolgen

Eine Seitenfolge wird durch das Element `page-sequence-master` definiert. Mit dem Attribut `master-name` wird der Name der Folge festgelegt. Dieser Name kann nachher ebenfalls wieder im Attribut `master-reference` eines `page-sequence`-Elements verwendet werden. Die Erstellung von Seitenfolgen ist komplex und würde in voller Ausführung dieses Tutorial sprengen. Daher beschäftigen wir uns hier nur mit der gängigsten und zugleich vielfältigsten Variante.

Innerhalb des `page-sequence-master`-Elements werden `repeatable-page-master-alternatives`-Elemente untergeordnet (i. d. R. jedoch nur eines). Als Attribut ist `maximum-repeat` verfügbar, um die **maximale Anzahl der Seitenwiederholungen** festzulegen.

```
1 <fo:page-sequence-master master-name="DIN-A4-Buch">
2   <fo:repeatable-page-master-alternatives>
3
4   </fo:repeatable-page-master-alternatives>
5 </fo:page-sequence-master>
```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Kartekarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Seitenlayout](#)

Innerhalb des `repeatable-page-master-alternatives`-Elements werden die einteiligen `conditional-page-master-reference`-Elemente untergeordnet. Mit dem Attribut `master-reference` wird die **Referenz zu einer der definierten Seiten** festgelegt. Über die Attribute `page-position`, `odd-or-even` und `blank-or-not-blank` können Bedingungen definiert werden. Wird beim Erstellungsvorgang des Dokuments eine neue Seite einer Seitenfolge „angefragt“, so müssen alle Bedingungen zutreffen, sodass die mit dem Attribut `master-reference` referenzierte Seite in das Ausgabedokument eingefügt wird. Mit dem Attribut `page-position` legen Sie fest, **an welcher Position sich die Seite befinden muss**. Mögliche Werte sind `first` (erste Seite), `last` (letzte Seite), `only` (erste oder letzte Seite), `rest` (restliche Seiten) und `any` (alle Seiten, Standardwert). Das Attribut `odd-or-even` wird verwendet, um die **Bedingung für die Seitennummer** festzulegen. Als Wert kann `odd` (ungerade Seitennummer), `even` (gerade Seitennummer) oder `any` (alle Seiten, Standardwert) verwendet werden. Das letzte Attribut, um eine Bedingung aufzustellen, ist `blank-or-not-blank`. Wird der Wert `blank` angegeben, so trifft die Bedingung nur dann zu, wenn für die Seite kein Fließinhalt mehr zur Verfügung steht, d. h. ist kein Fließinhalt mehr vorhanden, so wird die Seite eingefügt (sofern die anderen Bedingungen auch zutreffen). Dies ist dann sinnvoll, wenn Sie ihr Dokument auf eine gerade Seitenanzahl bringen möchten. Weitere Werte für das `blank-or-not-blank`-Attribut sind `not-blank` (Bedingung trifft zu, wenn Fließinhalt zur Verfügung steht) und `any` (Bedingung trifft immer zu, Standardwert).

```

1 <fo:page-sequence-master master-name="DIN-A4-Buch">
2   <fo:repeatable-page-master-alternatives>
3     <fo:conditional-page-master-reference master-reference="DIN-A4-Erste" page-position="first" />
4     <fo:conditional-page-master-reference master-reference="DIN-A4-Links" page-position="rest" odd-or-even="even" />
5     <fo:conditional-page-master-reference master-reference="DIN-A4-Rechts" page-position="rest" odd-or-even="odd" />
6     <fo:conditional-page-master-reference master-reference="DIN-A4-Rechts" page-position="rest" odd-or-even="odd" blank-
or-not-blank="blank" />
7     <fo:conditional-page-master-reference master-reference="DIN-A4-Letzte" page-position="last" />
8   </fo:repeatable-page-master-alternatives>
9 </fo:page-sequence-master>

```

Im folgenden **Beispiel** werden 3 unterschiedliche Seitenvorlagen definiert: `DIN-A4-Deckblatt`, `DIN-A4-Inhalt-Links` und `DIN-A4-Inhalt-Rechts`. Zudem wird die Vorlage für die Seitenfolge `DIN-A4-Inhalt` definiert. In der Seitenfolge wird lediglich die Bedingung für die Seitennummer verwendet, um somit einen unterschiedlichen Abstand für linke und rechte Seiten zu verwenden. Das zu diesem Beispiel gehörende XML-Dokument ist ebenfalls komplexer und enthält Themen mit Namen und Absätzen und soll damit ein Buch darstellen. Mit Hilfe des XSL-Skripts wird aus dieser XML-Datei eine buchähnliche PDF-Datei generiert.

#### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="seitenfolgen.xsl" type="text/xsl" ?>
3
4 <buch>
5   <titel>Homepage-Webhilfe XSL-FO-Kurs</titel>
6   <copyright>Copyright by Homepage-Webhilfe</copyright>
7   <thema>
8     <name>Grundlagen</name>
9     <absatz>...</absatz>
10    <absatz>...</absatz>
11    <absatz>...</absatz>
12  </thema>
13  <thema>
14    <name>Seitenlayout</name>
15    <absatz>...</absatz>
16    <absatz>...</absatz>
17    <absatz>...</absatz>
18    <absatz>...</absatz>
19    <absatz>...</absatz>
20  </thema>
21  <thema>
22    <name>Textformatierung</name>
23    <absatz>...</absatz>
24    <absatz>...</absatz>
25  </thema>
26  <thema>
27    <name>Abstände und Rahmen</name>
28    <absatz>...</absatz>
29    <absatz>...</absatz>
30    <absatz>...</absatz>
31  </thema>
32  <thema>
33    <name>Listen</name>
34    <absatz>...</absatz>
35    <absatz>...</absatz>
36  </thema>
37 </buch>

```

#### XSL-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:fo="http://www.w3.org/1999/XSL/Format">
4   <xsl:output method="xml" omit-xml-declaration="no" version="1.0" encoding="UTF-8" indent="yes" />
5
6   <xsl:template match="/">
7     <fo:root>
8       <fo:layout-master-set>
9         <fo:simple-page-master master-name="DIN-A4-Deckblatt" page-height="297mm" page-width="210mm">

```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: Homepage-Webhilfe » XML & Co. » XSL-FO » Seitenlayout

```

10         <fo:region-body region-name="inhalt" margin="50mm" />
11         <fo:region-after region-name="fuss" extent="20mm" />
12     </fo:simple-page-master>
13     <fo:simple-page-master master-name="DIN-A4-Inhalt-Links" page-height="297mm" page-width="210mm">
14         <fo:region-body region-name="inhalt" margin-left="25mm" margin-right="10mm" margin-top="15mm" margin-
bottom="15mm" />
15     </fo:simple-page-master>
16     <fo:simple-page-master master-name="DIN-A4-Inhalt-Rechts" page-height="297mm" page-width="210mm">
17         <fo:region-body region-name="inhalt" margin-left="10mm" margin-right="25mm" margin-top="15mm" margin-
bottom="15mm" />
18     </fo:simple-page-master>
19     <fo:page-sequence-master master-name="DIN-A4-Inhalt">
20         <fo:repeatable-page-master-alternatives>
21             <fo:conditional-page-master-reference master-reference="DIN-A4-Inhalt-Links" odd-or-even="even" />
22             <fo:conditional-page-master-reference master-reference="DIN-A4-Inhalt-Rechts" odd-or-even="odd" />
23         </fo:repeatable-page-master-alternatives>
24     </fo:page-sequence-master>
25 </fo:layout-master-set>
26
27     <fo:page-sequence master-reference="DIN-A4-Deckblatt">
28         <fo:static-content flow-name="fuss">
29             <fo:block>
30                 <xsl:value-of select="/buch/copyright" />
31             </fo:block>
32         </fo:static-content>
33         <fo:flow flow-name="inhalt">
34             <fo:block>
35                 <xsl:value-of select="/buch/titel" />
36             </fo:block>
37         </fo:flow>
38     </fo:page-sequence>
39     <fo:page-sequence master-reference="DIN-A4-Inhalt">
40         <fo:flow flow-name="inhalt">
41             <xsl:apply-templates />
42         </fo:flow>
43     </fo:page-sequence>
44 </fo:root>
45 </xsl:template>
46
47 <xsl:template match="/buch/thema">
48     <!-- Titel -->
49     <fo:block>
50         <xsl:value-of select="name" />
51     </fo:block>
52     <!-- Absätze anwenden -->
53     <xsl:apply-templates />
54     <!-- Platzhalter -->
55     <fo:block>
56         <fo:leader />
57     </fo:block>
58 </xsl:template>
59
60 <xsl:template match="/buch/thema/absatz">
61     <fo:block>
62         <xsl:value-of select="." />
63     </fo:block>
64 </xsl:template>
65 </xsl:stylesheet>

```

PDF:

Übrigens: Das `leader`-Element in diesem Beispiel wird lediglich dazu verwendet, eine leere Zeile zu erzeugen.

### Seitenumbruch

Um den **Seitenumbruch nach, vor oder in Elementen** zu steuern, gibt es die Attribute `page-break-after`, `page-break-before` und `page-break-inside`. Mögliche Werte für `page-break-after` oder `page-break-before` sind `auto` (Umbruch je nach Bedarf, Standardwert), `always` (Umbruch erzwingen), `left` (Umbruch erzwingen, sodass nächste Seite eine linke Seite ist), `right` (Umbruch erzwingen, sodass nächste Seite eine rechte Seite ist) und `avoid` (Umbruch vermeiden). Für das Attribut `page-break-inside`, mit welchem der Umbruch innerhalb eines Elements gesteuert wird, sind nur die Werte `auto` und `avoid` zulässig.

```

1 <fo:block>
2     Dieser Block enthält einen Text ...
3 </fo:block>
4 <fo:block page-break-before="always">
5     Dieser Block beginnt immer auf einer neuen Seite.
6 </fo:block>

```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Textformatierung](#)

## Textformatierung



In XSL-FO gibt es einige Attribute, mit welchen Texte formatiert und Schrifteinstellungen festgelegt werden können. Diese **Typografie-Attribute** wollen wir in diesem Thema genauer erläutern. Verwendet werden diese Attribute entweder in `block`- oder in `inline`-Elementen. Zudem sind in einigen weiteren Elementen diese Attribute ebenfalls gültig. Falls Sie sich bereits mit CSS auskennen, werden Ihnen einige Attribute bekannt vorkommen.

**Inhalt dieser Seite:**

1. Schriftfarbe
2. Schriftart und Schriftgröße
3. Schriftstile
4. Ausrichtung und Abstände
5. Horizontale Linien

### Schriftfarbe

Die Schriftfarbe lässt sich über das Attribut `color` festlegen. Als Werte sind Hex-RGB-Werte (`#RRGGBB` oder in der Kurzform `#RGB`), RGB-Funktionswerte (`rgb(r, g, b)`) und englische Farbnamen möglich. Die unterstützten **Farbnamen** sind vom FO-Prozessor abhängig, jedoch kann man davon ausgehen, dass grundlegende Farbnamen, wie z. B. `red`, `blue` oder `lime`, unterstützt werden.

**XSL-FO-Code:**

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4   <fo:layout-master-set>
5     <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6       <fo:region-body region-name="inhalt" margin="2cm" />
7     </fo:simple-page-master>
8   </fo:layout-master-set>
9
10  <fo:page-sequence master-reference="DIN-A4">
11    <fo:flow flow-name="inhalt">
12      <fo:block>
13        In diesem Dokument stehen alle Farben des
14        <fo:inline color="#FF0000">R</fo:inline>
15        <fo:inline color="#00FF00">G</fo:inline>
16        <fo:inline color="#0000FF">B</fo:inline> Farbraums zur Verfügung!
17      </fo:block>
18    </fo:flow>
19  </fo:page-sequence>
20 </fo:root>
    
```

PDF:

## Schriftart und Schriftgröße

Um die Schriftart, also den **Namen der Schrift**, festzulegen, verwenden wir das Attribut `font-family`. Auch hier hängt es wieder vom FO-Prozessor ab, welche Schriften zur Verfügung stehen. Es ist jedoch zu empfehlen, nur Schriftarten zu verwenden, die laut der PostScript-Spezifikation in jedem PDF-Reader enthalten sein müssen: `Helvetica`, `Times`, `Courier`, `Symbol` und `ZapfDingbats`. Die **Größe der Schrift** wird über das Attribut `font-size` festgelegt. Hier wird ein Zahlenwert in Verbindung mit einer Maßeinheit angegeben.

**XSL-FO-Code:**

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4   <fo:layout-master-set>
5     <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6       <fo:region-body region-name="inhalt" margin="2cm" />
7     </fo:simple-page-master>
8   </fo:layout-master-set>
9
10  <fo:page-sequence master-reference="DIN-A4">
11    <fo:flow flow-name="inhalt">
12      <fo:block font-family="Times" font-size="24px">Schriftart und Schriftgröße</fo:block>
13      <fo:block font-family="Times" font-size="18px">
14        XSL-FO unterstützt verschiedene Schriftarten (z. B.
15        <fo:inline font-family="Courier">Courier</fo:inline> oder
16        <fo:inline font-family="Helvetica">Helvetica</fo:inline>)
17        und verschiedene Schriftgrößen (z. B.
18        <fo:inline font-size="14px">14px</fo:inline>,
19        <fo:inline font-size="17px">17px</fo:inline> oder
20        <fo:inline font-size="20px">20px</fo:inline>).
21      </fo:block>
22    </fo:flow>
23  </fo:page-sequence>
24 </fo:root>
    
```

PDF:

## Schriftstile

Mit dem Attribut `font-style` können Sie die **Schriftlage** festlegen. Mögliche Werte für das Attribut sind `normal` (Normallage / gerade Schrift, Standardwert),

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen </p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Textformatierung](#)

`italic` (Kursiv-Schrift) und `oblique` (schräg gestellte Schrift). Oftmals ist zwischen `italic` und `oblique` kein Unterschied zu erkennen.

Die **Schriftbreite** bzw. **Schriftdicke** lässt sich über das Attribut `font-weight` festlegen. Als Werte können `normal` (normale Breite, Standardwert), `bold` (fette Schrift), `light` (dünne Schrift) sowie Zahlenwerte zwischen 100 und 900 (immer in 100er-Schritten), `bolder` und `lighter` angegeben werden. Keine der gängigen Schriftarten unterstützt jedoch andere Werte als `normal` und `bold`.

Mit Hilfe des Attributs `text-decoration` können Sie einem Text eine „Dekoration“ hinzufügen, gemeint sind damit Linien, um einen Text zu **unterstreichen** (`underline`), zu **überstreichen** (`overline`) oder **durchzustreichen** (`line-through`). Die Standardeinstellung ist `none`, welche keine Linie erzeugt.

Um die **Groß- und Kleinschreibung** in einem Text zu manipulieren, gibt es das Attribut `text-transform`. Mit dem Wert `uppercase` werden alle Buchstaben in Großbuchstaben umgewandelt und mit dem Wert `lowercase` werden alle Buchstaben in Kleinbuchstaben umgewandelt. Neben dem Standardwert `none` gibt es noch einen weiteren möglichen Wert: `capitalize`. Der Wert `capitalize` erzeugt Kapitälchen, dabei wird der erste Buchstabe jedes Wortes in einen Großbuchstaben und alle anderen Buchstaben in Kleinbuchstaben umgewandelt.

## XSL-FO-Code:

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4    <fo:layout-master-set>
5      <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6        <fo:region-body region-name="inhalt" margin="2cm" />
7      </fo:simple-page-master>
8    </fo:layout-master-set>
9
10   <fo:page-sequence master-reference="DIN-A4">
11     <fo:flow flow-name="inhalt">
12       <fo:block font-style="italic">Dieser Text wird kursiv dargestellt!</fo:block>
13       <fo:block font-family="Times" font-weight="bold">Dieser Text wird fett dargestellt!</fo:block>
14       <fo:block>
15         Dieser Text wird
16         <fo:inline text-decoration="underline">unterstrichen</fo:inline>,
17         <fo:inline text-decoration="overline">überstrichen</fo:inline> und
18         <fo:inline text-decoration="line-through">durchgestrichen</fo:inline> angezeigt!
19       </fo:block>
20       <fo:block text-transform="uppercase">Dieser Text verwendet nur Großbuchstaben!</fo:block>
21       <fo:block text-transform="lowercase">Dieser Text verwendet nur Kleinbuchstaben!</fo:block>
22       <fo:block text-transform="capitalize">Dieser Text verwendet Kapitälchen!</fo:block>
23     </fo:flow>
24   </fo:page-sequence>
25 </fo:root>

```

PDF:

## Ausrichtung und Abstände

Um einen **Text auszurichten**, wird das Attribut `text-align` verwendet. Für die letzte Zeile kann mit Hilfe des Attributs `text-align-last` eine gesonderte Ausrichtung durchgeführt werden. Mögliche Werte für diese beiden Attribute sind `left` (linksbündig), `right` (rechtsbündig), `start` (bündig mit dem `start-Region`), `end` (bündig mit der `end-Region`), `inside` (bündig zum Bund), `outside` (bündig zum äußeren Rand), `justify` (Blocksatz) und `center` (zentriert).

Um einen **Block einzurücken**, gibt es die Attribute `start-indent`, `end-indent`, `text-indent` und `last-line-indent`. Die Attribute `start-indent` und `end-indent` werden dazu verwendet, den kompletten Absatz einzurücken. `text-indent` wird für die Einrückung der ersten Zeile verwendet, `last-line-indent` hingegen für die letzte Zeile. Als Wert für diese Attribute wird eine Zahl zusammen mit einer Maßeinheit verwendet. Auch negative Zahlen sind möglich.

Das Attribut `line-height` legt die **Zeilenhöhe** fest. Auch hier wird eine Zahl mit einer Maßeinheit als Wert angegeben. Der **Abstand zwischen Wörtern und Zeichen** lässt sich ebenfalls steuern. Dafür werden die Attribute `word-spacing` und `letter-spacing` verwendet.

Möchten Sie einen **Text hoch- oder tiefstellen**, so können Sie das Attribut `baseline-shift` verwenden. Neben numerischen Angaben (zusammen mit einer Maßeinheit) sind auch die Schlüsselwörter `super` (hochgestellt) und `sub` (tiefgestellt) möglich. Der Standardwert ist `baseline` und erzeugt keine Verschiebung nach oben oder unten.

## XSL-FO-Code:

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4    <fo:layout-master-set>
5      <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6        <fo:region-body region-name="inhalt" margin="2cm" />
7      </fo:simple-page-master>
8    </fo:layout-master-set>
9
10   <fo:page-sequence master-reference="DIN-A4">
11     <fo:flow flow-name="inhalt">
12       <fo:block text-align="left">Dieser Text wird linksbündig dargestellt!</fo:block>
13       <fo:block text-align="right">Dieser Text wird rechtsbündig dargestellt!</fo:block>
14       <fo:block text-align="center">Dieser Text wird zentriert dargestellt!</fo:block>
15       <fo:block text-align="justify">
16         Dieser Text wird als Blocksatz dargestellt: ...
17       </fo:block>
18       <fo:block start-indent="20mm" end-indent="20mm" text-indent="5mm">
19         Dieser Text wird mit einem Einzug von 5mm (erste Zeile) dargestellt: ...

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Textformatierung](#)

```

20     </fo:block>
21     <fo:block line-height="10mm">
22         Dieser Text wird mit einer Zeilenhöhe von 10mm dargestellt: ...
23     </fo:block>
24     <fo:block letter-spacing="1mm" word-spacing="3mm">
25         Dieser Text wird mit einem Zeichenabstand von 1mm und einem Wortabstand von 3mm dargestellt: ...
26     </fo:block>
27     <fo:block>Dieser Text wird <fo:inline baseline-shift="1mm">hochgestellt</fo:inline> und <fo:inline baseline-
shift="-1mm">tiefgestellt</fo:inline>!</fo:block>
28     </fo:flow>
29 </fo:page-sequence>
30 </fo:root>

```

PDF:

**Übrigens:** Die vertikale Ausrichtung lässt sich mit Hilfe des Attributs `vertical-align` steuern. Verwendet werden kann das Attribut aber nicht bei normalen Blöcken. Der Einsatz des Attributs `vertical-align` findet sich hauptsächlich bei Tabellenzellen. Mögliche Werte für das Attribut sind unter anderem `top` (Ausrichtung oben), `bottom` (Ausrichtung unten) und `middle` (Ausrichtung mittig).

## Horizontale Linien

Eine horizontale Linie lässt sich mit dem Element `leader` erzeugen. Das Element `leader` wird innerhalb des `block`-Elements platziert. Oft werden die horizontalen Linien dazu verwendet, **Führungslinien** (z. B. wie in Inhaltsverzeichnissen) zu erzeugen. Als Attribute stehen unter anderem `leader-pattern` (Füllmuster: `space` für Leerzeichen, `dots` für Punkte, `rule` für eine Linie oder `use-content` für beliebigen Inhalt, der innerhalb des `leader`-Elements angegeben wird), `rule-thickness` (Dicke der Linie) und `color` (Farbe der Linie) zur Verfügung.

### XSL-FO-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4     <fo:layout-master-set>
5         <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6             <fo:region-body region-name="inhalt" margin="2cm" />
7         </fo:simple-page-master>
8     </fo:layout-master-set>
9
10    <fo:page-sequence master-reference="DIN-A4">
11        <fo:flow flow-name="inhalt">
12            <fo:block text-align-last="justify">Grundlagen <fo:leader leader-pattern="dots" />Seite 1</fo:block>
13            <fo:block text-align-last="justify">Seitenlayout <fo:leader leader-pattern="dots" />Seite 2</fo:block>
14            <fo:block text-align-last="justify">Textformatierung <fo:leader leader-pattern="dots" />Seite 3</fo:block>
15        </fo:flow>
16    </fo:page-sequence>
17 </fo:root>

```

PDF:

**Übrigens:** Ein `leader`-Element ohne Angabe weiterer Attribute kann dazu verwendet werden, eine leere Zeile zu erzeugen. Das hier verwendete Attribut `text-align-last` und der Wert `justify` ist zwingend erforderlich, sodass sich der Inhalt über die komplette Zeile erstreckt.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislöcher



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Abstände und Rahmen](#)

## Abstände und Rahmen

Um einen Block oder ein anderes Element weiter zu formatieren, gibt es die Möglichkeit, Abstände und Rahmen festzulegen. Bei den Abständen wird zwischen dem Außenabstand und dem Innenabstand unterschieden. Eine weitere Möglichkeit zur Abstandsteuerung ist die Einrückung, welche Sie im vorherigen Thema bereits kennengelernt haben.

### Inhalt dieser Seite:

1. Außenabstand
2. Rahmen
3. Innenabstand

### Außenabstand

Der Außenabstand legt den **Abstand des Blocks zu anderen Elementen** (oder dem Regions-Bereich) fest. Der eigentliche Inhaltsbereich von Blöcken, deren Größe nicht festgelegt ist, wird durch die Angabe eines Außenabstands verkleinert. Der Außenabstand wird durch das Attribut `margin` festgelegt. Mit den Erweiterungen `-left` (links), `-right` (rechts), `-top` (oben) und `-bottom` (unten) können unterschiedliche Werte für die verschiedenen Seiten festgelegt werden. Somit legt das Attribut `margin-left` also den Abstand für die linke Seite fest.

#### XSL-FO-Code:

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4    <fo:layout-master-set>
5      <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6        <fo:region-body region-name="inhalt" margin="2cm" />
7      </fo:simple-page-master>
8    </fo:layout-master-set>
9
10   <fo:page-sequence master-reference="DIN-A4">
11     <fo:flow flow-name="inhalt">
12       <fo:block background-color="red" text-align="center" margin-bottom="1cm">Block 1</fo:block>
13       <fo:block background-color="lime" text-align="center" margin-left="15mm" margin-right="5mm">Block 2</fo:block>
14       <fo:block background-color="blue" text-align="center" margin="3cm">Block 3</fo:block>
15     </fo:flow>
16   </fo:page-sequence>
17 </fo:root>

```

PDF:

### Rahmen

Der Rahmen ist ein **Teil zwischen Außenabstand und Innenabstand** und wird mit Hilfe des Attributs `border` festgelegt. Neben der optionalen Erweiterung für die Seite (also `-left`, `-right` etc.) kann ein weiterer Teil an die Attributnamen angehängt werden: `-color` (Farbangabe), `-style` (Stil des Rahmens) und `-width` (Breite). Für den **Rahmenstil** stehen unter anderem die Werte `solid` (durchgezogene Linie), `double` (doppelte Linie), `dashed` (gestrichelte Linie) und `dotted` (gepunktete Linie).

#### XSL-FO-Code:

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4    <fo:layout-master-set>
5      <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6        <fo:region-body region-name="inhalt" margin="2cm" />
7      </fo:simple-page-master>
8    </fo:layout-master-set>
9
10   <fo:page-sequence master-reference="DIN-A4">
11     <fo:flow flow-name="inhalt">
12       <fo:block background-color="red" text-align="center" border-bottom-width="1mm" border-bottom-style="dashed">Block
13 1</fo:block>
14       <fo:block background-color="lime" text-align="center" border-color="gray" border-style="solid" border-left-
15 width="1.5mm" border-right-width="0.5mm">Block 2</fo:block>
16       <fo:block background-color="blue" text-align="center" border-color="orange" border-style="dotted" border-top-
17 style="solid">Block 3</fo:block>
18     </fo:flow>
19   </fo:page-sequence>
20 </fo:root>

```

PDF:

**Übrigens:** Hintergründe erstrecken sich lediglich über den Inhaltsbereich sowie den dazugehörigen Innenabstand, nicht aber über den Außenabstand und Rahmen.

### Innenabstand

Der Innenabstand legt den **Abstand zwischen dem Rahmen und dem eigentlichen Inhalt** des Blocks fest. Das Festlegen des Innenabstands erfolgt über das Attribut `padding` oder bei Bedarf über die Attribute `padding-left`, `padding-right`, `padding-top` und `padding-bottom`.

#### XSL-FO-Code:

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Abstände und Rahmen](#)

```

4      <fo:layout-master-set>
5          <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6              <fo:region-body region-name="inhalt" margin="2cm" />
7          </fo:simple-page-master>
8      </fo:layout-master-set>
9
10     <fo:page-sequence master-reference="DIN-A4">
11         <fo:flow flow-name="inhalt">
12             <fo:block background-color="red" text-align="center" padding-bottom="1cm">Block 1</fo:block>
13             <fo:block background-color="lime" text-align="center" padding-left="15mm" padding-right="5mm">Block 2</fo:block>
14             <fo:block background-color="blue" text-align="center" padding="3cm">Block 3</fo:block>
15         </fo:flow>
16     </fo:page-sequence>
17 </fo:root>

```

PDF: 

**Wichtig:** Wenn Sie sich das obige Beispiel anschauen, werden Sie feststellen, dass durch die Angabe der `padding`-Attribute der Block vergrößert wird und sich nun über die Regionsgrenze erstreckt. Ist dies nicht erwünscht, so muss `margin="0"` notiert werden. Wird der Außenabstand bereits angegeben, so fällt diese Problematik sowieso weg.

**Übrigens:** Bei den Attributen für den Rahmen und Innenabstand können an Stelle der Erweiterungen `-left`, `-right`, `-top` oder `-bottom` für die Seitenbestimmung auch die Erweiterungen `-start`, `-end`, `-before` oder `-after` verwendet werden.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung

Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Grafiken und Hintergrund](#)

## Grafiken und Hintergrund

In diesem Thema lernen Sie, wie Sie Grafiken in Ihr Dokument integrieren und wie Sie Hintergründe festlegen.

### Inhalt dieser Seite:

1. Grafiken
2. Hintergrundfarbe
3. Hintergrundbild

### Grafiken

Um eine Grafik im Dokument zu platzieren, wird das einteilige Element `external-graphic` verwendet. Über die Attribute `content-height` und `content-width` lässt sich die **Höhe und Breite** festlegen. Dabei ist es auch möglich, eines der Attribute wegzulassen. In diesem Fall wird die andere Größe automatisch berechnet, wobei das Seitenverhältnis beibehalten wird. Werden beide Attribute nicht angegeben, so wird die Originalgröße des Bilds verwendet. Die **Quelle** des Bilds wird als Pfadangabe über das Attribut `src` angegeben.

#### XSL-FO-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4   <fo:layout-master-set>
5     <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6       <fo:region-body region-name="inhalt" margin="2cm" />
7     </fo:simple-page-master>
8   </fo:layout-master-set>
9
10  <fo:page-sequence master-reference="DIN-A4">
11    <fo:flow flow-name="inhalt">
12      <fo:block>
13        <fo:external-graphic src="../../../../Bilder/Logo/Logo.jpg" content-width="150px" content-height="150px" />
14      </fo:block>
15    </fo:flow>
16  </fo:page-sequence>
17 </fo:root>

```

PDF:

**Übrigens:** Beim Transformationsvorgang durch den FO-Prozessor wird das Bild in das Dokument eingebettet. Nach der Erzeugung muss das Bild nicht mehr unter dem angegebenen Pfad existieren.

### Hintergrundfarbe

Die Hintergrundfarbe lässt sich über das Attribut `background-color` festlegen. Als Werte sind, so wie beim `color`-Attribut auch, Hex-RGB-Werte, RGB-Funktionswerte und englische Farbnamen möglich.

#### XSL-FO-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4   <fo:layout-master-set>
5     <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6       <fo:region-body region-name="inhalt" margin="2cm" />
7     </fo:simple-page-master>
8   </fo:layout-master-set>
9
10  <fo:page-sequence master-reference="DIN-A4">
11    <fo:flow flow-name="inhalt">
12      <fo:block-container background-color="red" width="5cm" height="2cm"><fo:block></fo:block></fo:block-container>
13      <fo:block-container background-color="lime" width="5cm" height="2cm"><fo:block></fo:block></fo:block-container>
14      <fo:block-container background-color="blue" width="5cm" height="2cm"><fo:block></fo:block></fo:block-container>
15    </fo:flow>
16  </fo:page-sequence>
17 </fo:root>

```

PDF:

### Hintergrundbild

Möchten Sie als Hintergrund ein Bild (z. B. für einen Farbverlauf oder ein Wasserzeichen) an Stelle einer Farbe verwenden, so können Sie das Attribut `background-image` einsetzen.

#### XSL-FO-Code:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4   <fo:layout-master-set>
5     <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6       <fo:region-body region-name="inhalt" margin="2cm" />
7     </fo:simple-page-master>
8   </fo:layout-master-set>
9
10  <fo:page-sequence master-reference="DIN-A4">

```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Grafiken und Hintergrund](#)

```
11     <fo:flow flow-name="inhalt">
12         <fo:block-container background-image="../../../Bilder/Logo/Logo.jpg" width="5cm" height="2cm"><fo:block>
13     </fo:block></fo:block-container>
14 </fo:flow>
15 </fo:page-sequence>
16 </fo:root>
```

PDF: 

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » XML & Co. » XSL-FO » Links

## Links

Um in XSL-FO einen Link zu erzeugen, gibt es das Element `basic-link`. Das Element ist zweiteilig, wobei sich der definierte Link auf den untergeordneten Inhalt auswirkt. Bei dem Inhalt kann es sich um einen Text, um eine Grafik oder um beides handeln. Auch andere untergeordnete Elemente sind denkbar. Zum Erzeugen eines **externen Links** benötigen Sie das Attribut `external-destination`. Als Attributwert wird ein **vollständiger Hyperlink** angegeben. Möchten Sie einen **Verweis innerhalb des Dokuments** erzeugen, so können Sie das Attribut `internal-destination` verwenden. Als Wert des Attributs wird dann ein **eindeutiger Name** verwendet, welcher mit dem Wert des `id`-Attributs eines im Dokument enthaltenen Elements übereinstimmt. Das `id`-Attribut kann bei fast allen Elementen verwendet werden.

### XSL-FO-Code:

```

1 | <?xml version="1.0" encoding="UTF-8" ?>
2 |
3 | <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4 |   <fo:layout-master-set>
5 |     <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6 |       <fo:region-body region-name="inhalt" margin="2cm" />
7 |     </fo:simple-page-master>
8 |   </fo:layout-master-set>
9 |
10 |   <fo:page-sequence master-reference="DIN-A4">
11 |     <fo:flow flow-name="inhalt">
12 |       <fo:block>
13 |         <fo:basic-link external-destination="https://www.homepage-webhilfe.de/">Zur Webseite ...</fo:basic-link>
14 |       </fo:block>
15 |       <fo:block>
16 |         <fo:basic-link internal-destination="Seite2">Zur 2. Seite ...</fo:basic-link>
17 |       </fo:block>
18 |       <fo:block id="Seite2" page-break-before="always">
19 |         Dies ist der Inhalt der 2. Seite: ...
20 |       </fo:block>
21 |     </fo:flow>
22 |   </fo:page-sequence>
23 | </fo:root>

```

PDF: 

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Listen](#)

## Listen

XSL-FO enthält die Möglichkeit, Listen zu erzeugen. Hierfür wird zu allererst das Element `list-block` benötigt. In diesem Element werden nun `list-item`-Elemente untergeordnet, welche die einzelnen **Listenpunkte** repräsentieren. Jeder Listenpunkt enthält nun genau zwei weitere Unterelemente: `list-item-label` und `list-item-body`. Das `list-item-label` stellt die **Beschriftung für den Listenpunkt** dar. Dies ist i. d. R. ein Aufzählungszeichen oder ein fortlaufender Wert (numerisch, alphanummerisch oder römisch). XSL-FO enthält selbst keinen Mechanismus, um eine solche geordnete Aufzählung zu erzeugen. Hier wird dann meistens das `number`-Element von XSLT verwendet, um einen solchen Aufzählungswert zu erzeugen (siehe Beispiel). Der **eigentliche Inhalt des Aufzählungspunkts** wird im Element `list-item-body` notiert. Sowohl im `list-item-label`- als auch im `list-item-body`-Element darf der Inhalt nicht direkt notiert werden, sondern muss in einem `block`-Element untergeordnet werden. Standardmäßig überlagert sich der Inhalt der Elemente `list-item-label` und `list-item-body`. Um einen Abstand zu erzeugen, werden üblicherweise die Attribute `start-indent` und `end-indent` verwendet.

### XML-Code:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <?xml-stylesheet href="listen.xsl" type="text/xsl" ?>
3
4 <zahlen>
5   <zahl>45</zahl>
6   <zahl>39</zahl>
7   <zahl>22</zahl>
8   <zahl>70</zahl>
9   <zahl>68</zahl>
10  <zahl>12</zahl>
11  <zahl>51</zahl>
12  <zahl>63</zahl>
13  <zahl>27</zahl>
14  <zahl>94</zahl>
15 </zahlen>
    
```

### XSL-Code:

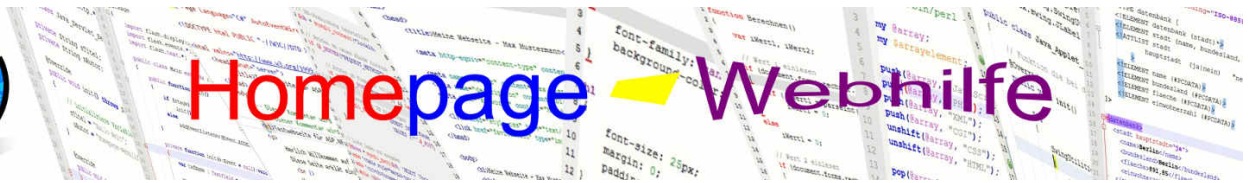
```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:fo="http://www.w3.org/1999/XSL/Format">
4   <xsl:output method="xml" omit-xml-declaration="no" version="1.0" encoding="UTF-8" indent="yes" />
5
6   <xsl:template match="/">
7     <fo:root>
8       <fo:layout-master-set>
9         <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
10          <fo:region-body region-name="inhalt" margin="2cm" />
11        </fo:simple-page-master>
12      </fo:layout-master-set>
13
14      <fo:page-sequence master-reference="DIN-A4">
15        <fo:flow flow-name="inhalt">
16          <fo:list-block>
17            <xsl:apply-templates />
18          </fo:list-block>
19        </fo:flow>
20      </fo:page-sequence>
21    </fo:root>
22  </xsl:template>
23
24  <xsl:template match="/zahlen/zahl">
25    <fo:list-item>
26      <fo:list-item-label>
27        <fo:block>
28          <xsl:number format="1." />
29        </fo:block>
30      </fo:list-item-label>
31      <fo:list-item-body start-indent="10mm">
32        <fo:block>
33          <xsl:value-of select="." />
34        </fo:block>
35      </fo:list-item-body>
36    </fo:list-item>
37  </xsl:template>
38 </xsl:stylesheet>
    
```

PDF:

**Wichtig:** Im obigen Beispiel wird ein fest definierter Abstand verwendet. Nachteil dieser Variante ist die Abhängigkeit der Schriftgröße sowie der evtl. zu kleine Abstand bei langen Aufzählungen. Abhilfe schafft der Attributwert `label-end()` im Attribut `end-indent` des `list-item-label`-Elements sowie der Attributwert `body-start()` im Attribut `start-indent` des `list-item-body`-Elements. Der Abstand zwischen dem `list-item-label`- und `list-item-body`-Element lässt sich dann über das Attribut `provisional-distance-between-starts`, welches im `list-block`-Element notiert wird, festlegen. Der Standardwert dieses Attributs ist 24pt.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Tabellen](#)

## Tabellen

Eine Tabelle kennzeichnet sich in XSL-FO durch das Element `table`. Jede XSL-FO-Tabelle hat **3 grundlegende Bereiche**: den Tabellenkopf (`table-header`), den Tabellenfuß (`table-footer`) und den Tabelleninhalt (`table-body`). Der Tabellenkopf und der Tabellenfuß sind jedoch optional. Innerhalb der Elemente für die einzelnen Tabellenbereiche werden `table-row`-Elemente untergeordnet, welche eine **Tabellenzeile** repräsentieren. Diesen Elementen werden wiederum **Tabellenzellen** (`table-cell`-Elemente) untergeordnet. Denken Sie daran, dass der Inhalt einer Tabellenzelle nicht direkt untergeordnet werden darf, sondern innerhalb eines `block`-Elements angegeben werden muss. Um eine Zelle über mehrere Zeilen oder Spalten zu spannen (engl. *span*), können Sie die Attribute `number-rows-spanned` und `number-columns-spanned` verwenden. Bei Bedarf können Sie direkt innerhalb des `table`-Elements `table-column`-Elemente unterordnen, um die **Definitionen einer Spalte** bereits im Voraus zu treffen. Das `table-column`-Element ist einteilig. Dort können dann z. B. Attribute für den Innen- und Außenabstand sowie für den Rahmen festgelegt werden. Mit dem Attribut `column-width` kann die Breite der Spalte festgelegt werden. Wird diese Variante nicht verwendet, so können Sie die Breite einer Zelle mit Hilfe des Attributs `width` festlegen. Die Angabe der Zellenbreite ist nur einmal pro Spalte erforderlich.

### XML-Code:

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2  <?xml-stylesheet href="tabellen.xsl" type="text/xsl" ?>
3
4  <firma>
5      <mitarbeiter>
6          <name>Max Mustermann</name>
7          <beruf>Geschäftsführer</beruf>
8          <durchwahl>10</durchwahl>
9      </mitarbeiter>
10     <mitarbeiter>
11         <name>Maria Musterfrau</name>
12         <beruf>Verkäuferin</beruf>
13         <durchwahl>20</durchwahl>
14     </mitarbeiter>
15     <mitarbeiter>
16         <name>Peter Müller</name>
17         <beruf>Lagerist</beruf>
18         <durchwahl>30</durchwahl>
19     </mitarbeiter>
20 </firma>
    
```

### XSL-Code:

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:fo="http://www.w3.org/1999/XSL/Format">
4      <xsl:output method="xml" omit-xml-declaration="no" version="1.0" encoding="UTF-8" indent="yes" />
5
6      <xsl:template match="/">
7          <fo:root>
8              <fo:layout-master-set>
9                  <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
10                     <fo:region-body region-name="inhalt" margin="2cm" />
11                 </fo:simple-page-master>
12             </fo:layout-master-set>
13
14             <fo:page-sequence master-reference="DIN-A4">
15                 <fo:flow flow-name="inhalt">
16                     <fo:table font-family="Times">
17                         <fo:table-column column-width="40%" border-style="solid" border-width="0.25mm" border-left-
18 width="0.5mm" />
19                         <fo:table-column column-width="40%" border-style="solid" border-width="0.25mm" />
20                         <fo:table-column column-width="20%" border-style="solid" border-width="0.25mm" border-right-
21 width="0.5mm" />
22                     <fo:table-header>
23                         <fo:table-row background-color="#CCCCCC" text-align="center" font-weight="bold" font-size="18px">
24                             <fo:table-cell number-columns-spanned="3">
25                                 <fo:block>Mitarbeiterübersicht</fo:block>
26                             </fo:table-cell>
27                         </fo:table-row>
28                         <fo:table-row background-color="#EEEEEE" border-style="solid" border-width="0.25mm" text-
29 align="center" font-weight="bold" font-size="16px">
30                             <fo:table-cell>
31                                 <fo:block>Name</fo:block>
32                             </fo:table-cell>
33                             <fo:table-cell>
34                                 <fo:block>Beruf</fo:block>
35                             </fo:table-cell>
36                             <fo:table-cell>
37                                 <fo:block>Durchwahl</fo:block>
38                             </fo:table-cell>
39                         </fo:table-row>
40                     </fo:table-header>
41                 </fo:flow>
42             </fo:page-sequence>
43         </fo:root>
44     </xsl:template>
45 </xsl:stylesheet>
    
```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Tabellen](#)

```

38         <fo:table-body>
39             <xsl:apply-templates />
40         </fo:table-body>
41     </fo:table>
42 </fo:flow>
43 </fo:page-sequence>
44 </fo:root>
45 </xsl:template>
46
47 <xsl:template match="/firma/mitarbeiter">
48     <fo:table-row border-style="solid" border-width="0.25mm" font-size="16px">
49         <fo:table-cell>
50             <fo:block>
51                 <xsl:value-of select="name" />
52             </fo:block>
53         </fo:table-cell>
54         <fo:table-cell>
55             <fo:block>
56                 <xsl:value-of select="beruf" />
57             </fo:block>
58         </fo:table-cell>
59         <fo:table-cell>
60             <fo:block>
61                 <xsl:value-of select="durchwahl" />
62             </fo:block>
63         </fo:table-cell>
64     </fo:table-row>
65 </xsl:template>
66 </xsl:stylesheet>
    
```

PDF:

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Fußnoten](#)

## Fußnoten

Mit Hilfe von XSL-FO haben Sie die Möglichkeit, auf einfache Art und Weise Fußnoten zu erzeugen. Um die **automatische Platzierung** des Fußnoteninhalts am Seitenende kümmert sich der FO-Prozessor. Um eine Fußnote zu definieren, benötigen Sie in erster Linie das Element `footnote`. Dem `footnote`-Element werden exakt zwei Elemente untergeordnet: `inline` und `footnote-body`. Das `inline`-Element kennen Sie ja bereits und dient hier dazu, den Fußnotentext (im Fließinhalt) festzulegen. Dieser Text ist im Regelfall eine **hochgestellte fortlaufende Nummer** (also 1, 2, 3, 4, ...). Das `footnote-body`-Element wird dazu verwendet, den eigentlichen **Inhalt der Fußnote** zu definieren. Dafür muss dem Element ein `block`-Element untergeordnet werden.

### XSL-FO-Code:

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4    <fo:layout-master-set>
5      <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6        <fo:region-body region-name="inhalt" margin="2cm" />
7      </fo:simple-page-master>
8    </fo:layout-master-set>
9
10   <fo:page-sequence master-reference="DIN-A4">
11     <fo:flow flow-name="inhalt">
12       <fo:block>
13         Die Website Homepage-Webhilfe wurde im Jahr 2013 gegründet!
14       <fo:footnote>
15         <fo:inline baseline-shift="super">1</fo:inline>
16         <fo:footnote-body>
17           <fo:block>
18             <fo:inline baseline-shift="super">1</fo:inline>
19             Quelle: <fo:basic-link external-destination="https://news.homepage-webhilfe.de/">news.homepage-
20             webhilfe.de</fo:basic-link>
21           </fo:block>
22         </fo:footnote-body>
23       </fo:footnote>
24     </fo:flow>
25   </fo:page-sequence>
26 </fo:root>

```

PDF: 

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSL-FO](#) » [Erweiterte Ausrichtung](#)

## Erweiterte Ausrichtung

In XSL-FO gibt es, so wie in CSS auch, das sogenannte **Float-Konzept** (zu Deutsch *schwebend*, *schwimmend*). Mit Hilfe dieses Konzepts ist es möglich, **Elemente innerhalb eines Textflusses zu platzieren**. Ein typisches Beispiel dafür ist die Platzierung eines Bilds in einem Text. Um dieses Konzept zu verwenden, ordnen Sie ihrem Fließinhalt an der gewünschten Stelle das Element `float` unter. Diesem Element wird nun wiederum ein `block`-Element untergeordnet, in welchem Sie die Inhalte, welche schwebend platziert werden sollen, notieren. Über das Attribut `float` kann die **Ausrichtung des schwebend zu platzierenden Inhalts** gesteuert werden. Mögliche Werte sind `left` (Ausrichtung links), `right` (Ausrichtung rechts), `start` (Ausrichtung zur Start-Region) und `end` (Ausrichtung zur End-Region).

### XSL-FO-Code:

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4    <fo:layout-master-set>
5      <fo:simple-page-master master-name="DIN-A4" page-height="297mm" page-width="210mm">
6        <fo:region-body region-name="inhalt" margin="2cm" />
7      </fo:simple-page-master>
8    </fo:layout-master-set>
9
10   <fo:page-sequence master-reference="DIN-A4">
11     <fo:flow flow-name="inhalt">
12       <fo:block text-align="justify">
13         <fo:float float="right">
14           <fo:block padding-left="3mm" padding-bottom="3mm">
15             <fo:external-graphic src="../../../Bilder/Logo/Logo.jpg" content-width="100px" content-
16             height="100px" />
17           </fo:block>
18         </fo:float>
19         ...
20       </fo:block>
21     </fo:flow>
22   </fo:page-sequence>
23 </fo:root>

```

PDF:

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [DTD](#) » [Grundlagen](#)

## Grundlagen

### Inhalt dieser Seite:

1. Nutzen und Funktionsweise
2. Grundaufbau und Einbindung

DTD (Document Type Definition, zu Deutsch *Dokumenttypdefinition*) ist keine eigene Sprache, sondern vielmehr ein **Sprachbestandteil von XML** (aber auch von SGML). Eine DTD enthält **Regeln über den Aufbau und Inhalt** eines XML- oder SGML-Dokuments.

Da die Dokumenttypdefinition Teil der XML-Spezifikation ist, muss der **Syntax einer DTD** natürlich weitestgehend mit dem von XML (also vor allem in Bezug auf die spitzen Klammern) kompatibel sein. Es gibt jedoch trotzdem ein paar Unterschiede, welche Sie innerhalb diese Tutorials kennenlernen werden. Im Allgemeinen kann man sagen, dass der Syntax von DTD vereinfacht ist.

## Nutzen und Funktionsweise

Bei der Definition eines eigenen XML-Dokumentaufbaus stellt sich wohl zunächst die Frage, **warum man ein DTD braucht**. Die Antwort ist einfach: Man braucht nicht zwingend eine DTD. Die Verwendung einer Dokumenttypdefinition ist jedoch **sinnvoll, wenn man das Format veröffentlichen möchte** bzw. es an andere Personen weitergeben möchte. Entwerfen Sie ein XML-Datenformat zum Datenaustausch von Informationen innerhalb Ihrer Website, so werden Sie wohl kaum eine DTD entwerfen. Haben Sie hingegen ein Programm entwickelt, bei welchem Einstellungen in einer XML-Datei gespeichert werden (vorausgesetzt die Datei ist auch dazu gedacht ist, vom Benutzer „manuell“ angepasst zu werden), so ist es schon eher sinnvoll, eine DTD zu schreiben. Möchten Sie einen eigenen Standard entwerfen, so könnte man das Erstellen einer DTD schon fast als Pflicht ansehen.

Mit einer Dokumenttypdefinition lassen sich Regeln aufstellen, um den **Aufbau sowie den Inhalt eines oder mehrerer XML-Dokumente festzulegen**. Ein Beispiel: Sie haben ein XML-Dokument, welches ein Adressbuch repräsentiert. In diesem Adressbuch befinden sich nun mehrere Kontakte. Die Kontakte enthalten z. B. Informationen über den Vornamen, den Nachnamen, die Straße, die Hausnummer, die Postleitzahl, den Ort, die Telefonnummer und die E-Mail-Adresse. Es kann jedoch vorkommen, dass in Ihrem Adressbuch Kontakte enthalten sind, deren postalische Adresse Sie nicht kennen, sondern z. B. nur deren Telefonnummer. Um nun zum einen festzulegen, welche Elemente und Attribute es gibt und zum anderen zu definieren, wie diese verschachtelt sind, in welcher Reihenfolge diese zu erscheinen haben und wie oft bestimmte Elemente vorkommen dürfen / müssen, können Sie eine Dokumenttypdefinition verwenden.

Wird für eine XML-Datei keine DTD verwendet (was nicht zwingend erforderlich ist), so kann das Dokument zwar wohlgeformt, nicht aber gültig sein. Dies liegt daran, da ohne eine DTD nicht nachweisbar ist, ob der Aufbau und die enthaltenen Daten korrekt sind oder nicht. Bei Verwendung einer Dokumenttypdefinition sieht dies anders aus, denn durch die vorliegenden Regeln kann (z. B. durch einen XML-Parser) **ermittelt werden, ob der Aufbau sowie die Daten gültig sind**.

## Grundaufbau und Einbindung

Bei Verwendung einer DTD kann man selbst entscheiden, ob die DTD im eigenen XML-Dokument angegeben wird (XML-Deklarations-Attribut `standalone="yes"`) oder in einer externen Datei (meist mit der Dateierdung `.dtd`, XML-Deklarations-Attribut `standalone="no"`) ausgelagert wird. Um die **gleiche DTD für mehrere XML-Dateien nutzen** zu können, ohne die DTD kopieren zu müssen, ist es erforderlich, die DTD in eine eigene Datei auszulagern.

Betrachten wir zunächst den Fall, dass die DTD im eigenen XML-Dokument enthalten ist: Als erstes müssen wir die **Wurzel des Dokuments deklarieren**. Hierfür gilt der Syntax `<!DOCTYPE Wurzel [ ]>`, wobei `Wurzel` durch den Namen des Wurzelements zu ersetzen ist. Innerhalb der eckigen Klammern werden nun die weiteren DTD-Anweisungen (dazu später mehr) angegeben. Dabei wird i. d. R. auch auf die Einrückung geachtet.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3  <!DOCTYPE adressbuch [
4      <!-- Hier werden die weiteren Anweisungen notiert -->
5  ]>
6
7  <adressbuch>
8
9  </adressbuch>
    
```

Möchten Sie die DTD hingegen in ein eigenes Dokument auslagern, so wird in der XML-Datei lediglich der Verweis zu der DTD angegeben. Als Schema gilt hier `<!DOCTYPE Wurzel SYSTEM "pfad/zur/datei.dtd">`. Auch hier muss `Wurzel` durch den Namen des Wurzelements ersetzt werden.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
2
3  <!DOCTYPE adressbuch SYSTEM "adressbuch.dtd">
4
5  <adressbuch>
6
7  </adressbuch>
    
```

Neben dem oben verwendeten `SYSTEM`-Identifikator, gibt es noch den `PUBLIC`-Identifikator. Hierbei muss dann zwischen dem Schlüsselwort `PUBLIC` und dem Dateinamen ein **öffentlicher Name** (anzugeben in doppelten Anführungszeichen) notiert werden. Dies ermöglicht, dass Programme wie z. B. Webbrowser die DTD nicht erneut laden, sofern die benötigte DTD über den gleichen öffentlichen Namen verfügt. Verwendet wird diese Form z. B. bei dem Dokumententyp (engl. *document type*, oder kurz *doctype*) einer XHTML-Datei.

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
    
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [DTD](#) » [Elemente](#)

## Elemente

Um Elemente zu deklarieren, wird der Ausdruck `<!ELEMENT Name Inhalt>` verwendet. `Name` muss dabei durch den Elementnamen ersetzt werden. Als Wert für den Platzhalter `Inhalt` gibt es mehrere Möglichkeiten: `EMPTY` (Element enthält keinen Inhalt und ist einteilig), `ANY` (Element darf beliebigen Inhalt enthalten) oder die explizite Angabe des Elementinhalts.

```
1 | <!ELEMENT umbruch EMPTY>
```

Wird der **Inhalt explizit angegeben**, so muss dieser in runden Klammern notiert werden. Die Klammern dienen zur **Gruppierung** und können auch für weitere Gruppen innerhalb der „Hauptgruppe“ verwendet werden. Innerhalb der Gruppen kann nun entweder das Schlüsselwort `#PCDATA`, um auf einen textuellen Inhalt hinzuweisen, oder der Name eines Elements angegeben werden. Mehrere Inhalte werden durch Komma getrennt.

```
1 | <!ELEMENT adresse (strasse,hausnummer,plz,ort)>
```

Um die **Anzahl der Vorkommen** von Elementen zu steuern, gibt es in DTD drei wichtige Zeichen: `*`, `+` und `?`. Mit dem Sternzeichen `*` können Sie festlegen, dass das Element beliebig oft vorkommen darf (also keinmal, einmal oder mehrmals). Das Pluszeichen `+` legt fest, dass das Element mindestens einmal bis beliebig oft vorkommen darf. Mit dem Fragezeichen `?` kennzeichnen Sie ein Element als optional, d. h. es kann einmal vorkommen oder auch nicht. Mehrere Vorkommen sind nicht erlaubt. Wird keines der drei Zeichen angegeben, so muss das Element exakt einmal vorkommen.

```
1 | <!ELEMENT computer (maus,tastatur,monitor+)>
```

Um anzugeben, dass entweder das eine oder das andere Element vorkommen soll, wird der Senkrechtstrich `|` verwendet. Auch drei oder mehrere sogenannte **Alternativen** können angegeben werden.

```
1 | <!ELEMENT telefon (festnetz|handy)>
```

Die oben genannten Verfahren können auch nach Belieben gemischt werden.

```
1 | <!ELEMENT computer ((funktmaus|kabelmaus),tastatur,(touchmonitor|monitor)+)>
```

Oft möchten Sie, dass die Reihenfolge von Elementen nicht starr ist, sondern dass diese variieren kann. Hierfür notiert man alle Elemente als Alternativen, gruppiert diese und versieht die Gruppe mit dem Sternzeichen. Man spricht hier auch von **gemischtem Inhalt**.

```
1 | <!ELEMENT kontakt (name|strasse|hausnummer|plz|ort|telefon|email)*>
```

Hier nun noch ein vollständiges Beispiel:

```
1 | <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 |
3 | <!DOCTYPE kontakt [
4 |   <!ELEMENT kontakt (name,adresse,telefon*,email?)>
5 |   <!ELEMENT name (vorname,nachname)>
6 |   <!ELEMENT vorname (#PCDATA)>
7 |   <!ELEMENT nachname (#PCDATA)>
8 |   <!ELEMENT adresse (strasse,hausnummer,plz,ort)>
9 |   <!ELEMENT strasse (#PCDATA)>
10 |  <!ELEMENT hausnummer (#PCDATA)>
11 |  <!ELEMENT plz (#PCDATA)>
12 |  <!ELEMENT ort (#PCDATA)>
13 |  <!ELEMENT telefon (#PCDATA)>
14 |  <!ELEMENT email (#PCDATA)>
15 | ]>
16 |
17 | <kontakt>
18 |   <name>
19 |     <vorname>Max</vorname>
20 |     <nachname>Mustermann</nachname>
21 |   </name>
22 |   <adresse>
23 |     <strasse>Musterstraße</strasse>
24 |     <hausnummer>123</hausnummer>
25 |     <plz>12345</plz>
26 |     <ort>Musterstadt</ort>
27 |   </adresse>
28 |   <telefon>01234 / 56789</telefon>
29 |   <telefon>0987 / 654321</telefon>
30 |   <email>m.mustermann@example.com</email>
31 | </kontakt>
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [DTD](#) » [Attribute](#)

## Attribute

Um Attribute bzw. Listen von Attributen und deren möglichen Werte zu deklarieren, wird der Ausdruck `<!ATTLIST Elementname Attribute>` verwendet. Der Platzhalter `Elementname` wird dabei durch den Namen des Elements ersetzt, d. h. **Attribute bzw. Attributlisten sind immer einem Element zugeordnet**. Gibt es ein Attribut, welches in mehreren Elementen vorkommen kann, so muss dieses auch mehrmals deklariert werden. Der Platzhalter `Attribute` wird durch die Definition eines oder mehrerer Attribute ersetzt. Verfügt ein Element über keine Attribute, so wird der `ATTLIST`-Ausdruck weggelassen. Um ein Attribut zu definieren, benötigen Sie drei Informationen: Attributname, Typ und Vorgabe. Für den **Typ** gibt es ein paar unterschiedliche Werte. In der Praxis werden jedoch nur wenige davon eingesetzt: `CDATA` (Wert enthält Zeicheninhalt), `ID` (Wert ist eine eindeutige ID), `IDREF` (Wert ist die ID eines anderen Attributwerts), `IDREFS` (Wert enthält IDs von anderen Attributwerten, getrennt durch Leerzeichen) und Aufzählungen. **Aufzählungen** werden in runden Klammern angegeben und durch Senkrechtstriche `|` getrennt. Der letzte Wert bei der Deklaration eines Attributs ist die **Vorgabe**. Als Vorgabewerte stehen `#REQUIRED` (Attribut ist erforderlich), `#IMPLIED` (Attribut ist optional), `#FIXED "Festwert"` (Attribut hat einen fest definierten Wert) und `"Standardwert"` (Attribut hat einen Standardwert, welcher verwendet wird, wenn das Attribut weggelassen wird) zur Verfügung. Hierzu nun ein Beispiel:

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3  <!DOCTYPE adressbuch [
4    <!ELEMENT adressbuch (kontakt)*>
5    <!ELEMENT kontakt (name,telefon,email)>
6    <!ATTLIST kontakt
7      typ (privat|arbeit) "privat"
8      geschlecht (m|w) #REQUIRED
9      alter CDATA #IMPLIED
10   >
11   <!ELEMENT name (#PCDATA)>
12   <!ELEMENT telefon (#PCDATA)>
13   <!ELEMENT email (#PCDATA)>
14 ]>
15
16 <adressbuch>
17   <kontakt typ="privat" geschlecht="m" alter="17">
18     <name>Max Mustermann</name>
19     <telefon>01234 / 56789</telefon>
20     <email>m.mustermann@example.com</email>
21   </kontakt>
22   <kontakt geschlecht="w" alter="24">
23     <name>Maria Musterfrau</name>
24     <telefon>01235 / 46789</telefon>
25     <email>m.musterfrau@example.com</email>
26   </kontakt>
27   <kontakt typ="arbeit" geschlecht="m">
28     <name>Peter Müller</name>
29     <telefon>09876 / 54321</telefon>
30     <email>mueller@example.com</email>
31   </kontakt>
32 </adressbuch>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [DTD](#) » [Entitäten](#)

## Entitäten

Mit Hilfe einer Entität (engl. *entity*) ist es möglich, bestimmte Texte oder Ausdrücke (auch als Bausteine bezeichnet) zu **speichern, um diese später wieder weiter verwenden zu können**. Dadurch sollen zum einen bestimmte Inhalte vordefiniert werden können, um diese dann später in der XML-Datei nutzen zu können, und zum anderen um die DTDs kürzer und übersichtlicher zu gestalten. Bei Entitäten unterscheidet man zwischen „normalen“ Entitäten und Parameter-Entitäten.

Die „normalen“ Entitäten werden dazu verwendet, Textbausteine zu definieren oder Zeichen zu benennen. Die **Zeichen-Entität** (also die Benennung von Zeichen) kennen Sie vielleicht bereits von HTML (z. B. wird `&larrr;` in `&#8592;` übersetzt, wodurch bei der Anzeige `→` angezeigt wird). Im Allgemeinen kann man also sagen, Entitäten werden dazu verwendet, „lesbare“ **Kurzformen zu erzeugen**. Das Schema einer Entität lautet `<!ENTITY Name "Wert">`. Um die Entität in einem XML-Dokument oder in einer anderen Entität zu nutzen, wird dann `&Name;` notiert. Hierzu ein kurzes Beispiel:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3 <!DOCTYPE text [
4   <!ELEMENT text (#PCDATA)>
5   <!ENTITY autor "Homepage-Webhilfe">
6   <!ENTITY jahr "2017">
7   <!ENTITY copyright "Copyright &jahr; by &autor;">
8 ]>
9
10 <text>
11   Die DTD für dieses Dokument wurde von &autor; im Jahr &jahr; erstellt!
12 </text>
```

Eine weitere Form von Entitäten sind **Parameter-Entitäten**, welche dazu gedacht sind, DTD-Bausteine zu definieren. Sicherlich haben Sie schon gemerkt, dass Sie beim Entwurf einer DTD, öfters mal die gleichen Ausdrücke notieren. Dies kann z. B. bei Attributen auftreten, da die Attributlisten für jedes Element neu definiert werden müssen. Parameter-Entitäten unterscheiden sich nur geringfügig von „normalen“ Entitäten: Bei der Deklaration muss zwischen dem Schlüsselwort `ENTITY` und dem Namen ein Prozentzeichen `%` angegeben werden. Des Weiteren wird zum „Aufruf“ bzw. zur Einbindung nicht `&Name;` notiert, sondern `%Name;`.

### DTD-Datei:

```

1 <!ENTITY % festplatte "speicher CDATA #REQUIRED">
2 <!ELEMENT computer (prozessor, arbeitsspeicher, hdd*, ssd*)>
3 <!ELEMENT prozessor (#PCDATA)>
4 <!ELEMENT arbeitsspeicher (#PCDATA)>
5 <!ELEMENT hdd (#PCDATA)>
6 <!ATTLIST hdd
7   %festplatte;
8   >
9 <!ELEMENT ssd (#PCDATA)>
10 <!ATTLIST ssd
11   %festplatte;
12   >
```

### XML-Datei:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3 <!DOCTYPE computer SYSTEM "computer.dtd">
4
5 <computer>
6   <prozessor>Intel Core i7-4790K</prozessor>
7   <arbeitsspeicher>Kingston DDR3 8GB 1600MHz</arbeitsspeicher>
8   <hdd speicher="1000">Western Digital Blue WD10EZEX</hdd>
9   <hdd speicher="500">Toshiba DT01ACA050</hdd>
10  <ssd speicher="250">Samsung 840 EVO</ssd>
11 </computer>
```

Bei komplexen DTDs kann es sinnvoll sein, **Ausdrücke in andere Dateien auszulagern**. Hierzu werden ebenfalls Entitäten benutzt. Als Wert wird die URI angegeben. Zudem wird zwischen Name und Wert der `SYSTEM`- oder `PUBLIC`-Identifikator angegeben (wie bei der Einbindung einer DTD in eine XML-Datei). Wird der `PUBLIC`-Identifikator verwendet, so muss auch noch ein öffentlicher Name angegeben werden.

```

1 <!-- Deklaration -->
2 <!ENTITY % erweiterung SYSTEM "erweiterung.dtd">
3
4 <!-- Aufruf / Einbindung -->
5 %erweiterung;
```

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

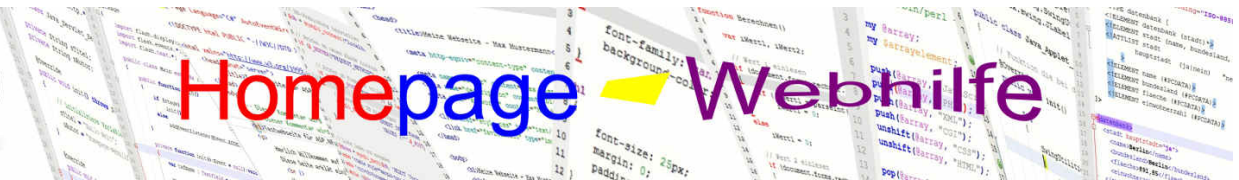
#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSD](#) » [Grundlagen](#)

## Grundlagen

Die XML-basierende Sprache XSD (*XML Schema Definition*, oftmals auch nur als *XML Schema* bezeichnet) ist eine **Schemasprache, die den Aufbau und Inhalt von XML-Dokumenten definieren soll**. Im Gegensatz zu den klassischen Dokumenttypdefinitionen (DTDs) bieten XSD-Skripte jedoch einige Vorteile.

Die Definition eines solchen sogenannten XML-Schemas erfolgt in einer separaten Datei (üblicherweise mit der Dateiendung `.xsd`). Die Vorgehensweise in XSD ist dabei etwas anders als bei DTD, da hier **Datentypen** definiert werden, welche wiederum einem Element oder Attribut zugeordnet werden. Neben der Verwendung der bereits vorhandenen Datentypen (z. B. für Zeichenketten oder Zahlen), können eigene Datentypen (z. B. durch bestimmte Einschränkungen) definiert werden. Zudem lässt sich die Elementanzahl, der Element- und Attributinhalt und vieles mehr **wesentlich genauer steuern**. Dies hat zur Folge, dass XSD-Dokumente i. d. R. umfangreicher als DTDs sind, jedoch auch eine deutlich höhere Anzahl an Möglichkeiten (z. B. Einschränkungen oder Ableitungen) zur Validierung bestehen.

## Nutzen und Funktionsweise

Wie bei DTDs stellt sich zunächst natürlich folgende Frage: **Brauche ich ein XSD-Dokument?** Um dies zu erklären, greifen wir zunächst auf die Begriffe wohlgeformt und gültig zurück. Ein XML-Dokument ist wohlgeformt, wenn die „Regeln von XML“ eingehalten werden. Zu diesen Regeln zählt z. B., dass das Dokument nur über ein Wurzelement verfügt, jedes geöffnete Element auch wieder geschlossen wird (in umgekehrter Reihenfolge) usw.. Das Dokument ist dadurch jedoch noch nicht gültig. Um die **Gültigkeit eines XML-Dokuments** zu erreichen, ist es notwendig, dass das Dokument über eine Definition (DTD) oder ein XML Schema (XSD) verfügt, dessen Regeln dann auch eingehalten werden.

Was heißt das jetzt konkret? Verwenden Sie keine DTD oder XSD, so kann Ihr Dokument nicht gültig (auch als valide bezeichnet) sein, da dessen Struktur und Inhalt nicht geprüft werden kann. Verfügt Ihr XML-Dokument hingegen über eine solche Definition, so **kann das Dokument geprüft werden** und ist, falls die Prüfung erfolgreich war, gültig (auch als valide bezeichnet). Daraus ergibt sich jetzt natürlich folgende Frage: Brauchen Sie für Ihr Vorhaben ein DTD- oder XSD-Skript? Grundsätzlich kann man sagen, dass eine DTD oder XSD **immer dann sinnvoll ist, wenn mit den XML-Dateien dritte Personen in Kontakt kommen** bzw. diese ändern oder sogar selber erstellen sollen. Beispiele dafür wären, Konfigurationsdateien für Programme oder „Skripte“, mit welchen Programme gesteuert werden. Für den Datenaustausch auf der eigenen Website hingegen werden Sie wohl kaum eine DTD oder XSD erstellen.

Ein XML Schema soll also, wie eine Dokumenttypdefinition auch, **die Struktur und den Inhalt des XML-Dokuments auf Gültigkeit überprüfen**. Dafür werden in XSD in erster Linie **Datentypen** festgelegt. Diese Datentypen sind von bereits vorhandenen Basis-Datentypen (z. B. Zeichenkette oder Zahl) abzuleiten und können **über bestimmte Definitionen weiter eingeschränkt** werden. Neben solchen Datentypen, welche auch als einfache Typen bezeichnet werden, gibt es noch komplexe Typen, die dazu genutzt werden, Datentypen zu definieren, welchen Elemente untergeordnet werden, die leer sind oder die Attribute besitzen.

## Grundaufbau

In erster Linie ist ein XSD-Dokument nichts anderes als ein normales XML-Dokument, d. h. auch hier gibt es eine optionale XML-Deklaration sowie ein Wurzelement. Das Wurzelement in XML Schema ist `schema`. Wie in XSLT und XSL-FO wird i. d. R. ein Namensraumpräfix verwendet. Bei XSD ist dies normalerweise `xs` oder `xsd`. Das Weglassen des Namensraumpräfixes ist nicht zu empfehlen, da dadurch Konflikte zwischen eigenen Namen und den vordefinierten Namen von XSD entstehen können (z. B. in Bezug auf Datentypen). Abhilfe könnte hier dann die Verwendung eines Namensraums für die „eigenen“ Namen schaffen. Da die Werte von Attributen im Regelfall dem Standardnamensraum angehören, ist für Features wie die Referenzierung von Elementen oder Datentypen, welche Sie im Normalfall immer benötigen, notwendig, den Standardnamensraum mittels des `xmlns`-Attributs festzulegen. Alternativ wäre es auch möglich, einen eigenen Namensraumpräfix zu definieren. Sie müssten dann jedoch bspw. bei der Referenzierung auf einen eigenen Datentyp (dazu später mehr) den definierten Namensraumpräfix davorstellen. Wir empfehlen Ihnen jedoch die Verwendung des Standardnamensraums. Das `schema`-Element verfügt des Weiteren noch über ein paar optionale Attribute: `version` (Versionsangabe von XSD, i. d. R. `1.0`), `targetNamespace` (Namensraum, der für die XML-Dokumente, welche dieses Schema verwenden, zulässig ist - meistens wird hier der gleiche Namensraum wie für den Standardnamensraum angegeben), `elementFormDefault` (Angabe zur Element-Qualifizierung) und `attributeFormDefault` (Angabe zur Attribut-Qualifizierung).

Für die Attribute `elementFormDefault` und `attributeFormDefault` sind die Werte `qualified` (Element / Attribut muss qualifiziert sein) und `unqualified` (Element / Attribut muss nicht qualifiziert sein) möglich. Der Standardwert der beiden Attribute ist `unqualified`. Eine kurze Erklärung dazu: In XSD wird zwischen **globalen und lokalen Elementen** unterschieden. Als globale Elemente gelten alle Elemente, welche direkt im `schema`-Element des XSD-Dokuments angegeben sind. Auch Elemente, die sich auf ein anderes Element bzw. einen Datentyp, welches direkt im `schema`-Element angegeben wurde, beziehen, gelten als globale Elemente. Alle anderen Elemente sind lokal. Doch was hat das ganze nun mit der Qualifizierung eines Elements zu tun? Durch den Wert `qualified` im Attribut `elementFormDefault` wird festgelegt, dass jedes XML-Element (egal ob global oder lokal) über einen vollständigen (auch als qualifiziert bezeichnet) Namen verfügen muss, d. h. der **Name muss sich aus Namensraumpräfix und Elementname zusammensetzen**. Der Namensraumpräfix fällt natürlich weg, wenn ein Standardnamensraum verwendet wird. Anders herum heißt dies also, dass bei Verwendung des Werts `unqualified` nicht jeder Elementname (sondern nur Namen von globalen Elementen) über einen vollständigen / qualifizierten Namen verfügen muss. Lokale Elemente können dann also z. B. ohne Namensraumpräfix angegeben werden. Dies ist in der Praxis jedoch selten erwünscht, da dadurch nicht mehr direkt erkennbar ist, welches Element zu welchem Namensraum gehört. Zudem ist bei dem Wert `unqualified` die Verwendung eines Standardnamensraums unzulässig. Der einzige Vorteil dieser Variante ist, dass bei Verwendung mehrerer Namensräume die Angabe von Präfixen gespart werden kann. Wie sieht das Ganze aber nun bei Attributen aus? Bei Attributen ist es irrelevant, ob es sich um ein globales oder lokales Attribut handelt. Hier geht es lediglich um die **Notation des Attributs**: Der Wert `qualified` bedeutet, dass der Attributname aus Namensraumpräfix und dem eigentlichen Attributnamen bestehen muss. Der Wert `unqualified` hingegen bedeutet, dass der Name nicht qualifiziert ist und somit nur aus dem Attributnamen besteht. Wird ein Standardnamensraum verwendet, so ist der Wert `qualified` ungültig (also umgekehrt wie bei Elementen!). Als Resultat lässt sich sagen, dass i. d. R. für das Attribut `elementFormDefault` der Wert `qualified` und für das Attribut `attributeFormDefault` der Wert `unqualified` (keine explizite Angabe notwendig, da es der Standardwert ist) verwendet wird.

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0" xmlns="https://www.homepage-webhilfe.de/XML/XSD/"
4  targetNamespace="https://www.homepage-webhilfe.de/XML/XSD/" elementFormDefault="qualified">
5  </xs:schema>
    
```

**Wichtig:** Im obigen Beispiel haben wir bezüglich den Namensräumen genau das gemacht, was wir im Text oben als empfohlene Variante genannt haben: Für das Schema selbst wird der Namensraumpräfix `xs` verwendet. Zudem wird ein Standardnamensraum verwendet (`xmlns`-Attribut), welcher den gleichen Wert wie das Attribut `targetNamespace` hat. Dies ist die gängige Praxis.

### Inhalt dieser Seite:

1. Nutzen und Funktionsweise
2. Grundaufbau
3. Einbindung

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSD](#) » [Grundlagen](#)

## Einbindung

Wenn Sie eine XML-Datei erstellen, welche einem bestimmten (oder auch dem eigenen) Schema angehört, so müssen Sie eine **Verbindung zwischen XML- und XSD-Datei** herstellen. Dafür verwenden Sie im „Basiselement“ (meistens ist dies das Wurzelement) Ihres XML-Dokuments das Attribut `schemaLocation`. Als Wert geben Sie die URI zur XSD-Datei an. Das Attribut `schemaLocation` gehört zum XML Schema Instanz Namensraum. Dieser wird in die XML-Datei meistens über den Namensraumpräfix `xsi` „eingebunden“.

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3 <adressbuch xmlns="https://www.homepage-webhilfe.de/XML/XSD/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="adressbuch.xsd">
4
5 </adressbuch>
  
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » XML & Co. » XSD » Einfache Typen

## Einfache Typen

### Inhalt dieser Seite:

1. Listen
2. Vereinigungen

Als einfacher Typ wird ein Datentyp bezeichnet, welcher lediglich „einfache“ Werte (z. B. eine Zeichenkette / einen Text oder eine Zahl) annehmen kann. Die definierten Datentypen können dann später sowohl **in Attributen als auch in Elementen verwendet werden**. Attribute unterstützen jedoch nur einfache Datentypen. Als Basis für alle Datentypen dienen die sogenannten Basisdatentypen. Die **Basisdatentypen** wurden vom W3C festgelegt und sind fester Bestandteil der XSD-Spezifikation. Die folgende Tabelle zeigt die wichtigsten verfügbaren Basisdatentypen:

Name	Beschreibung	Beispiele
string	Zeichenkette (bestehend aus 0, 1 oder mehreren Zeichen)	Hallo
boolean	Wahrheitswert	true, 1, false, 0
integer	Zahlenwert ggf. mit Vorzeichen	-46, 0, 875
nonNegativeInteger	positiver Zahlenwert oder 0	0, 72, 875
positiveInteger	positiver Zahlenwert	1, 72, 875
nonPositiveInteger	negativer Zahlenwert oder 0	-463, -97, 0
negativeInteger	negativer Zahlenwert	-463, -97, -1
float	Fließkommazahl 32bit	2.4
double	Fließkommazahl 64bit	3.14159
decimal	Dezimalzahl	69, 581.4603
date	Datumsangabe	2017-01-29
time	Uhrzeitangabe	20:30:45
dateTime	Datumsangabe mit Uhrzeit	2017-01-29T20:30:45
ID	eindeutiger Name	arbeitsspeicher
IDREF	Referenz auf einen eindeutigen Namen	arbeitsspeicher
IDREFS	Referenz auf einen oder mehrere eindeutige(n) Namen	prozessor arbeitsspeicher

**Wichtig:** Verwenden Sie im XSD-Skript einen Namensraum für die Bestandteile von XSD selbst, was i. d. R. der Fall ist, so müssen Sie bei der Angabe eines solchen vordefinierten Datentyps auch den Namensraumpräfix mit angeben (z. B. `xs:string` an Stelle von `string`).

Um nun einen **einfachen Datentyp zu deklarieren**, nutzen wir das Element `simpleType`. Mit dem Attribut `name` lässt sich der Name des definierten Datentyps festlegen. Dieser Name kann dann später bei der Datentypangabe in Elementen und Attributen innerhalb des XSD-Skripts verwendet werden. Innerhalb des `simpleType`-Elements notieren Sie nun das `restriction`-Element. Über das Attribut `base` können Sie die „Grundlage“ des Datentyps festlegen. Hier können Sie einen der oberen Datentypen (inklusive dem Namensraumpräfix) oder auch einen eigendefinierten Datentyp angeben. Das `restriction`-Element dient eigentlich dazu, einen Datentyp einzuschränken (z. B. in der Länge oder im Wertebereich). Darauf gehen wir jedoch erst **später** ein. Die Angabe eines eigenen Datentyps als Basis wäre daher aus jetziger Sicht sinnlos bzw. unnötig. Das `restriction`-Element hat fürs Erste also keine Kindelemente und darf deshalb auch als leeres Element notiert werden.

```
1 <xs:simpleType name="speicher">
2   <xs:restriction base="xs:integer" />
3 </xs:simpleType>
```

Nachdem wir einen Datentyp definiert haben, können wir den Datentyp in einem Element verwenden. Um ein **Element zu deklarieren**, benötigen wir das `element`-Element. Über das Attribut `name` wird der Name des Elements (dieser wird dann auch in der XML-Datei als Elementname verwendet) und über das `type`-Attribut der dazugehörige Datentyp festgelegt.

```
1 <xs:element name="arbeitsspeicher" type="speicher" />
```

Nachdem unser selbst definierter Datentyp bisher nichts anderes als der Datentyp `xs:integer` ist, hätten wir die Definition des Datentyps auch weglassen können und bei der Element-Definition direkt `xs:integer` als Datentyp angeben können.

```
1 <xs:element name="arbeitsspeicher" type="xs:integer" />
```

Die an dieser Stelle vielleicht etwas ausführliche und „umständliche“ Erklärung soll Ihnen jedoch später den Einstieg in die Themen [Einschränkungen](#) und [Ableitung](#) erleichtern. Eine weitere Variante zur Datentypdefinition ist die Definition eines **anonymen Datentyps**. Anonyme Datentypen haben keinen Namen und werden direkt bei der Element- oder Attribut-Deklaration angegeben. Sinnvoll ist diese Variante nur dann, wenn Sie den dort definierten Datentyp an keiner anderen Stelle benötigen. Es gibt jedoch Personen, die von dieser Art und Weise generell komplett abraten. Dies ist letztendlich aber auch wieder eine Geschmacksfrage. Das Beispiel von oben würde mit einem anonymen simplen Datentyp wie folgt aussehen:

```
1 <xs:element name="arbeitsspeicher">
2   <xs:simpleType>
3     <xs:restriction base="xs:integer" />
4   </xs:simpleType>
5 </xs:element>
```

Bei größeren oder komplexeren XML-Skripten kommt es immer wieder vor, dass Sie das gleiche Element mehrmals, jedoch an unterschiedlichen Stellen im XML-Baum, benötigen. Für ein solches Vorhaben dient eine **Referenz**. Dabei notieren Sie Ihr Element einmal global (also direkt innerhalb des `schema`-Elements) und können es dann mittels des `element`-Elements und des `ref`-Attributs referenzieren. Die Referenzierung des Elements `arbeitsspeicher` würde wie folgt aussehen:

```
1 <xs:element ref="arbeitsspeicher" />
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSD](#) » [Einfache Typen](#)

Das `element`-Element verfügt des Weiteren noch über einige **Attribute, mit welchen Einstellungen festgelegt werden können**. Die wichtigsten Attribute sind `default` (Standardwert, falls das Element keinen Wert hat), `fixed` (Festwert - andere Werte sind nicht erlaubt), `minOccurs` (minimale Anzahl an Vorkommnissen) und `maxOccurs` (maximale Anzahl an Vorkommnissen). Der Standardwert für die Attribute `minOccurs` und `maxOccurs` ist 1. Für das Attribut `maxOccurs` kann auch der Wert `unbounded` verwendet werden, um somit die maximale Anzahl an Elementen nicht zu begrenzen.

```
1 | <xs:element name="festplatte" type="speicher" maxOccurs="6" />
```

## Listen

Als Liste bezeichnet man einen speziellen Datentyp, dessen **Wert aus mehreren Werten getrennt durch ein Leerzeichen bestehen**. Um eine Liste zu definieren, benötigen Sie das Element `list`, welches einem `simpleType`-Element untergeordnet wird. Über das Attribut `itemType` wird der Datentyp der einzelnen Werte (auch als Item bezeichnet) festgelegt.

```
1 | <xs:element name="zahlen">
2 |   <xs:simpleType>
3 |     <xs:list itemType="xs:integer" />
4 |   </xs:simpleType>
5 | </xs:element>
```

**Übrigens:** An Stelle des Attributs `itemType` kann dem `list`-Element auch ein `simpleType`-Element (anonymer Datentyp) untergeordnet werden.

## Vereinigungen

Ein weiterer simpler Typ ist die Vereinigung (engl. *union*). Bei der Vereinigung werden **zwei oder mehrere Datentypen miteinander verbunden**, d. h. der Wert, der diesen Datentyp verwendet, darf einen Wert annehmen, der zu einem der angegebenen Datentypen der Vereinigung gehört. Um die Vereinigung zu deklarieren, wird das `union`-Element verwendet. Die Datentypen werden durch Leerzeichen getrennt im Attribut `memberTypes` angegeben.

```
1 | <xs:element name="zeitangabe">
2 |   <xs:simpleType>
3 |     <xs:union memberTypes="xs:date xs:dateTime" />
4 |   </xs:simpleType>
5 | </xs:element>
```

**Übrigens:** Sie können an Stelle des `memberTypes`-Attributs auch `simpleType`-Elemente innerhalb des `union`-Elements notieren.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSD](#) » [Komplexe Typen](#)

## Komplexe Typen

Neben den einfachen Typen gibt es in XSD auch noch komplexe Typen. Ein komplexer Datentyp **kommt lediglich bei Elementen zum Einsatz** und erlaubt es, Elemente mit untergeordneten Elementen oder einteilige Elemente zu definieren. Elemente, die Attribute besitzen, müssen ebenfalls als komplexe Typen definiert werden. Die Deklaration eines komplexen Typs erfolgt mit dem Element `complexType`. Der grundlegende Syntax unterscheidet sich nicht von der Deklaration eines einfachen Typs mittels `simpleType`. Die Notation eines anonymen komplexen Datentyps ist ebenfalls möglich.

### Inhalt dieser Seite:

1. Sequenzen
2. Auswahl
3. Attribute
4. Leere Elemente
5. Text-Elemente

## Sequenzen

Die einfachste Art eines komplexen Typs ist die Sequenz. Eine Sequenz **besteht aus mehreren Elementen**. Möchten Sie eine Sequenz definieren, so müssen Sie dem Element `complexType` ein `sequence`-Element unterordnen. Innerhalb des `sequence`-Elements werden dann die einzelnen Elemente angegeben (entweder direkt oder über eine Referenz).

### XML-Datei:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <kontakt xmlns="https://www.homepage-webhilfe.de/XML/XSD/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="kontakt.xsd">
4   <name>Max Mustermann</name>
5   <strasse>Musterstraße</strasse>
6   <hausnr>123</hausnr>
7   <plz>12345</plz>
8   <ort>Musterstadt</ort>
9 </kontakt>

```

### XSD-Datei:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0" xmlns="https://www.homepage-webhilfe.de/XML/XSD/"
  targetNamespace="https://www.homepage-webhilfe.de/XML/XSD/" elementFormDefault="qualified">
4   <xs:element name="kontakt">
5     <xs:complexType>
6       <xs:sequence>
7         <xs:element name="name" type="xs:string" />
8         <xs:element name="strasse" type="xs:string" />
9         <xs:element name="hausnr" type="xs:string" />
10        <xs:element name="plz" type="xs:string" />
11        <xs:element name="ort" type="xs:string" />
12      </xs:sequence>
13    </xs:complexType>
14  </xs:element>
15 </xs:schema>

```

**Wichtig:** In dem Beispiel von oben muss jedes Element der Sequenz exakt einmal vorkommen. Möchten Sie dies ändern, so müssen Sie die Attribute `minOccurs` und `maxOccurs` der `element`-Elemente ändern. Die Reihenfolge muss jedoch in allen Fällen eingehalten werden.

**Übrigens:** Bei einer Sequenz ist es nicht erlaubt, dass vor, zwischen oder nach den Elementen ein Text notiert wird. Möchten Sie dies erlauben, so müssen Sie das Attribut `mixed` im `complexType`-Element mit dem Wert `true` belegen.

## Auswahl

Mit dem Element `choice` können Sie in XSD eine Auswahl definieren. Eine Auswahl ist ein komplexer Typ, welcher angibt, dass **eines der angegebenen Elemente notiert werden muss**.

```

1 <xs:element name="datentraeger">
2   <xs:complexType>
3     <xs:choice>
4       <xs:element name="hdd" type="xs:string" />
5       <xs:element name="ssd" type="xs:string" />
6       <xs:element name="usbstick" type="xs:string" />
7     </xs:choice>
8   </xs:complexType>
9 </xs:element>

```

Wie beim `element`-Element auch, gibt es beim `choice`-Element die Attribute `minOccurs` und `maxOccurs`, um somit die **Anzahl der Vorkommnisse** festzulegen. Dies hat zur Folge, dass Sie im XML-Element mehrmals zwischen einem der definierten Elemente auswählen können.

### XML-Datei:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <datentraeger xmlns="https://www.homepage-webhilfe.de/XML/XSD/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="datentraeger.xsd">
4   <hdd>Western Digital Blue WD10EZEX</hdd>
5   <ssd>Samsung 840 EVO</ssd>
6   <usbstick>SanDisk Cruzer Fit 32GB</usbstick>
7   <hdd>Toshiba DT01ACA050</hdd>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSD](#) » [Komplexe Typen](#)

```
8 | </datentraeger>
```

## XSD-Datei:

```
1 | <?xml version="1.0" encoding="UTF-8" ?>
2 |
3 | <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0" xmlns="https://www.homepage-webhilfe.de/XML/XSD/"
4 | targetNamespace="https://www.homepage-webhilfe.de/XML/XSD/" elementFormDefault="qualified">
5 |   <xs:element name="datentraeger">
6 |     <xs:complexType>
7 |       <xs:choice maxOccurs="unbounded">
8 |         <xs:element name="hdd" type="xs:string" />
9 |         <xs:element name="ssd" type="xs:string" />
10 |        <xs:element name="usbstick" type="xs:string" />
11 |       </xs:choice>
12 |     </xs:complexType>
13 |   </xs:element>
14 | </xs:schema>
```

Übrigens: Diese Variante kann in etwas angepasster Form auch dazu verwendet werden, die Reihenfolge von Elementen im XML-Baum dynamisch zu gestalten.

## Attribute

Die Definition eines Attributs erfolgt mittels des Elements `attribute`. Das `attribute`-Element wird dabei i. d. R. direkt im `complexType`-Element notiert und ist dem `element`-Element sehr ähnlich, denn auch hier gibt es die Attribute `name`, `type`, `default` und `fixed`. Auch die Referenzierung von Attributen mittels des `ref`-Attributs sowie die Deklaration eines anonymen Datentyps (dabei fällt das `type`-Attribut weg und der Datentyp wird im Elementinhalt deklariert) sind möglich.

```
1 | <xs:attribute name="speicher" type="xs:integer" />
```

**Wichtig:** Denken Sie daran, dass als Datentyp für Attribute nur einfache Typen zulässig sind.

Erforderlich Sie festlegen, **ob ein Attribut vorkommen muss**, so können Sie das Attribut `use` verwenden. Als Wert für das `use`-Attribut sind `required` (Attribut ist erforderlich), `optional` (Attribut kann, muss aber nicht vorkommen - dies ist der Standardwert) und `prohibited` (Attribut darf nicht angegeben werden) möglich.

```
1 | <xs:attribute name="speicher" type="xs:integer" use="required" />
```

Übrigens: Der Wert `prohibited` wird dazu verwendet, Attribute aus einem bestehenden Datentyp wieder zu entfernen. Mehr dazu erfahren Sie im [Thema Ableitung](#).

## Leere Elemente

Möchten Sie mit XSD ein leeres Element deklarieren, so können Sie das Element `complexType` einfach leer lassen. Soll das Element über Attribute verfügen, so können Sie dem `complexType`-Element natürlich trotzdem `attribute`-Elemente unterordnen.

```
1 | <xs:element name="datentraeger">
2 |   <xs:complexType>
3 |     <xs:attribute name="speicher" type="xs:integer" use="required" />
4 |   </xs:complexType>
5 | </xs:element>
```

## Text-Elemente

Im Regelfall sind komplexen Elementen weitere Elemente untergeordnet. Möchten Sie einen komplexen Datentyp definieren, dem lediglich ein direkter Inhalt (z. B. ein Text oder eine Zahl) untergeordnet werden soll (z. B. weil das Element über Attribute verfügen soll), so benötigen Sie das Element `simpleContent`. Diesem wird das `extension`-Element untergeordnet, welches eigentlich für die Ableitung genutzt wird (dazu [später mehr](#)). Mit Hilfe des Attributs `base` legen Sie den Basistyp (z. B. `xs:string` oder `xs:integer`) fest. Soll das Element über Attribute verfügen, so können Sie die `attribute`-Elemente dem `extension`-Element unterordnen.

```
1 | <xs:element name="speicher">
2 |   <xs:complexType>
3 |     <xs:simpleContent>
4 |       <xs:extension base="xs:string">
5 |         <xs:attribute name="groesse" type="xs:integer" />
6 |       </xs:extension>
7 |     </xs:simpleContent>
8 |   </xs:complexType>
9 | </xs:element>
```

Ein gültiger XML-Syntax zum obigen Ausschnitt eines XML Schemas würde wie folgt aussehen:

```
1 | <speicher groesse="250">Samsung 840 EVO</speicher>
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSD](#) » [Einschränkungen](#)

## Einschränkungen

Mit Hilfe des `restriction`-Elements können Sie in XSD Einschränkungen definieren. Einschränkungen beziehen sich dabei immer auf einen bereits bestehenden Datentyp (dies kann ein selbst definierter Datentyp oder aber auch einer der vordefinierten Basisdatentypen sein). Dieser wird über das Attribut `base` festgelegt. Dem `restriction`-Element können dann verschiedene Elemente untergeordnet werden (dazu gleich mehr), um die eigentlichen Einschränkungen festzulegen. In diesem Thema lernen Sie lediglich die Einschränkung von einfachen Typen kennen. Im [Thema Ableitung](#) gehen wir dann noch auf die Einschränkung von komplexen Datentypen ein.

### Inhalt dieser Seite:

1. Länge
2. Wertebereich
3. Dezimalstellen
4. Listenwerte
5. Reguläre Ausdrücke

### Länge

Die Elemente `minLength`, `maxLength` und `length` werden dazu verwendet, die **Länge von Zeichenketten oder Listen** festzulegen. Mit den Elementen `minLength` und `maxLength` können Sie eine **Unter- und Obergrenze** für die Zeichenkette oder Liste festlegen. Das `length`-Element legt die **explizite Länge** fest, d. h. die Zeichenkette oder Liste muss exakt so viele Zeichen bzw. Einträge haben. Die Elemente `minLength`, `maxLength` und `length` sind, so wie die anderen Einschränkungs-Elemente auch, einteilig, wobei der Wert (in diesem Fall die Länge) für die Einschränkung mit dem Attribut `value` festgelegt wird.

```

1 <xs:element name="vorname">
2   <xs:simpleType>
3     <xs:restriction base="xs:string">
4       <xs:minLength value="2" />
5       <xs:maxLength value="50" />
6     </xs:restriction>
7   </xs:simpleType>
8 </xs:element>

```

### Wertebereich

Für Zahlen können Sie mit den Elementen `minInclusive` (Wert muss größer oder gleich dem definierten Wert sein), `minExclusive` (Wert muss größer als der definierte Wert sein), `maxInclusive` (Wert muss kleiner oder gleich dem definierten Wert sein) und `maxExclusive` (Wert muss kleiner als der definierte Wert sein) einen **zulässigen Wertebereich** festlegen. Natürlich ist es auch möglich, **nur eine Unter- oder Obergrenze** festzulegen.

```

1 <xs:element name="alter">
2   <xs:simpleType>
3     <xs:restriction base="xs:integer">
4       <xs:minInclusive value="18" />
5       <xs:maxInclusive value="65" />
6     </xs:restriction>
7   </xs:simpleType>
8 </xs:element>

```

### Dezimalstellen

Das Element `fractionDigits` kann dazu verwendet werden, die **maximale Anzahl der Nachkommastellen** festzulegen (nur für das Element `decimal`). Mit dem Element `totalDigits` ist es hingegen möglich, die **maximale Anzahl an Dezimalstellen** (sowohl vor als auch nach dem Punkt) zu definieren.

```

1 <xs:element name="temperatur">
2   <xs:simpleType>
3     <xs:restriction base="xs:decimal">
4       <xs:fractionDigits value="2" />
5     </xs:restriction>
6   </xs:simpleType>
7 </xs:element>

```

### Listenwerte

Eine weitere beliebte Form für die Nutzung von Einschränkungen ist die **Definition einer Auswahl an erlaubten Werten**. Hierfür dient das Element `enumeration`. Von diesem ordnen Sie dem `restriction`-Element beliebig viele unter. Alle Elemente (bzw. deren Werte) zusammen, stellen die Liste der Auswahlwerte dar. Den Wert des Auswahlwerts legen Sie, wie bereits bekannt, mit dem `value`-Attribut fest.

```

1 <xs:element name="farbe">
2   <xs:simpleType>
3     <xs:restriction base="xs:string">
4       <xs:enumeration value="rot" />
5       <xs:enumeration value="blau" />
6       <xs:enumeration value="grün" />
7     </xs:restriction>
8   </xs:simpleType>
9 </xs:element>

```

### Reguläre Ausdrücke

Eine weitere sehr interessante Möglichkeit zur Einschränkung eines Werts ist das Element `pattern`, mit welchem Sie **Werte mit Hilfe eines regulären Ausdrucks einschränken**. Der reguläre Ausdruck wird dabei im `value`-Attribut angegeben.

#### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

#### Community

- Blog
- Forum
- News

#### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSD](#) » [Einschränkungen](#)

```
1 <xs:element name="hex">
2   <xs:simpleType>
3     <xs:restriction base="xs:string">
4       <xs:pattern value="[0-9A-Fa-f]+" />
5     </xs:restriction>
6   </xs:simpleType>
7 </xs:element>
```

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [XSD](#) » [Ableitung](#)

## Ableitung

Den Begriff Ableitung kennen Sie vielleicht von der Objektorientierung bzw., um es genauer zu sagen, von der Vererbung. Die Bedeutung in XSD hat zwar Parallelen, darf jedoch keineswegs gleichgesetzt werden. Unter Ableitung (im Sinne von XML Schema) versteht man im Allgemeinen das **Erweitern oder Einschränken eines bestehenden Datentyps**. In den nächsten zwei Abschnitten werden wir diese beiden Arten der Ableitung genauer erläutern.

### Inhalt dieser Seite:

1. Erweiterung
2. Einschränkung

## Erweiterung

Eine Erweiterung eines Datentyps wird mit dem `extension`-Element durchgeführt. Dieses wird dem `simpleContent`- oder `complexContent`-Element untergeordnet. Innerhalb des `extension`-Elements können Sie dann angeben, **welche Inhalte erweitert werden sollen** (z. B. eine Sequenz oder ein Attribut). Über das Attribut `base` wird im `extension`-Element festgelegt, von welchem Datentyp der neu definierte Datentyp abgeleitet werden soll. Im folgenden Beispiel definieren wir die Datentypen `computer` und `laptop`. Der `laptop`-Datentyp wird vom `computer`-Datentyp abgeleitet und um ein Element erweitert:

```

1 <xs:complexType name="computer">
2   <xs:sequence>
3     <xs:element name="prozessor" type="xs:string" />
4     <xs:element name="arbeitspeicher" type="xs:string" />
5     <xs:element name="festplatte" type="xs:string" />
6   </xs:sequence>
7 </xs:complexType>
8 <xs:complexType name="laptop">
9   <xs:complexContent>
10    <xs:extension base="computer">
11      <xs:sequence>
12        <xs:element name="display" type="xs:string" />
13      </xs:sequence>
14    </xs:extension>
15  </xs:complexContent>
16 </xs:complexType>
    
```

Übrigens: Das `extension`-Element haben wir im Thema [Komplexe Typen](#) bereits dazu verwendet, einen komplexen Datentyp zu definieren, welcher aus einem direkten Inhalt (einfacher Typ) und einem Attribut besteht.

## Einschränkung

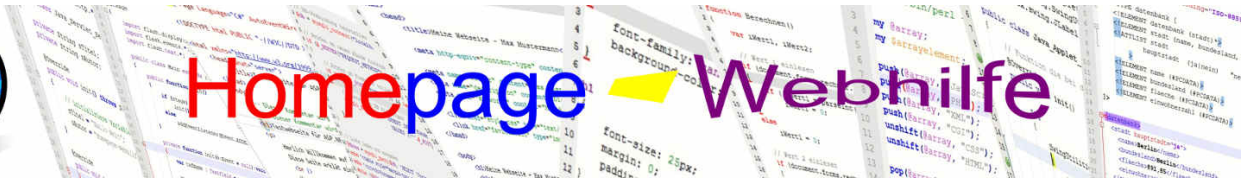
Das Element `restriction` kann dazu verwendet werden, Einschränkungen zu definieren. Sie haben das Element ja bereits im [vorherigen Thema](#) kennengelernt. Dort haben wir das Element aber nur dazu verwendet, Einschränkungen auf einfache Typen (z. B. eine Längenangabe oder einen Wertebereich) anzuwenden. Es ist aber möglich, bestehende komplexe Datentypen einzuschränken. Der Basistyp, also der Datentyp von welchem abgeleitet werden soll, wird im `restriction`-Element über das Attribut `base` festgelegt. Im abzuleitenden Datentyp müssen Sie **alle Elemente und Attribute des ursprünglichen Datentyps wiederholen**. Der Wert von Attributen wie z. B. `minOccurs` und `maxOccurs` kann natürlich durch eine Einschränkung verändert werden.

```

1 <xs:complexType name="person">
2   <xs:sequence>
3     <xs:element name="vorname" type="xs:string" minOccurs="0" />
4     <xs:element name="name" type="xs:string" />
5     <xs:element name="strasse" type="xs:string" />
6     <xs:element name="hausnr" type="xs:string" />
7     <xs:element name="plz" type="xs:string" />
8     <xs:element name="ort" type="xs:string" />
9   </xs:sequence>
10  <xs:attribute name="alter" type="xs:integer" />
11 </xs:complexType>
12 <xs:complexType name="firma">
13   <xs:complexContent>
14     <xs:restriction base="person">
15       <xs:sequence>
16         <xs:element name="name" type="xs:string" />
17         <xs:element name="strasse" type="xs:string" />
18         <xs:element name="hausnr" type="xs:string" />
19         <xs:element name="plz" type="xs:string" />
20         <xs:element name="ort" type="xs:string" />
21       </xs:sequence>
22       <xs:attribute name="alter" type="xs:integer" use="prohibited" />
23     </xs:restriction>
24   </xs:complexContent>
25 </xs:complexType>
    
```

**Wichtig:** Bei der Einschränkung kann ein Datentyp nicht so eingeschränkt werden, dass er mit dem ursprünglichen Datentyp komplett inkompatibel ist, d. h. das Weglassen des Elements `vorname` im XSL Schema des Datentyps `firma` im obigen Beispiel ist nur deshalb erlaubt, weil es im Datentyp `person` optional ist (`minOccurs="0"`). Attribute können hingegen über den Wert `prohibited` im `use`-Attribut verboten werden. Auf Grund der genannten „Problematik“ bei Elementen bzw. der Umständlichkeit bei der Einschränkung von Datentypen wird meistens die Erweiterung von Datentypen bei der Ableitung vorgezogen.

<b>Über uns</b> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<b>Community</b> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<b>Nachschlagewerk</b> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	Benjamin Jung Krummstraße 9/3 73054 Eislingen
Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a>			



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [RSS](#) » [Grundlagen](#)

## Grundlagen



Mit RSS-Dokumenten (*Really Simple Syndication*) ist es möglich, Web-Feeds (auch News-Feeds genannt) bereitzustellen. RSS ist eine XML basierende Sprache mit einem sehr geringen und übersichtlichen Sprachumfang.

RSS wurde, im Gegensatz zu vielen anderen XML-Sprachen, nicht vom World Wide Web Consortium (W3C) entworfen. Die Entwicklung von RSS begann bereits 1999. Die erste Version (RSS 0.90) wurde von Netscape entwickelt. Die aktuelle Version (und vermutlich letzte Version), RSS 2.0 vom Jahre 2002, wurde vom Unternehmen UserLand Software (bekannt durch Dave Winer) entwickelt und kann heute als verbreitetste Version angesehen werden. Dieses Tutorial bezieht sich daher auf RSS 2.0. Eine Alternative zu RSS ist Atom (ASF). Einen kurzen Überblick zu den Unterschieden bekommen Sie im [letzten Thema dieses Kapitels](#).

RSS-Dateien haben i. d. R. die Dateierdung `.rss`. Als MIME-Typ wird `application/rss+xml` angegeben. Web-Feeds können **auf Websites zur Verfügung gestellt** werden und können dann **durch Besucher abonniert werden**, wodurch diese benachrichtigt werden, sobald es eine Aktualisierung im Feed gibt.

### Inhalt dieser Seite:

1. Funktionsweise
2. Grundaufbau
3. Einbindung

## Funktionsweise

Ein RSS-Feed (auch als RSS-Channel bezeichnet) ist nichts anderes als ein XML-Dokument, welches aus zwei wesentlichen Teilen besteht: den Kanal-Informationen und den Einträgen. In den **Kanal-Informationen** sind grundlegende Informationen über das Web-Feed hinterlegt. Dazu zählen z. B. ein Name und eine Beschreibung, aber auch ein Link zur Website, die Sprache des Feeds und das Veröffentlichungsdatum. Neben den Kanal-Informationen kann das Feed beliebig viele Einträge besitzen. Ein solcher **Eintrag** (z. B. ein Newseintrag, eine Nachrichtenmeldung oder ein Blog-Beitrag) enthält ebenfalls wieder einen Titel, eine Beschreibung, einen Link (i. d. R. zum vollständigen Artikel) und ggf. weitere Informationen (z. B. die Angabe des Autors oder der Zeitpunkt der Veröffentlichung).

Um RSS-Feeds zu lesen (ohne dabei die XML-Datei betrachten zu müssen), benötigen Sie ein Programm: den sogenannten **Feedreader**. Feedreader sind als eigenständige Programme erhältlich, sind aber auch in einigen E-Mail-Clients und Webbrowsern enthalten. Möchten Sie **über die Änderungen an einem RSS-Feed informiert** werden, so müssen Sie das Feed abonnieren. Die Verwaltung von Abonnements sowie die automatische Aktualisierung von News-Feeds übernehmen die Feedreader. Anders als bei einer E-Mail muss der Feedreader selbst aktiv werden, um auf eine Aktualisierung zu prüfen. Dabei lädt sich der Feedreader das RSS-Feed in regelmäßigen Abständen vom Server herunter und überprüft die Datei auf Änderungen. Dies hat zur Folge, dass **Meldungen des Feeds evtl. nicht sofort angezeigt werden**, sondern erst mit einer bestimmten Verzögerung (je nach eingestellter Aktualisierungsrate im Feedreader).

Eingesetzt werden RSS-Feeds hauptsächlich bei Nachrichtenseiten oder Blogs. Aber auch Firmen können RSS-Feeds als „moderne Alternative“ zum klassischen Newsletter anbieten.

## Grundaufbau

Ein RSS-Dokument hat einen ziemlich einfachen Grundaufbau: Es gibt das Wurzelement `rss`, welchem exakt ein `channel`-Element untergeordnet wird, d. h. jedes RSS-Dokument muss exakt einen RSS-Kanal enthalten. Möchten Sie mehrere RSS-Kanäle anbieten, so müssen Sie für jeden Kanal ein eigenes Dokument erstellen. Im `rss`-Element wird des Weiteren über das Attribut `version` die RSS-Version des Dokuments festgelegt.

```

1 | <?xml version="1.0" encoding="UTF-8" ?>
2 |
3 | <rss version="2.0">
4 |   <channel>
5 |
6 |   </channel>
7 | </rss>
    
```

## Einbindung

RSS-Feeds können **in ganz normale HTML-Seiten eingebunden** werden. Dazu verwenden Sie das `link`-Element (im `head`-Element des HTML-Dokuments) mit dem Wert `alternate` im `rel`-Attribut. Zudem werden die Attribute `href` (Link zum Dokument) und `type` (MIME-Typ des Dokuments) benötigt. Über das optionale Attribut `title` kann dem RSS-Feed noch ein Titel gegeben werden.

```

1 | <link rel="alternate" href="feed.rss" type="application/rss+xml" title="Titel für das Feed" />
    
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisligen</p> <p><b>Web:</b> <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> <b>E-Mail:</b> <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [RSS](#) » [Kanal-Informationen](#)

## Kanal-Informationen

Wie bereits im vorherigen Thema angesprochen, besteht ein Teil des RSS-Feeds aus **allgemeinen Informationen über das Feed**. Dafür stehen uns einige Elemente zur Verfügung, welche dem `channel`-Element direkt untergeordnet werden. Die folgende Tabelle zeigt eine Auflistung der wichtigsten verfügbaren Elemente und deren Bedeutung:

<b>title</b>	Titel des Kanals (erforderlich)
<b>description</b>	(Kurz-)Beschreibung des Kanals (erforderlich)
<b>link</b>	Link zur Website, zu welcher der Kanal gehört (erforderlich)
<b>language</b>	Sprache des Kanals (Angabe als Sprachcodes nach Netscape oder HTML)
<b>category</b>	Angabe von Kategorie(n), getrennt durch Leerzeichen
<b>copyright</b>	Angabe des Copyrights
<b>managingEditor</b>	E-Mail-Adresse und Name (in Klammern) der Person, die für den redaktionellen Inhalt verantwortlich ist
<b>webMaster</b>	E-Mail-Adresse und Name (in Klammern) der Person, die für technische Probleme verantwortlich ist
<b>pubDate</b>	Veröffentlichungsdatum des Kanals (Angabe nach RFC 822)
<b>lastBuildDate</b>	Datum der letzten Änderung des Kanals (Angabe nach RFC 822)
<b>ttl</b>	Anzahl der Minuten, in welcher der Kanal aktualisiert wird

Das folgende Beispiel enthält einige der obigen Elemente (jedoch noch keine Einträge). Beim Klicken auf das Vorschau-Icon wird Ihnen eine HTML-Seite angezeigt, in welcher das RSS-Feed eingebunden wurde. In einigen Browsern müsste Ihnen in der Titelleiste oder im Menü das RSS-Feed angezeigt werden.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <rss version="2.0">
4   <channel>
5     <title>Newsticker der Firma Example GmbH</title>
6     <description>Hier bekommen Sie alle aktuellen Informationen über unsere Firma und Produkte.</description>
7     <link>http://www.example.org/News/</link>
8     <language>de-DE</language>
9     <copyright>Copyright 2017 by Example GmbH</copyright>
10    <pubDate>Sat, 18 Mar 2017 15:03:42 +0100</pubDate>
11   </channel>
12 </rss>
    
```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » XML & Co. » RSS » Einträge

## Einträge

Neben den Kanal-Informationen enthält das RSS-Dokument **beliebig viele Einträge** (auch Items genannt). Auch ein RSS-Feed ohne Einträge wäre von Syntax gültig. Einträge kennzeichnen sich durch das Element `item`, welches dem `channel`-Element untergeordnet wird. Im `item`-Element können dann verschiedene Elemente untergeordnet werden, um den Eintrag genauer zu spezifizieren. Die folgende Tabelle zeigt die wichtigsten Elemente:

<b>title</b>	Titel des Eintrags (erforderlich)
<b>description</b>	(Kurz-)Beschreibung des Eintrags (erforderlich)
<b>link</b>	Link zum (vollständigen) Artikel, zu welchem der Eintrag gehört (erforderlich)
<b>category</b>	Angabe von Kategorie(n), getrennt durch Leerzeichen
<b>author</b>	E-Mail-Adresse und Name (in Klammern) der Person, die den Eintrag verfasst hat
<b>pubDate</b>	Veröffentlichungsdatum des Eintrags (Angabe nach RFC 822)
<b>guid</b>	GUID, die den Eintrag eindeutig kennzeichnet

Hierzu ebenfalls wieder ein Beispielcode:

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <rss version="2.0">
4    <channel>
5      <title>Newsticker der Firma Example GmbH</title>
6      <description>Hier bekommen Sie alle aktuellen Informationen über unsere Firma und Produkte.</description>
7      <link>http://www.example.org/News/</link>
8      <language>de-DE</language>
9      <copyright>Copyright 2017 by Example GmbH</copyright>
10     <pubDate>Sat, 18 Mar 2017 15:03:42 +0100</pubDate>
11
12     <item>
13       <title>Wir ziehen um</title>
14       <description>Wir verlegen unseren Hauptsitz von Musterstadt nach Musterdorf.</description>
15       <link>http://www.example.org/News/Umzug/</link>
16       <author>mustermann@example.com (M. Mustermann)</author>
17       <pubDate>Wed, 22 Feb 2017 11:32:40 +0100</pubDate>
18     </item>
19     <item>
20       <title>IT-Messe in Musterberg</title>
21       <description>Vom 27. bis 31. März findet die IT-Messe in Musterberg statt: Sichern Sie sich hier Ihre kostenlose
22       Eintrittskarte.</description>
23       <link>http://www.example.org/News/Messe/</link>
24       <author>musterfrau@example.com (S. Musterfrau)</author>
25       <pubDate>Sat, 18 Mar 2017 15:03:42 +0100</pubDate>
26     </item>
27   </channel>
28 </rss>
    
```

**Wichtig:** Oftmals tritt die Frage auf, ob es möglich ist, im `description`-Element HTML-Inhalt zu notieren. Die Antwort: Ja, aber der HTML-Inhalt muss maskiert werden (z. B. `&lt;` statt `<`). Eine weitere Möglichkeit wäre die Platzierung des HTML-Inhalts in einem CDATA-Abschnitt.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [RSS](#) » [Vergleich mit Atom](#)

## Vergleich mit Atom

### Inhalt dieser Seite:

1. Elemente
2. Beispiel

Atom bzw. um es genauer zu sagen, das *Atom Syndication Format* (kurz ASF) ist ein **Format, um Web-Feeds zu formulieren**. ASF soll in Zukunft das Format RSS ablösen. Wie auch RSS, ist ASF eine XML-basierte Sprache. Im Gegensatz zu RSS ist ASF jedoch ein eingetragener IETF-Standard. ASF-Dokumente haben i. d. R. die Datei-Endung `.atom`. Als MIME-Typ kommt `application/atom+xml` zum Einsatz. ASF-Dateien können, so wie RSS-Dateien auch, von Feedreadern (dazu zählen auch einige E-Mail-Clients und Webbrowser) verarbeitet werden.

Atom kann im Gegensatz zu RSS als **sauberes Format** bezeichnet werden, da es keine wirklichen Designfehler enthält und vor allem als Standard offiziell anerkannt ist. RSS war und ist eher eine Art „Bastelformat“. Als Vorteil von Atom kann auch die Verwendung von Namensräumen genannt werden. Bei ASF soll laut den Autoren zudem der Inhalt der Feed-Einträge wieder im Vordergrund stehen.

Trotz den Vorteilen des Atom-Formats werden auch auf aktuellen Websites und Blogs immer noch RSS-Feeds verwendet. Dies ist auf die **ursprüngliche Popularität** zurückzuführen. Auch das Angebot von RSS- und Atom-Feeds ist auf einigen Websites zu finden.

## Elemente

Atom ist zwar nicht mit RSS kompatibel, es ist jedoch durchaus möglich, ein **RSS-Dokument in ein ASF-Dokument zu transformieren**. In den meisten Fällen müssen Sie lediglich die Elementnamen ersetzen. Die folgende Tabelle zeigt, wie sich die Elementnamen in RSS und ASF unterscheiden:

	RSS	Atom	Hinweis zu Atom
	rss	-	-
	channel	feed	ist das Wurzelement
K a n a l - I n f o r m a t i o n e n	title	title	erforderlich
	description	subtitle	-
	link	link	einteiliges Element mit href-Attribut
	language	-	-
	category	category	jede Kategorie in einem eigenen Element
	copyright	rights	-
	managingEditor	author oder contributor	Element enthält name- und ggf. email- und uri-Element
	webMaster	-	-
	pubDate	published	Angabe nach RFC 3339
	lastBuildDate	updated	Angabe nach RFC 3339, erforderlich
ttl	-	-	
-	id	eindeutige ID (GUID), erforderlich	
	item	entry	-
E i n t r ä g e	title	title	erforderlich
	description	summary	-
	-	content	(vollständiger) Inhalt des Eintrags
	link	link	einteiliges Element mit href-Attribut
	category	category	jede Kategorie in einem eigenen Element
	-	rights	Angabe der Rechte
	author	author oder contributor	Element enthält name- und ggf. email- und uri-Element
	pubDate	published	Angabe nach RFC 3339
	-	updated	Angabe nach RFC 3339, erforderlich
guid	id	erforderlich	

## Beispiel

Die **Einbindung eines ASF-Dokuments in eine HTML-Seite** unterscheidet sich, abgesehen von MIME-Typ (und natürlich dem Dateinamen und / oder der Dateiendung), nicht von der Einbindung einer RSS-Datei:

```
1 | <link rel="alternate" href="feed.atom" type="application/atom+xml" title="Titel für das Feed" />
```

Das folgende Beispiel zeigt den **Code eines Atom-Feeds** mit einigen Kanal-Informationen sowie zwei Einträgen:

```
1 | <?xml version="1.0" encoding="UTF-8" ?>
2 |
3 | <feed xmlns="http://www.w3.org/2005/Atom">
4 |   <title>Newsticker der Firma Example GmbH</title>
```

<b>Über uns</b> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<b>Community</b> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<b>Nachschlagewerk</b> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	Benjamin Jung Krummstraße 9/3 73054 Eisingen
			Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a>





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [RSS](#) » [Vergleich mit Atom](#)

```

5     <subitle>Hier bekommen Sie alle aktuellen Informationen über unsere Firma und Produkte.</subitle>
6     <link href="http://www.example.org/News/" />
7     <rights>Copyright 2017 by Example GmbH</rights>
8     <id>urn:uuid:b0ecf184-31c5-4485-b96d-17fae704d277</id>
9     <updated>2017-03-18T15:03:42+01:00</updated>
10
11     <entry>
12         <title>Wir ziehen um</title>
13         <summary>Wir verlegen unseren Hauptsitz von Musterstadt nach Musterdorf.</summary>
14         <link href="http://www.example.org/News/Umzug/" />
15         <author>
16             <name>M. Mustermann</name>
17             <email>mustermann@example.com</email>
18         </author>
19         <id>urn:uuid:9656699e-48de-4935-ab74-05916fef78c4</id>
20         <updated>2017-02-22T11:32:40+01:00</updated>
21     </entry>
22     <entry>
23         <title>IT-Messe in Musterberg</title>
24         <summary>Vom 27. bis 31. März findet die IT-Messe in Musterberg statt: Sichern Sie sich hier Ihre kostenlose
25         Eintrittskarte.</summary>
26         <link href="http://www.example.org/News/Messe/" />
27         <author>
28             <name>S. Musterfrau</name>
29             <email>musterfrau@example.com</email>
30         </author>
31         <id>urn:uuid:d33709e8-3c57-49c4-a49c-c6ee31e88b81</id>
32         <updated>2017-03-18T15:03:42+01:00</updated>
33     </entry>
</feed>

```



## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

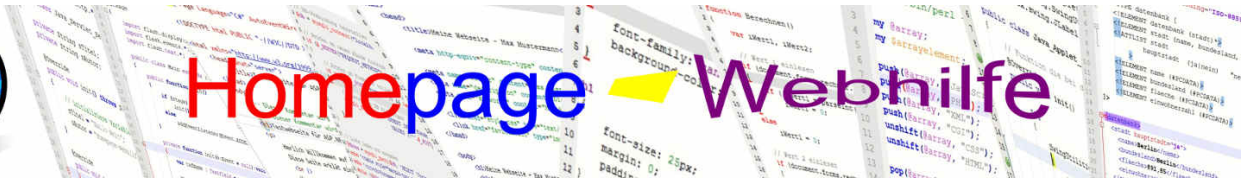
## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [SVG](#) » [Grundlagen](#)

## Grundlagen



SVG (*Scalable Vector Graphics*, zu Deutsch *skalierbare Vektorgrafik*) ist eine XML basierende Sprache, welche es erlaubt, **zweidimensionale Vektorgrafiken zu beschreiben**. Die Sprache wurde vom W3C entworfen und wird von den meisten Browsern nahezu vollständig unterstützt.

In SVG gibt es verschiedene Elemente, die Sie im XML-Dokument platzieren können, um dadurch **Linien, Rechtecke, Kreise, Ellipsen, Polylinien, Polygone, aber auch Texte und komplexe Pfade zu zeichnen**. SVG entstand durch den Wunsch auf die Standardisierung der Sprachen VML und PGML. Das W3C standardisierte keine dieser Sprachen, sondern schuf eine eigene: SVG. Die erste Version erschien bereits 2001. Die aktuelle Version, SVG 1.1, wurde 2003 veröffentlicht und **wird heute von fast allen Browsern unterstützt**. Seit 2011 wird an SVG 2 gearbeitet. SVG 2 bringt viele Erweiterungen, wird jedoch von nur wenigen Browsern (vollständig) unterstützt.

Eine SVG-Datei hat üblicherweise die Dateierweiterung `.svg`. Als MIME-Typ kommt `image/svg+xml` zum Einsatz. Die Einbindung eines SVG-Dokuments in eine HTML-Seite kann auf verschiedene Arten erfolgen (z. B. als Bild oder per Direktnotation). Diese werden

auf dieser Seite aber [noch näher beschrieben](#).

### Inhalt dieser Seite:

1. Verwendung
2. Grundaufbau
3. Einbindung
4. Positions- und Größenangaben
5. Attribute

## Verwendung

SVG-Grafiken können an ganz unterschiedlichen Stellen verwendet und eingesetzt werden. Trotzdem sind Vektorgrafiken noch lange nicht so verbreitet wie die üblichen Rastergrafiken. Vektorgrafiken haben im Gegensatz zu Rastergrafiken den Vorteil, dass diese **beliebig skaliert** werden können, **ohne einen Qualitätsverlust** zu erleiden. Diese Aufbereitung einer Vektorgrafik erfordert für den Rendering-Vorgang zwar einen höheren Ressourcenverbrauch, dieser ist jedoch bei den heutigen Computern kein Hindernis mehr.

SVG ist ein Vektorgrafikformat, welches vor allem **für die Verwendung im World Wide Web entwickelt und entworfen wurde** (z. B. für responsives Webdesign). Gerade deshalb wird der Großteil der SVG-Spezifikation (zu mindestens, wenn man von SVG 1.1 ausgeht) von so gut wie allen Webbrowsern unterstützt. Der Einsatz von SVG-Grafiken auf Webseiten und unter der Annahme, dass einigermaßen aktuelle Browserversionen verwendet werden, kann man also als problemlos bezeichnen.

SVG-Grafiken können auch **über Skripte dynamisiert werden**. Für den Zugriff auf die Dokument-Bestandteile aus dem Skript heraus wird das DOM (*Document Object Model*) verwendet, welches Ihnen vielleicht aus HTML oder JavaScript bekannt ist. Gerade durch diese Funktionalität ermöglicht sich eine völlig neue Welt in dynamische und **interaktive Grafiken**.

## Grundaufbau

Im Prinzip ist eine SVG-Datei nur eine ganz normale XML-Datei, bei welcher die Namen von Elementen, Attributen und Attributwerten über eine DTD festgelegt sind. Eine SVG-Datei enthält also i. d. R. zuallererst eine XML-Deklaration. Es folgt der Verweis zur DTD, welcher unter <http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd> zu finden ist. Als `PUBLIC`-Identifikator wird `--//W3C//DTD SVG 1.1//EN` angegeben.

Das Wurzelement von SVG ist `svg`, in welchem im Normalfall das `version`-Attribut (zumeist mit dem Wert `1.1`) notiert wird. Als Namensraum wird <http://www.w3.org/2000/svg> verwendet. Für einige Features, wie z. B. das Erstellen eines Hyperlinks oder das Referenzieren eines Skripts, benötigen Sie **XLink**. XLink wird über den Namensraum <http://www.w3.org/1999/xlink> eingebunden. Als Namensraumpräfix wird meistens `xlink` verwendet.

Ein weiteres wichtiges Attribut ist `viewBox`. Das Attribut ist optional, sollte jedoch immer angegeben werden. Als Werte werden vier numerische Werte erwartet. Die ersten zwei Werte geben die X- und Y-Position des **Anzeigebereichs** (engl. *viewport*) an und sind i. d. R. 0. Wird das `viewBox`-Attribut bei anderen Elementen (die innerhalb des `svg`-Elements platziert sind) verwendet, dann kann es auch vorkommen, dass hier positive oder negative Werte zum Einsatz kommen. Die nächsten zwei Zahlen im Wert des `svg`-Attributs geben die Breite und Höhe des Anzeigebereichs an. Der mit diesem Attribut definierte Anzeigebereich ist elementar für die restlichen Positions- und Größenangaben innerhalb des `svg`-Elements. Durch diese Angaben ist es dann letztendlich auch möglich, dass die Anzeigesoftware (z. B. der Browser) die Grafik nach Belieben skalieren kann. Diese Tatsache führt dazu, dass es sich bei der Grafik um eine Vektorgrafik handelt. Der **Standard-Anzeigebereich** wird über die Attribute `width` und `height` festgelegt. Diese Attribute können aber auch durchaus weggelassen werden.

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5 <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1" viewBox="0 0 200 200">
6
7 </svg>
    
```

## Einbindung

SVG-Grafiken können auf unterschiedliche Art und Weisen in eine HTML-Seite eingebunden werden. Der vermutlich einfachste Weg ist die Verwendung des `img`-Elements, welches auch dazu verwendet wird, „normale“ Rastergrafiken einzubinden. Dabei sollte jedoch beachtet werden, dass die Referenzierung auf Stylesheets sowie die Verwendung von Links und Skripten aus Sicherheitsgründen von den Browsern im Regelfall geblockt werden.

```
1 
```

Möchten Sie eine **interaktive SVG-Grafik** (z. B. mit Links oder Skripten) einbinden, dann empfiehlt sich die Verwendung des `embed`-Elements.

```
1 <embed type="image/svg+xml" src="Grafik.svg" width="200" height="200" />
```

Eine weitere Möglichkeit zur Einbindung einer SVG-Grafik wäre die Verwendung der `background-image`-Eigenschaft von CSS. Weitere Einstellungen können dann natürlich mittels den Eigenschaften `background-size`, `background-position` und `background-repeat` vorgenommen werden.

```
1 <div style="width: 200px; height: 200px; background-image: url('Grafik.svg'); background-repeat: no-repeat;"></div>
```

Mit HTML5 ist es nun auch möglich, SVG-Elemente direkt in HTML zu notieren. Dafür wird einfach das `svg`-Element inkl. dessen Unterelemente innerhalb eines HTML-Elements platziert. Die Größenangabe erfolgt durch das `width`- und `height`-Attribut. In einem solchen Fall ist dann z. B. auch möglich, das `viewBox`-

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

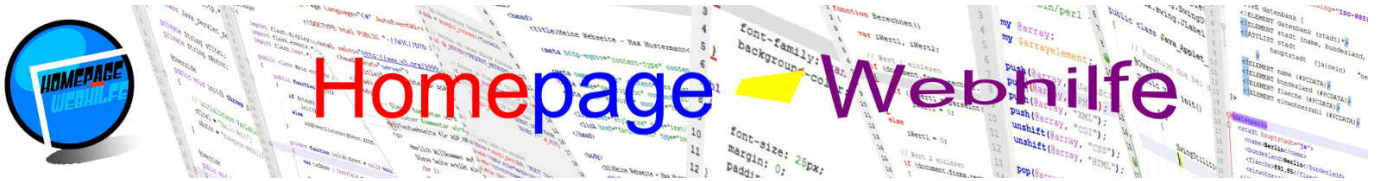
### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [SVG](#) » [Grundlagen](#)

Attribut wegzulassen, sofern der Wert des `width`- und `height`-Attributs einen „festen“ Wert hat und sich die Größen- und Positionsangabe, der im `svg`-Element platzierten Elemente, auf die zwei Attributwerte beziehen.

```

1 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="200" height="200">
2
3 </svg>
    
```

## Positions- und Größenangaben

In SVG beziehen sich die Positions- und Größenangaben immer auf das `width`- und `height`-Attribut oder auf die Größenangabe im `viewBox`-Attribut. Wird in diesen Attributen keine Maßeinheit explizit angegeben, so wird von Pixel ausgegangen. Natürlich ist es aber auch möglich, eine SVG-Grafik in einem anderen Koordinatensystem aufzubauen. Dafür wird hinter einem Wert einfach die gewünschte Maßeinheit notiert. Als **Maßeinheiten** stehen `em` (Schriftgröße), `ex` (Höhe des kleinen x), `px` (Pixel), `in` (Inch), `cm` (Zentimeter), `mm` (Millimeter), `pt` (Punkt) und `pc` (Pica) zur Verfügung. In den **Positionierungsattributen** von Elementen (z. B. für eine Linie oder ein Rechteck), welche innerhalb des `svg`-Elements platziert werden, sind keine Maßeinheiten anzugeben, da sich die Angaben immer auf das **Koordinatensystem der SVG-Grafik** (und somit im Regelfall auf die Werte des `viewBox`-Attributs) beziehen. Bei **Attributen für Größenangaben** ist es hingegen wieder möglich, eine Maßeinheit mit anzugeben. Wird keine Maßeinheit angegeben, so wird von der Maßeinheit des Koordinatensystems ausgegangen.

## Attribute

Bevor wir uns in den nächsten Themen auf die SVG-Elemente stürzen, um etwas in unsere SVG-Grafik zeichnen zu können, möchten wir Ihnen hier ein paar wichtige Attribute nennen, welche Sie immer wieder benötigen werden.

Zu den **Attributen zur Positionierung** gehören `x`, `x1`, `x2`, `cx` (*center-x*), `y`, `y1`, `y2`, `cy` (*center-y*). Für **Größenangaben** werden die Attribute `width` und `height` sowie `r` (*radius*), `rx` (*radius-x*) und `ry` (*radius-y*) verwendet. Diese Attribute werden auch als „primitive“ bzw. grundlegende Attribute bezeichnet.

Neben diesen Attributen gibt es noch eine weitere wichtige Gruppe an Attributen: die **Präsentationsattribute**. Die folgende Tabelle zeigt ein paar wichtige Attribute dieser Gruppe:

<b>fill</b>	Füllfarbe
<b>fill-opacity</b>	Transparenz der Füllfarbe
<b>stroke</b>	Farbe des Umrisses (Rahmen)
<b>stroke-opacity</b>	Transparenz der Farbe des Umrisses (Rahmen)
<b>stroke-width</b>	Dicke des Umrisses (Rahmen)
<b>transform</b>	Angabe von Transformations-Funktionen ( <code>translate(x y)</code> , <code>scale(x y)</code> , <code>rotate(a x y)</code> , <code>skewX(a)</code> , <code>skewY(a)</code> )

**Übrigens:** Für die Farbangaben gibt es ähnliche Möglichkeiten wie in CSS. Es ist möglich, einen (englischen) Farbnamen, einen Hex-Wert (Kurz- oder Langform) oder die RGB-Funktion zu notieren.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p><b>Web:</b> <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> <b>E-Mail:</b> <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [SVG](#) » [Formen](#)

## Formen

In SVG stehen uns einige **Grundformen** zur Verfügung, welche direkt als **einteilige Elemente** notiert werden können.

**Inhalt dieser Seite:**

1. Linie
2. Rechteck
3. Kreis
4. Ellipse
5. Polylinie
6. Polygon

### Linie

Die einfachste Form ist eine Linie. Eine Linie (Element `line`) besteht aus **zwei Koordinatenpunkten**. Der erste Punkt wird über die Attribute `x1` und `y1` festgelegt. Der zweite Positionierungspunkt wird mit den Attributen `x2` und `y2` spezifiziert. Die **Linienfarbe und -breite** kann über die Attribute `stroke` und `stroke-width` festgelegt werden. Der **Stil der Linienenden** kann über das Attribut `stroke-linecap` festgelegt werden. Als Werte stehen `round` (Enden haben eine abgerundete Kappe), `square` (Enden haben eine eckige Kappe) und `butt` (Enden sind flach, Standardwert) zur Verfügung.

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" viewBox="0 0 600 400">
6   <line x1="50" y1="50" x2="400" y2="300" stroke="red" />
7   <line x1="0" y1="200" x2="600" y2="200" stroke="lime" stroke-width="3" />
8   <line x1="300" y1="0" x2="300" y2="400" stroke="lime" stroke-width="3" />
9   <line x1="100" y1="350" x2="250" y2="100" stroke="blue" stroke-width="10" stroke-linecap="round" />
10 </svg>
    
```



### Rechteck

Eine weitere primäre Form ist das Rechteck. Das Rechteck, welches durch das `rect`-Element angegeben wird, wird an Hand der linken oberen Ecke über die Attribute `x` und `y` platziert. Die Größenangabe erfolgt über die Attribute `width` und `height`. Möchten Sie ein **Quadrat** zeichnen, so verwenden Sie ebenfalls das `rect`-Element. Als Werte für die Attribute `width` und `height` müssen Sie dann einfach den gleichen Wert angeben. Um das Rechteck zu stylen, stehen Ihnen natürlich wieder einige Präsentationsattribute zur Verfügung (z. B. `fill`, `stroke` und `stroke-width`).

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" viewBox="0 0 600 400">
6   <rect x="50" y="50" width="200" height="100" fill="lime" />
7   <rect x="300" y="100" width="250" height="250" fill="red" stroke="blue" stroke-width="10" />
8 </svg>
    
```



**Übrigens:** Möchten Sie die Ecken eines Rechtecks abrunden, so können Sie die Attribute `rx` und `ry` verwenden. Für eine „gleichmäßige“ Abrundung muss der Wert des `rx`-Attributs mit dem Wert des `ry`-Attributs übereinstimmen.

### Kreis

Der Kreis ist eine weitere Grundform von SVG und wird über das Element `circle` angegeben. Die Platzierung eines Kreises erfolgt durch die Attribute `cx` und `cy`. Die Werte geben dabei die Position der Kreismitte an. Der Radius eines Kreises wird über das Attribut `r` festgelegt.

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" viewBox="0 0 600 400">
6   <circle cx="100" cy="100" r="50" fill="red" />
7   <circle cx="300" cy="200" r="100" fill="blue" />
8 </svg>
    
```



### Ellipse

Die Ellipse ist ebenfalls eine Grundform, hat jedoch im Gegensatz zu einem Kreis einen **separaten Radius für die X- und Y-Achse**. Um eine Ellipse zu notieren, benötigen Sie das Element `ellipse`. Die Positionierung erfolgt durch die Attribute `cx` und `cy` (Ellipsenmitte). Der Radius wird über die Attribute `rx` und `ry` festgelegt. Haben die Attribute `rx` und `ry` den gleichen Wert, so handelt es sich bei der Ellipse um einen Kreis und Sie könnten dann auch einfach das `circle`-Element verwenden.

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" viewBox="0 0 600 400">
6   <ellipse cx="100" cy="100" rx="50" ry="75" fill="red" />
7   <ellipse cx="350" cy="200" rx="150" ry="100" fill="blue" />
8 </svg>
    
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [SVG](#) » [Formen](#)



## Polylinie

Die Polylinie (auch Polygonzug genannt, Element `polyline`) zählt auch zu den SVG-Grundformen und besteht eigentlich nur aus **mehreren Punkten, welche über eine Linie verbunden werden**. Um die Punkte festzulegen, wird das Attribut `points` verwendet. Im `points`-Attribut geben Sie nun mehrere Koordinatenpunkte (in der Form `x,y`) getrennt durch ein Leerzeichen an. Besteht Ihre Polylinie also z. B. aus 3 Punkten, so lautet die Angabe `x1,y1 x2,y2 x3,y3`. Auch hier können selbstverständlich wieder die Attribute `stroke`, `stroke-width` etc. verwendet werden.

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" viewBox="0 0 600 400">
6   <polyline points="150,50 450,350 450,50 150,50 150,350 450,50" fill="none" stroke="blue" stroke-width="5" />
7 </svg>

```



## Polygon

Das Polygon ist sehr ähnlich zu der Polylinie, denn auch hier besteht die Form, welche über das `polygon`-Element notiert wird, aus mehreren Punkten, die über das Attribut `points` festgelegt werden. Anders als bei der Polylinie wird beim Polygon **der letzte Punkt automatisch mit dem ersten Punkt verbunden**, d. h. das Polygon wird dazu verwendet, „geschlossene“ Formen zu zeichnen, wohingegen die Polylinie zur Zeichnung von „offenen“ Formen verwendet wird.

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" viewBox="0 0 600 400">
6   <polygon points="300,50 450,150 450,250 300,350 150,250 150,150" fill="blue" />
7 </svg>

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [SVG](#) » [Texte](#)

## Texte

In SVG-Dokumenten können Sie selbstverständlich auch einen Text platzieren. Hierfür benötigen Sie das SVG-Element `text`. Die **Positionierung** erfolgt wie gewohnt mit den Attributen `x` und `y`. Die **Farbe des Textes** wird über das Attribut `fill` festgelegt. Soll der Text zusätzlich über einen **Umriss** verfügen, so können Sie die Attribute `stroke` und `stroke-width` nutzen. Die **Schrifteinstellungen** können über einige weitere Attribute festgelegt werden. Die Namen und deren Bedeutung sind mit den CSS-Eigenschaften vergleichbar: `font-family` (Name der Schriftart), `font-size` (Schriftgröße), `font-style` (Stil der Schrift, `italic` für Kursivschrift), `font-weight` (Schriftstärke, `bold` für Fettdruck) und `text-decoration` (Dekoration der Schrift, z. B. `underline` zum Unterstreichen). Ein weiteres nützliches Element ist `tspan`. `tspan` kann innerhalb des `text`-Elements notiert werden und wird dazu verwendet, einzelne **Teile eines Textes hervorzuheben** (z. B. durch eine Farbe, einen Umriss oder eine andere Schriftgröße).

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5  <svg xmlns="http://www.w3.org/2000/svg" version="1.1" viewBox="0 0 600 400">
6    <text x="30" y="75" fill="lime" font-size="60px">Herzlich Willkommen</text>
7    <text x="275" y="150" fill="lime" font-size="40px">auf</text>
8    <text x="75" y="250" font-size="50px">
9      <tspan fill="red">Home</tspan><tspan fill="blue">page</tspan>
10     <tspan fill="yellow">-</tspan>
11     <tspan fill="purple">Webhilfe</tspan>
12   </text>
13 </svg>

```

**Wichtig:** Das `x`- und `y`-Attribut legt standardmäßig die Position des ersten Buchstabens (X-Achse) auf der Grundlinie (Y-Achse) fest, d. h. der Text ist linksbündig. Um dies zu ändern, können Sie das Attribut `text-anchor` verwenden. Mögliche Werte für dieses Attribut sind `start` (linksbündig, Standardwert), `middle` (zentriert) und `end` (rechtsbündig). Verwenden Sie z. B. den Wert `middle`, so gibt der Wert des `x`-Attributs die mittlere Position des Textes an.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung

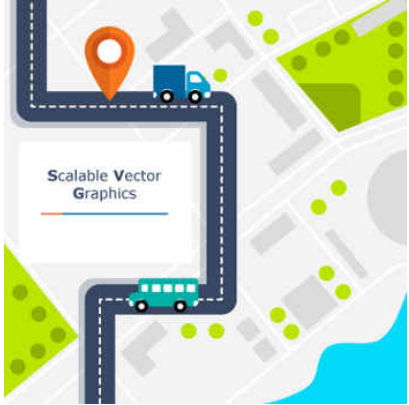


Sie befinden sich hier: Homepage-Webhilfe » XML & Co. » SVG » Pfade

## Pfade

Als Pfad (engl. *path*) bezeichnet man den **Weg von Linien und Kurven**. Pfade erlauben es daher, also sozusagen formfrei, etwas zu zeichnen. Als Grundlage zum Zeichnen dienen Linien und Kurven. Einige werden den Begriff Pfad vielleicht aus Bildbearbeitungsprogrammen kennen. Ein Pfad wird über das Element `path` definiert. Um Pfade zu zeichnen, gibt es verschiedene **Kommandos**, diese werden als Kette inkl. der jeweiligen Kommando-Werte im Attribut `d` angegeben. Die folgende Tabelle zeigt eine Auflistung der Pfad-Kommandos:

<b>M x y</b>	Setzt den „Cursor“ an die angegebene Position.
<b>L x y</b>	Zeichnet an die Linie zu der angegebenen Position.
<b>H x</b>	Zeichnet eine horizontale Linie zu der angegebenen X-Position (Wert der Y-Achse bleibt gleich).
<b>V y</b>	Zeichnet eine vertikale Linie zu der angegebenen Y-Position (Wert der X-Achse bleibt gleich).
<b>C x1 y1 x2 y2 x y</b>	Zeichnet eine kubische Kurve zu der angegebenen Position (x1, y1, x2 und y2 sind die Koordinaten der zwei Kontrollpunkte).
<b>S x2 y2 x y</b>	Zeichnet eine kubische Kurve zu der angegebenen Position (x2 und y2 sind die Koordinaten des zweiten Kontrollpunkts). Als erster Kontrollpunkt wird die „Spiegelung“ des zweiten Kontrollpunkts der vorherigen Kurve verwendet.
<b>Q x1 y1 x y</b>	Zeichnet eine quadratische Kurve zu der angegebenen Position (x1 und y1 sind die Koordinaten des Kontrollpunkts).
<b>T x y</b>	Zeichnet eine quadratische Kurve zu der angegebenen Position. Als Kontrollpunkt wird die Spiegelung des Kontrollpunkts der vorherigen Kurve verwendet.
<b>A rx ry rotate large sweep x y</b>	Zeichnet einen Bogen (an Hand zweier Ellipsen) zu der angegebenen Position (rx und ry sind der Radius). Über rotate lässt sich der Bogen drehen. large ist ein Flag, welches festlegt, ob der Bogen größer ist als 180° (1) oder nicht (0). sweep ist ebenfalls ein Bit und legt fest, ob bei einem negativen (1) oder positiven Winkel (0) begonnen werden soll zu zeichnen (hierdurch erfolgt „die Auswahl der Ellipse“).
<b>Z</b>	Schließt den Pfad (gerade Linie von der aktuellen Position zur Startposition).



**Übrigens:** Möchten Sie keine absolute Positionierung durchführen, sondern eine relative Positionierung in Bezug auf den letzten Punkt, dann müssen Sie als Kommandobuchstaben einfach einen Kleinbuchstaben verwenden (z. B. `m` statt `M`).

Bildquelle: [Vektor-Grafik von Freepik](#)

Im folgenden Beispiel wird zu allererst eine gerade Linie gezeichnet. Anschließend werden vier kubische Kurven gezeichnet, um eine Art Sinuswelle auf der Achse darzustellen:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" viewBox="0 0 600 400">
6   <path d="M 25 200 L 575 200 M 60 200 C 100 300 140 300 180 200 C 220 100 260 100 300 200 C 340 300 380 300 420 200 C 460
7   100 500 100 540 200" fill="none" stroke="blue" />
8 </svg>
    
```

**Wichtig:** Wenn Sie das obige Beispiel oder die Tabelle nicht gleich verstehen, ist das völlig normal. Probieren Sie einfach die verschiedenen Kommandos selbst aus und Sie werden die Funktionsweise besser verstehen.

**Übrigens:** Pfade besitzen standardmäßig eine Füllfarbe (i. d. R. `black`, also schwarz). Um die Füllfarbe zu entfernen, können Sie einfach den Wert `none` oder `transparent` im `fill`-Attribut angeben.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p><b>Web:</b> <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> <b>E-Mail:</b> <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: Homepage-Webhilfe » XML & Co. » SVG » Gruppierung

## Gruppierung

In SVG haben wir die Möglichkeit, Elemente (z. B. Linien und Rechtecke, aber auch Texte und Pfade) zu gruppieren. Eine solche Gruppierung kann für verschiedene Zwecke nützlich sein (z. B. zum Vererben oder zum Wiederverwenden). In diesem Thema werden Sie unterschiedliche Möglichkeiten zur Gruppierung und Wiederverwendung kennenlernen.

**Inhalt dieser Seite:**

1. Gruppe
2. Linkgruppe
3. Wiederverwendung
4. Symbol

### Gruppe

Beginnen wir zunächst mit der klassischen Gruppe. Hierfür benötigen wir das `g`-Element. Das `g`-Element ist zweiteilig und innerhalb des Elements können weitere Elemente (wie z. B. `line`, `rect`, `text` und `path`) notiert werden. Auch das Unterordnen von weiteren Gruppen ist möglich. Doch wozu brauche ich eine solche Gruppe? Eine Gruppe kann zu mehreren Zwecken dienen. Ein Vorteil einer Gruppe ist, dass alle Präsentationsattribute, welche im `g`-Element notiert sind, sich automatisch auf die Kindelemente auswirken. Stellen Sie sich vor, Sie haben eine Gruppe, in welcher Sie die Attribute `fill` und `stroke` festlegen. Allen Kindelementen (z. B. `rect`-Elementen) wird nun automatisch der Wert der Gruppe zugewiesen (siehe Beispiel unten). Man spricht hier auch von **Vererbung** (ähnlich wie bei HTML und CSS). Eine weitere praktische Möglichkeit zur Verwendung einer Gruppe ist die **Wiederverwendung** im Dokument, dazu jedoch [später mehr](#).

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5  <svg xmlns="http://www.w3.org/2000/svg" version="1.1" viewBox="0 0 600 400">
6      <g stroke="red" stroke-width="10" fill="blue">
7          <rect x="50" y="50" width="200" height="100" />
8          <rect x="300" y="100" width="250" height="250" />
9      </g>
10 </svg>
    
```

### Linkgruppe

Mit dem `a`-Element ist es möglich, in SVG-Grafiken einen **Link zu erzeugen**. Ein solcher Link ist im Prinzip nichts anderes als ein Hyperlink in HTML. Das `a`-Element ist, wie das `g`-Element auch, ein sogenanntes Container-Element und dient ebenfalls zur Gruppierung (es können mehrere Elemente untergeordnet werden). Oft wird dem `a`-Element jedoch trotzdem nur ein Element untergeordnet, da der Link z. B. nur für einen Text gelten soll. Um das Linkziel festzulegen, benötigen Sie das `href`-Attribut aus dem XLink-Namensraum. Über das Attribut `target` lässt sich festlegen, ob die verwiesene Seite im gleichen Fenster (`_self`, Standardwert) oder in einem neuen Fenster (`_blank`) angezeigt werden soll.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5  <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1" viewBox="0 0 600 400">
6      <text x="30" y="75" fill="lime" font-size="60px">Herzlich Willkommen</text>
7      <text x="275" y="150" fill="lime" font-size="40px">auf</text>
8      <a xlink:href="https://www.homepage-webhilfe.de/" target="_blank">
9          <text x="75" y="250" font-size="50px">
10             <tspan fill="red">Home</tspan><tspan fill="blue">page</tspan>
11             <tspan fill="yellow"></tspan>
12             <tspan fill="purple">Webhilfe</tspan>
13         </text>
14     </a>
15 </svg>
    
```

**Wichtig:** Ab SVG 2 ist die Verwendung des XLink-Namensraums nicht mehr notwendig und zudem auch als veraltet eingestuft. Es reicht hier dann die Notation des Attributs `href` (ohne Namensraum).

### Wiederverwendung

In SVG haben Sie auch die Möglichkeit, Formen, Texte und Pfade sowie Gruppen **einmal (global) zu definieren** und diese dann nachher **an beliebigen Stellen wieder einzusetzen**. Zuerst müssen wir unsere Elemente definieren: Dafür benötigen wir innerhalb des `svg`-Elements das `defs`-Element. In diesem ordnen wir nun die zu definierten Elemente unter (z. B. ein `rect`- oder `g`-Element) und geben diesen mit Hilfe des Attributs `id` jeweils einen **eindeutigen Namen**. An der Stelle, an welcher Sie nun den Grafikbestandteil wieder nutzen möchten, platzieren Sie das Element `use`. In diesem benötigen Sie nun das `href`-Attribut aus dem XLink-Namensraum (als Wert wird `#` gefolgt von der vergebenen ID angegeben). Des Weiteren benötigen Sie nun noch das Attribut `x` und `y`, um den Grafikbestandteil zu positionieren. Auch die Angabe von weiteren Attributen wie z. B. `fill` (um die Füllfarbe zu ändern) wäre denkbar.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5  <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1" viewBox="0 0 600 400">
6      <defs>
7          <rect id="bausteinRot" width="100" height="100" fill="red" stroke="purple" stroke-width="3" />
8          <rect id="bausteinBlau" width="100" height="100" fill="blue" stroke="purple" stroke-width="3" />
9      </defs>
    
```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---





# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [SVG](#) » [Gruppierung](#)

```

10 <use xlink:href="#bausteinRot" x="150" y="50" />
11 <use xlink:href="#bausteinBlau" x="150" y="150" />
12 <use xlink:href="#bausteinRot" x="150" y="250" />
13 <use xlink:href="#bausteinBlau" x="250" y="50" />
14 <use xlink:href="#bausteinRot" x="250" y="150" />
15 <use xlink:href="#bausteinBlau" x="250" y="250" />
16 <use xlink:href="#bausteinRot" x="350" y="50" />
17 <use xlink:href="#bausteinBlau" x="350" y="150" />
18 <use xlink:href="#bausteinRot" x="350" y="250" />
19 </svg>

```



**Wichtig:** Die im `defs`-Element definierten Elemente werden nicht angezeigt. Erst durch die Verwendung eines dort definierten Elements wird der Grafikbestandteil (also das Rechteck, die Gruppe etc.) sichtbar.

## Symbol

Eine meist wesentlich praktischere Möglichkeit zur Wiederverwendung von Grafikbestandteilen sind Symbole. Ein Symbol wird über das `symbol`-Element definiert und im Regelfall direkt dem `svg`-Element untergeordnet. Das `symbol`-Element ist, wie das `g`- und `a`-Element auch, ein Container-Element und gruppiert somit Elemente. Die „Einbindung“ eines Symbols erfolgt ebenfalls mit dem `use`-Element und somit auf die gleiche Art und Weise wie die Einbindung von anderen Elementen (siehe Beschreibung im [vorherigen Abschnitt](#)). Wird ein Symbol nicht eingebunden, so wird es auch niemals in der resultierenden Grafik angezeigt. Der gravierende Vorteil von Symbolen ist, dass das `viewBox`-Attribut verwendet werden kann. Daraus resultiert, dass das Symbol bei der Einbindung **nach Belieben skaliert werden kann** und somit unterschiedliche Größen haben kann.

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5 <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1" viewBox="0 0 600 400">
6   <symbol id="maennchen" viewBox="0 0 50 150">
7     <circle cx="25" cy="25" r="15" fill="none" stroke="black" />
8     <line x1="25" y1="40" x2="25" y2="95" stroke="black" />
9     <line x1="25" y1="50" x2="5" y2="75" stroke="black" />
10    <line x1="25" y1="50" x2="45" y2="75" stroke="black" />
11    <line x1="25" y1="95" x2="0" y2="140" stroke="black" />
12    <line x1="25" y1="95" x2="50" y2="140" stroke="black" />
13  </symbol>
14  <use xlink:href="#maennchen" x="50" y="50" width="100" height="300" />
15  <use xlink:href="#maennchen" x="260" y="110" width="80" height="240" />
16  <use xlink:href="#maennchen" x="450" y="50" width="100" height="300" />
17 </svg>

```



### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: Homepage-Webhilfe » XML & Co. » SVG » Stylesheet

## Stylesheet

In SVG-Dokumenten haben Sie die Möglichkeit, CSS zu nutzen, um die Elemente zu stylen. Dabei kommen nicht die „normalen Eigenschaften“ von CSS für HTML zum Einsatz. Vielmehr ist es möglich, dass alle **Präsentationsattribute** (wie z. B. fill, stroke, stroke-width, font-family und transform) auch als CSS-Eigenschaften notiert werden können. Um ein Element mit CSS zu stylen, können Sie, wie in HTML auch, das Attribut style verwenden. Die Notation fill="red" hat also die gleiche Auswirkung wie style="fill: red;". In CSS können wie gewohnt Element-Selektoren, Klassifizierungen, Identifikationen, Attribut-Selektoren und Kombinatoren verwendet werden. CSS-Anweisungen können Sie in einem SVG-Dokument, so wie in einem HTML-Dokument auch, im style-Element notieren.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5  <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1" viewBox="0 0 600 400">
6    <style type="text/css">
7      .rot
8      {
9        stroke: orange;
10       stroke-width: 2.5;
11       fill: red;
12     }
13     .blau
14     {
15       stroke: cyan;
16       stroke-width: 2.5;
17       fill: blue;
18     }
19   </style>
20   <symbol id="baustein" viewBox="0 0 100 100">
21     <rect x="5" y="5" width="90" height="90" rx="5" ry="5" />
22   </symbol>
23   <rect x="150" y="50" width="300" height="300" fill="black" />
24   <use xlink:href="#baustein" class="rot" x="150" y="50" width="100" height="100" />
25   <use xlink:href="#baustein" class="blau" x="150" y="150" width="100" height="100" />
26   <use xlink:href="#baustein" class="rot" x="150" y="250" width="100" height="100" />
27   <use xlink:href="#baustein" class="blau" x="250" y="50" width="100" height="100" />
28   <use xlink:href="#baustein" class="rot" x="250" y="150" width="100" height="100" />
29   <use xlink:href="#baustein" class="blau" x="250" y="250" width="100" height="100" />
30   <use xlink:href="#baustein" class="rot" x="350" y="50" width="100" height="100" />
31   <use xlink:href="#baustein" class="blau" x="350" y="150" width="100" height="100" />
32   <use xlink:href="#baustein" class="rot" x="350" y="250" width="100" height="100" />
33 </svg>

```

**Übrigens:** Ab SVG 2 können Sie mit CSS sogar die Attributwerte für die Positionierung und Größenangabe setzen. Dies wird jedoch von kaum einem Browser unterstützt.

Möchten Sie die CSS-Anweisungen in einem **separaten Dokument** notieren, dann können Sie die CSS-Datei über den XML-Verarbeitungshinweis `stylesheet` einbinden. Als Attribute benötigen Sie `href` und `type`.

```

1 | <?xml-stylesheet href="stylesheet.css" type="text/css" ?>

```

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eisingen</p> <p><b>Web:</b> <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> <b>E-Mail:</b> <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	--

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [SVG](#) » [Skripte](#)

## Skripte

Eine SVG-Grafik können Sie über ein Skript (im Regelfall JavaScript) **dynamisieren** (z. B. zur Bewegung oder Benutzerinteraktion). Der Skriptcode wird dazu, wie auch in HTML, zwischen den `script`-Tags notiert.

Als **Schnittstelle zwischen dem Dokument unter dem Skript** wird das DOM (*Document Object Model*) verwendet. Des Weiteren steht auch das BOM (*Browser Object Model*) zur Verfügung. Dadurch ist die grundlegende Verwendung der `document`- und `window`-Objekte gleich, wie wenn man JavaScript für HTML verwendet.

Um auf **Benutzeraktionen** (z. B. ein Klicken) zu reagieren, stehen uns natürlich einige Events zur Verfügung. Dazu zählen unter anderem `click` (klicken), `mousedown` (Mauktaste gedrückt), `mouseup` (Mauktaste losgelassen), `mousemove` (Maus bewegt) und `mouseout` (Maus verlässt das Element). Möchten Sie einem Element ein Ereignis zuordnen, so verwenden Sie als Attribut die Kombination aus dem Schlüsselwort `on` und dem Eventnamen.

Im folgenden Beispiel sehen Sie eine Analoguhr. Die Uhr wird automatisch gestartet und zeigt dann die aktuelle Uhrzeit. Sobald Sie auf die Uhr klicken, wird die Uhr angehalten. Ein weiterer Klick würde die Uhr wieder starten.

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2
3  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4
5  <svg xmlns="http://www.w3.org/2000/svg" version="1.1" viewBox="0 0 600 400">
6    <script type="text/javascript">
7      var bClockRunning = false;
8      var hIntv = null;
9
10     function StartStopClock()
11     {
12       // Uhr-Intervall starten oder stoppen
13       if (!bClockRunning)
14         hIntv = window.setInterval(SetClock, 1000);
15       else
16         window.clearInterval(hIntv);
17       // Bit umkehren
18       bClockRunning = !bClockRunning;
19     }
20
21     function SetClock()
22     {
23       var dt = new Date();
24       document.getElementById("stunde").setAttribute("transform", "rotate(" + ((dt.getHours() % 12) * 30) + " 300 200)");
25       document.getElementById("minute").setAttribute("transform", "rotate(" + (dt.getMinutes() * 6) + " 300 200)");
26       document.getElementById("sekunde").setAttribute("transform", "rotate(" + (dt.getSeconds() * 6) + " 300 200)");
27     }
28
29     // Uhr starten
30     StartStopClock();
31   </script>
32   <circle cx="300" cy="200" r="150" fill="silver" onclick="StartStopClock()" />
33   <line id="stunde" x1="300" y1="80" x2="300" y2="200" stroke="black" stroke-width="8" />
34   <line id="minute" x1="300" y1="60" x2="300" y2="200" stroke="black" stroke-width="3" />
35   <line id="sekunde" x1="300" y1="60" x2="300" y2="200" stroke="red" stroke-width="1" />
36   <circle cx="300" cy="200" r="8" fill="red" />
37 </svg>

```



Möchten Sie das Skript lieber in einer **externen Datei** notieren, so können Sie das Skript auch über das `href`-Attribut des `XLink`-Namensraums einbinden:

```

1 | <script type="text/javascript" xlink:href="skript.js"></script>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [XML & Co.](#) » [Abschluss](#)

## Abschluss

In diesem Kapitel bzw. in den Unterkapiteln haben Sie nun **Sprachen der XML-Familie kennengelernt**. Die Anzahl an XML basierten Sprachen ist enorm und natürlich können wir hier nicht alle vorstellen. Trotzdem haben wir versucht, die wichtigsten und geläufigsten Sprachen hier vorzustellen.

Die Inhalte der jeweiligen Kapitel haben sich im Großen und Ganzen auf das Wichtigste der jeweiligen Sprache beschränkt. Als **Referenz** für Elemente, Attribute, Attributwerte und Schlüsselwörter sind die **Spezifikationen vom W3C** zu empfehlen.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung

HTML

CSS

JavaScript

ActionScript

Java Applet

# Homepage Webhilfe

PHP

Perl

ASP.NET

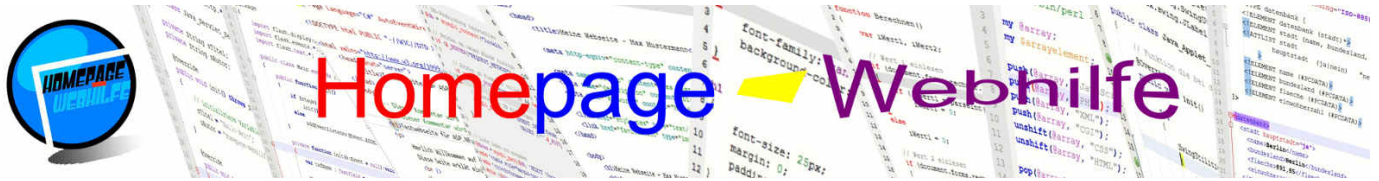
Java EE

XML

# E-Book

## Weiterführendes





Sie befinden sich hier: [Homepage-Webhilfe](#) » [Weiterführendes](#) » [Einleitung](#)

## Einleitung



In diesem Kapitel finden Sie einige interessante Themen, die teilweise aus komplett unterschiedlichen Bereichen kommen und sich vor allem an fortgeschrittene Webentwickler, Webadministratoren und Webdesigner richten. Prinzipiell soll dieses Kapitel die anderen Kapitel dieser Website abrunden und abschließen. Weitere ergänzende Themen, sowohl für Einsteiger als auch für Fortgeschrittene, finden Sie auch in unserem [Blog](#).

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

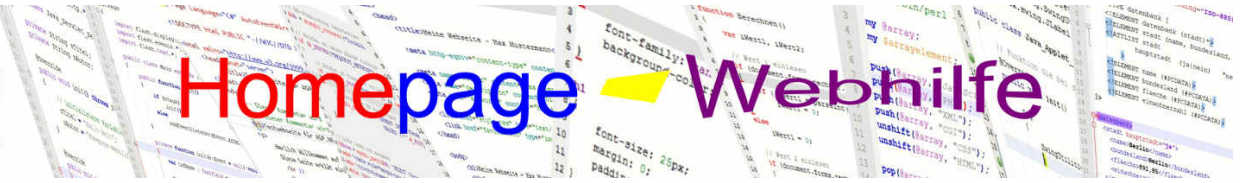
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislöcher



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Weiterführendes](#) » [Sonderzeichen](#)

## Sonderzeichen

Als Sonderzeichen werden im Allgemeinen Zeichen bezeichnet, welche **nicht direkt mit Hilfe einer Tastatur eingegeben** werden können. Trotzdem zählen z. B. auch die Zeichen Ä, ö und ü zu den Sonderzeichen, da diese z. B. nicht auf englischen Tastaturen zu finden sind. Zeichen, die nicht zu den Sonderzeichen zählen, sind also hauptsächlich die Buchstaben des lateinischen Alphabets sowie die Ziffern 0 bis 9. Um Sonderzeichen in HTML zu verwenden, gibt es verschiedene Möglichkeiten: direkte Notation, Angabe einer HTML-Entität (HTML-Zeichename) oder Angabe einer HTML-Nummern-Entität (NCR, *Numeric Character Reference*).

Welche (Sonder-)Zeichen in einer HTML-Seite zur Verfügung stehen, hängt natürlich auch von der Zeichenkodierung ab. Hierbei muss zwischen der **Zeichenkodierung der Datei** und der **Zeichenkodierung der Seite** (bzw. des definierten Zeichensatzes) unterschieden werden. Diese Unterscheidung ist vor allem dann wichtig, wenn es um die Direktnotation geht, denn in einer Datei, welche mit dem Zeichensatz ISO-8859-1 gespeichert ist, kann kein UTF-8-Zeichen gespeichert werden. Verwendet Ihre Seite aber als Zeichensatz UTF-8, so können Sie immer noch eine HTML-Entität oder HTML-Nummern-Entität verwenden, um das Sonderzeichen in der Seite zu platzieren. Verwenden Sie in der Datei als Zeichenkodierung ebenfalls UTF-8, so können Sie das Zeichen auch „direkt“ notieren. Trotzdem ist es aber möglich, eine HTML-Entität oder HTML-Nummern-Entität zu verwenden.

Des Weiteren gibt es in HTML (und XML) einige Zeichen, welche immer als HTML-Entität oder HTML-Nummern-Entität angegeben werden müssen. Diese Zeichen werden auch als **HTML-eigene Zeichen** bezeichnet.

Bildquelle: [Foto von Freepik](#)



Die folgende Tabelle zeigt eine Übersicht über oft verwendete Sonderzeichen inkl. deren HTML-Entitäten und HTML-Nummern-Entitäten (Dezimal und Hexadezimal):

Zeichen	Zeichenname	NCR (dez)	NCR (hex)	Beschreibung
<	&lt;	&#60;	&#x3C;	kleiner als ( <i>HTML-eigenes Zeichen</i> )
>	&gt;	&#62;	&#x3E;	größer als ( <i>HTML-eigenes Zeichen</i> )
&	&amp;	&#38;	&#x26;	Und-Zeichen ( <i>HTML-eigenes Zeichen</i> )
"	&quot;	&#34;	&#x22;	(doppeltes) Anführungszeichen ( <i>HTML-eigenes Zeichen, Direktnotation in HTML außerhalb von Attributen erlaubt</i> )
'	&apos;	&#39;	&#x27;	einfaches Anführungszeichen / Apostroph ( <i>HTML-eigenes Zeichen, Direktnotation in HTML außerhalb von Attributen erlaubt</i> )
	&nbsp;	&#160;	&#xA0;	geschütztes / erzwungenes Leerzeichen
Ä	&Auml;	&#196;	&#xC4;	A-Umlaut (groß)
ä	&auml;	&#228;	&#xE4;	A-Umlaut (klein)
Ö	&Ouml;	&#214;	&#xD6;	O-Umlaut (groß)
ö	&ouml;	&#246;	&#xF6;	O-Umlaut (klein)
Ü	&Uuml;	&#220;	&#xDC;	U-Umlaut (groß)
ü	&uuml;	&#252;	&#xFC;	U-Umlaut (klein)
ß	&szlig;	&#223;	&#xDF;	scharfes S
←	&larr;	&#8592;	&#x2190;	Pfeil nach links
→	&rarr;	&#8594;	&#x2192;	Pfeil nach rechts
↑	&uarr;	&#8593;	&#x2191;	Pfeil nach oben
↓	&darr;	&#8595;	&#x2193;	Pfeil nach unten
«	&laquo;	&#171;	&#xAB;	angewinkeltes Anführungszeichen nach links
»	&raquo;	&#187;	&#xBB;	angewinkeltes Anführungszeichen nach rechts
•	&bull;	&#8226;	&#x2022;	Aufzählungspunkt (Bullet)
©	&copy;	&#169;	&#xA9;	Copyright
®	&reg;	&#174;	&#xAE;	Registriermarken
™	&trade;	&#8482;	&#x2122;	Warenzeichen
§	&sect;	&#167;	&#xA7;	Paragraph
¤	&curr;	&#164;	&#xA4;	Währungszeichen
€	&euro;	&#8364;	&#x20AC;	Euro
£	&pound;	&#163;	&#xA3;	Pfund
¥	&yen;	&#165;	&#xA5;	Yen
¢	&cent;	&#162;	&#xA2;	Cent
¶	&para;	&#182;	&#xB6;	Absatzzeichen
±	&plusmn;	&#177;	&#xB1;	Plus / Minus

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislöding



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Weiterführendes](#) » [Sonderzeichen](#)

·	&middot;	&#183;	&#xB7;	Mittelpunkt
°	&deg;	&#176;	&#xB0;	Grad
∑	&sum;	&#8721;	&#x2211;	Summe
∏	&prod;	&#8719;	&#x220F;	Produkt
∞	&infin;	&#8734;	&#x221E;	Unendlich
≤	&le;	&#8804;	&#x2264;	kleiner-gleich
≥	&ge;	&#8805;	&#x2265;	größer-gleich
≠	&ne;	&#8800;	&#x2260;	ungleich
√	&radic;	&#8730;	&#x221A;	Wurzel
×	&times;	&#215;	&#xD7;	Mal
÷	&divide;	&#247;	&#xF7;	Division
f	&fnof;	&#402;	&#x192;	Funktion
μ	&micro;	&#181;	&#xB5;	Mikro
½	&frac12;	&#189;	&#xBD;	Ein-Halb
¼	&frac14;	&#188;	&#xBC;	Ein-Viertel
¾	&frac34;	&#190;	&#xBE;	Drei-Viertel
	&brvbar;	&#166;	&#xA6;	durchbrochener Senkrechtstrich

**Wichtig:** Nachdem heute in den Dateien für HTML-Seiten zumeist die Zeichenkodierung UTF-8 verwendet wird, wäre es möglich, die Sonderzeichen direkt zu notieren. Da diese aber meist nicht über eine Tastatur eingegeben werden können, werden trotzdem oft die Entitäten verwendet.

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

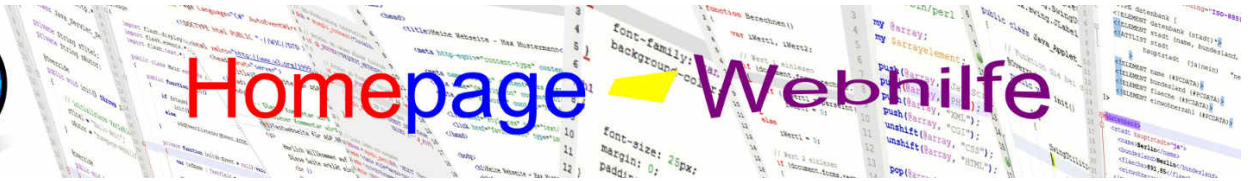
- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



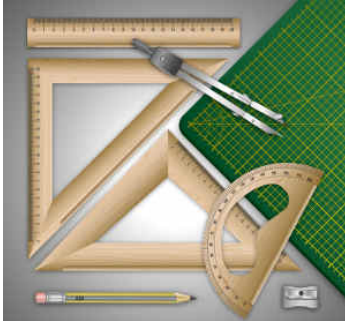


Sie befinden sich hier: [Homepage-Webhilfe](#) » [Weiterführendes](#) » [Maßeinheiten](#)

## Maßeinheiten

### Inhalt dieser Seite:

1. Umrechner



Wie Ihnen vermutlich bekannt ist, gibt es in den Stylesheet- und Auszeichnungssprachen, wie z. B. HTML, CSS, XSL-FO oder SVG, Maßeinheiten, mit welchen **Positionen, Längen und Abstände angegeben werden** können. Welche Maßeinheiten zur Verfügung stehen, hängt von der jeweiligen Sprache sowie ggf. dem Anzeigeprogramm (i. d. R. dem Browser) ab.

Bei Maßeinheiten wird zwischen absoluten und relativen Maßeinheiten unterschieden. **Absolute Maßeinheiten** sind auf jedem Ausgabegerät identisch, d. h. ein Bild mit einer Breite von `5cm`, hat auf einem Monitor mit einer Auflösung von 1024x768 die gleiche reale Größe, wie auf einem Monitor mit einer Auflösung von 1920x1080 sowie auf einem Ausdruck. Je nach Monitorgröße oder Mediengröße hat dies jedoch auch zur Folge, dass das Bild auf dem einen Monitor vielleicht die Hälfte der verfügbaren Bildschirmbreite ausfüllt und auf einem anderen nur ein Viertel. Zu den gängigen absoluten Maßeinheiten zählen Millimeter (`mm`), Zentimeter (`cm`), Inch bzw. Zoll (`in`), Punkt (`pt`) und Pica (`pc`). **Relative Maßeinheiten** hingegen sind variabel und beziehen sich auf andere Größenangaben (Angabe in einem Kindelement bezieht sich auf das Elternelement) oder auf das Ausgabemedium. Ein Bild mit einer Breite von `60%` würde also auf jedem Ausgabegerät (egal ob Monitor oder Papier) 60% der verfügbaren Breite ausfüllen. Dies hat jedoch zur Folge, dass das Bild auf einem Monitor in Realität evtl. 10cm breit ist und auf einem anderen Monitor dagegen schon 20cm. Bei relativen Einheiten sind vor allem Prozent (`%`), die Schriftgröße (`em`) und die x-Höhe (`ex`) verbreitet.

Eine weitere Einheit ist **Pixel** (`px`). Die Einheit Pixel lässt sich jedoch weder in die Gruppe absolute Maßeinheit, noch in die Gruppe relative Maßeinheit einordnen. Pixel ist jedoch trotzdem die Maßeinheit, welche **am meisten auf Webseiten verwendet** wird. Gerade in Bezug auf responsives Webdesign sollten jedoch vermehrt auch rein relative Maßeinheiten (wie z. B. Prozent) verwendet werden. Einerseits kann man Pixel als absolute Maßeinheit sehen, da die **Angabe nicht in Bezug auf ein Elternelement** getroffen wird. Jedoch erzeugt die Einheit Pixel **nicht auf jedem Ausgabegerät die gleiche reale (messbare) Größe**, weshalb man Pixel auch als relative Maßeinheit sehen kann. Der Grund dafür ist, dass die tatsächliche Größe zum einen von der Auflösung und zum anderen von der Größe des Ausgabemediums und somit also von der sogenannten **Pixeldichte** abhängt. Sie werden aber in Bezug auf diese Maßeinheit im Internet viele unterschiedliche Meinungen und Interpretationen finden.

Absolute Maßeinheiten können umgerechnet werden, so wissen auch Sie vermutlich, dass zehn Millimeter genau einem Zentimeter entsprechen. Relative Maßeinheiten können nicht untereinander umgerechnet werden, da diese von unterschiedlichen Faktoren abhängen. Die Einheit Pixel stellt ebenfalls wieder einen Sonderfall dar, da es sich hier um eine Art Mischung von absoluter und relativer Maßeinheit handelt. Die Umrechnung von einer absoluten Maßeinheit (wie z. B. Zentimeter) in Pixel (oder auch umgekehrt) wäre also nur durch die zusätzliche Angabe der Pixeldichte (`dpi`, *dots per inch*) möglich. Im Regelfall wird diese Umrechnung jedoch nicht benötigt.

Abschließend lässt sich sagen, dass **für Druckmedien hauptsächlich absolute Maßeinheiten** verwendet werden, wohingegen **bei Webseiten die Einheiten Pixel und relative Maßeinheiten** (wichtig, bei responsivem Webdesign) am häufigsten verwendet werden.

Bildquelle: [Vektor-Grafik von Freepik](#)

## Umrechner

Das folgende Tool erlaubt Ihnen die **Umrechnung zwischen den verschiedenen absoluten Maßeinheiten**. Geben Sie dazu einfach den Wert von der Einheit, von welcher Sie einen Wert in eine andere Einheit umrechnen möchten, in das gewünschte Feld ein und lesen Sie anschließend den Wert aus dem Feld ab, in welchen Sie den eingegebenen Wert umrechnen wollten.

Millimeter:	<input type="text"/>	mm
Zentimeter:	<input type="text"/>	cm
Inch / Zoll:	<input type="text"/>	in
Punkt:	<input type="text"/>	pt
Pica:	<input type="text"/>	pc

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Weiterführendes](#) » [Farben](#)

## Farben



Wert `#33AB77` kann hingegen nicht abgekürzt werden, da das zweite Byte (für den Grünanteil) nicht aus zwei gleichen Oktetts besteht. Eine weitere Notationsmöglichkeit ist die Verwendung der **RGB-Funktion**. Dabei wird das Schlüsselwort `rgb`, gefolgt von einem runden Klammernpaar, in welchem die drei Werte des RGB-Farbraums in dezimaler Schreibweise angegeben werden, notiert. Die Farbe `#33AA77` würde mit Hilfe der RGB-Funktion wie folgt aussehen: `rgb(51, 170, 119)`.

Ein weiteres Farbmodell ist HSL. Das **HSL-Farbmodell** ist unter Designern sehr bekehrt, da man sich die Farben viel einfacher vorstellen kann. Eine HSL-Farbangabe setzt sich ebenfalls aus drei Werten zusammen: dem Farbwert (*Hue*, Winkel auf dem Farbrad in Grad), der Sättigung (*Saturation*, Sättigung in Prozent) und die Helligkeit (*Lightness*, Helligkeit in Prozent). Der HSL-Farbraum wird nicht in allen Sprachen unterstützt, in CSS wird er jedoch unterstützt. Die Notation erfolgt mit Hilfe der HSL-Funktion. Dabei wird das Schlüsselwort `hsl`, gefolgt von einem Klammernpaar, in welchem die drei Werte angegeben werden, notiert. Für den Farbwert darf keine Einheit angegeben werden, für den Sättigungs- und Helligkeitswert muss jedoch das Prozentzeichen als Einheit angegeben werden. Dies sieht dann also bspw. so aus: `hsl(176, 34%, 58%)`.

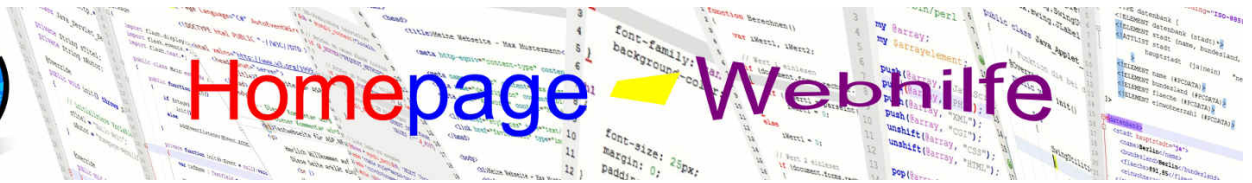
Die dritte und letzte Möglichkeit zur Angabe einer Farbe ist die Verwendung eines **Farbnamens**. Diese Methode ist besonders beliebt, da diese sehr einfach ist, denn wenn wir wissen, dass wir eine gelbe Farbe suchen, so müssen wir nicht lange überlegen, wie diese Farbe im RGB- oder HSL-Farbraum lautet, sondern müssen die Farbe lediglich ins Englische übersetzen. Das Ergebnis wäre also `yellow`. Nachteil dieser Variante ist natürlich, dass Farbangaben nicht genau abgestimmt werden können und natürlich nur eine begrenzte Anzahl an solchen vordefinierten Farben existiert. Trotzdem ist die Liste an Farbnamen lang: `red, green, blue, lime, yellow, orange, purple, cyan, black, white, usw.`

## Umrechner

Die Umrechnung zwischen dem RGB- und HSL-Farbmodell ist komplex. Deshalb stellen wir Ihnen hier ein kleines Tool zur Verfügung, mit welchem Sie RGB-Werte und HSL-Werte umrechnen können und umgekehrt. Sie müssen dazu lediglich die Felder für die RGB-Werte oder für die HSL-Werte ausfüllen und dann die Werte des anderen Farbraums ablesen.

Red (R)	<input type="text"/>
Green (G)	<input type="text"/>
Blue (B)	<input type="text"/>
Hue (H)	<input type="text"/> °
Saturation (S)	<input type="text"/> %
Lightness (L)	<input type="text"/> %

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislinsen</p> <p>Web: <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> E-Mail: <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Weiterführendes](#) » [Apache-Konfiguration](#)

## Apache-Konfiguration

Der Apache HTTP Server ist ein Webserver von der Apache Software Foundation (kurz ASF) und zählt zu den **verbreitetsten Webservern**. Jedoch hat im Laufe der letzten Jahre die Verwendung des Microsoft Internet Information Services (IIS) sowie die Verwendung anderer Webserver zugenommen. Falls Sie ein „normales“ bzw. durchschnittliches Hosting-Angebot nutzen, werden Sie jedoch mit sehr hoher Wahrscheinlichkeit Apache als Webserver haben. Apache ist plattformunabhängig und unter anderem auch im Software-Paket XAMPP enthalten.

In diesem Thema wollen wir uns damit beschäftigen wie man den Apache-Webserver konfigurieren kann. Eine der wichtigsten Konfigurationsdateien hierfür ist die Datei `.htaccess`. Die Datei `.htaccess` darf in jedem Verzeichnis vorkommen. Alle „Regeln“ die dort definiert sind, gelten dann für das Verzeichnis, in welchem die Datei hinterlegt ist inkl. allen Unterverzeichnissen.

### Inhalt dieser Seite:

1. Fehlerdokumente
2. MIME-Typen
3. Zugriffseinstellung
4. HTTPS-Weiterleitung
5. Verzeichnisauflistung

## Fehlerdokumente

In der `.htaccess`-Datei können Sie individuelle Fehlerdokumente festlegen. Diese Fehlerdokumente werden aufgerufen, wenn eine **HTTP-Antwort mit dem Statuscode 4xx oder 5xx** beantwortet werden würde. Belieb ist dieses Verfahren vor allem für den Statuscode 404, welcher auf das **Fehlen eines Verzeichnisses oder einer Datei** hinweist, für den Statuscode 403, welcher bei einer **Zugriffsverweigerung** auftritt, oder für den Statuscode 500, welcher auf einen (allgemeinen) **internen Serverfehler** hinweist. Ein Eintrag für ein solches Fehlerdokument besteht aus dem Schlüsselwort `ErrorDocument`, dem Statuscode und dem Pfad zum Fehlerdokument. Wird ein absoluter Pfad (z. B. `http://www.example.com/error-404.html`) angegeben, so erfolgt eine **Umleitung**. Bei der Angabe eines relativen Pfads (dazu zählt auch eine Pfadangabe mit Bezug auf das Wurzelverzeichnis, z. B. `/error-404.html`) erfolgt keine Umleitung, es wird stattdessen einfach der **Inhalt der angegebenen Fehlerseite an den Browser gesendet**.

```
1 | ErrorDocument 403 /zugriff-verweigert.html
2 | ErrorDocument 404 /nicht-gefunden.html
3 | ErrorDocument 500 /server-fehler.html
```

**Übrigens:** Wenn Sie statt einer Fehlerseite lediglich einen Text anzeigen möchten, dann können Sie auch einfach den anzuzeigenden Text (anzugeben in doppelten Anführungszeichen) notieren:

```
1 | ErrorDocument 404 "Die Seite wurde nicht gefunden!"
```

## MIME-Typen

Der Apache-Webserver enthält natürlich bereits eine **lange Liste an Zuordnungen** von MIME-Typen (z. B. `text/html`) zu Dateierweiterungen (z. B. `.html`). Manchmal kann es jedoch notwendig sein, eine solche **Zuordnung manuell hinzuzufügen**. Hierfür notieren Sie das Schlüsselwort `AddType`, gefolgt von dem MIME-Typ und der Angabe von einer oder mehreren Dateierweiterung(en).

```
1 | AddType text/plain .txt
2 | AddType text/html .html .htm
```

## Zugriffseinstellungen

Im Apache-Webserver können Sie Zugriffseinstellungen setzen, d. h. Sie können den externen Zugriff (Zugriff per HTTP, i. d. R. per Webbrowser) auf bestimmte Dateien und Ordner verbieten. Wird trotzdem auf eine Datei oder einen Ordner zugegriffen, auf welchen der Zugriff verweigert ist, so wird ein **403-Fehler ausgelöst**. Die wichtigsten Schlüsselwörter hierbei sind `allow` (Zugriff erlauben) und `deny` (Zugriff verbieten). Bevor wir jedoch die Zugriffseinstellungen setzen, müssen wir die **Reihenfolge** festlegen. Dafür dient das Schlüsselwort `order`. Mögliche Angaben sind daher also `order allow,deny` (erst erlauben, dann verbieten) und `order deny,allow` (erst verbieten, dann erlauben). Um eine Zugriffsregel zu definieren, benötigen Sie das Schlüsselwort `allow` oder `deny`, gefolgt vom Schlüsselwort `from` und einer IP-Adresse, einem DNS-Namen oder dem Schlüsselwort `all` (für alle). Die IP-Adressen und DNS-Namen können dabei auch abgekürzt werden. Hierzu ein Beispiel, bei welchem der Zugriff nur für die IP-Adressen `123.234.x.x` und den DNS-Namen `*.example.com` erlaubt sind:

```
1 | order deny,allow
2 | deny from all
3 | allow from 123.234.
4 | allow from *.example.com
```

Möchten Sie den Zugriff für alle Adressen verbieten, so kann Ihnen folgender Code helfen:

```
1 | order allow,deny
2 | deny from all
```

Natürlich können Sie den Zugriff auch mit der `FilesMatch`-Direktive kombinieren, um somit Zugriffseinstellungen z. B. nur für Dateien mit der Dateierweiterung `.php` anzuwenden:

```
1 | <FilesMatch \"\.(php)$">
2 |     order allow,deny
3 |     deny from all
4 | </FilesMatch>
```

## HTTPS-Weiterleitung

Eine oft gesuchte Konfiguration ist die **Umleitung von HTTP auf HTTPS**. Meist wird diese Einstellung nicht in der `.htaccess`-Datei, sondern eher in der Datei `httpd.conf` oder `httpd-vhosts.conf` gesetzt. Um eine solche Umleitung durchzuführen, benötigen Sie das **Rewrite-Modul**, mit welchem komplexe Regeln aufgestellt werden können. Die Konfiguration zur Umleitung von HTTP auf HTTPS sieht wie folgt aus:

```
1 | RewriteEngine On
2 | RewriteCond %{HTTPS} off
```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Weiterführendes](#) » [Apache-Konfiguration](#)

3 | `RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}`

## Verzeichnisauflistung

Im Regelfall ist es nicht erwünscht, dass man den **Inhalt eines Verzeichnisses über den Browser einsehen** kann (das sogenannte *directory listing*, zu Deutsch Verzeichnisauflistung), sofern der Ordner über **keine Verzeichnisindex-Datei** (zumeist `index.htm`, `index.html` oder `index.php`) verfügt. Ist die Verzeichnisauflistung deaktiviert und keine Verzeichnisindex-Datei vorhanden, so wird ein 403-Fehler (Zugriff verweigert) ausgelöst. Die Einstellung für die Verzeichnisauflistung kann mit der Option `+Indexes` (Auflistung aktivieren) und `-Indexes` (Auflistung deaktivieren) gesetzt werden.

1 | `Options -Indexes`

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislingen



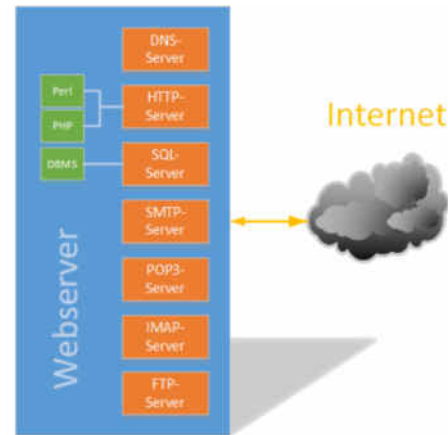
Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

Copyright & Design 2013 - 2017 by Homepage-Webhilfe, Benjamin Jung



Sie befinden sich hier: Homepage-Webhilfe » Weiterführendes » Funktionsweise eines Webservers

## Funktionsweise eines Webservers



### Inhalt dieser Seite:

1. HTTP-Server
2. SQL-Server
3. Mail-Server
4. DNS und Domain

Bei dem **klassischen Webserver** handelt es sich um einen HTTP-Server. Doch in der Zwischenzeit beinhaltet ein Webserver weitaus **mehr als nur einen HTTP-Server**. Dies liegt daran, dass nicht mehr mehrere Server eingesetzt werden, wobei dann jeder Server für eine einzelne Aufgabe zuständig ist, sondern **Server eingesetzt werden, welche mehrere Aufgaben auf einmal erledigen**. Ein „moderner Webserver“ beinhaltet also neben dem HTTP-Server i. d. R. einen DNS-Server (zum Beantworten von DNS-Anfragen), einen SQL-Server (zur Verwaltung von Datenbanken), einen SMTP-Server (zum Annehmen von E-Mails), einen POP3- und / oder IMAP-Server (zum Herunterladen von E-Mails) und einen FTP-Server (zum Hochladen der Website). In diesem Thema wollen wir die Funktion eines solchen modernen Webservers etwas genauer erklären.

### HTTP-Server

Der wohl wichtigste Teil eines Webservers ist der HTTP-Server (*HyperText Transfer Protocol*). Der HTTP-Server wartet auf Anfragen (engl. *request*) von Clients (z. B. einem Webbrowser), bearbeitet diese und sendet dann eine Antwort (engl. *response*) an den Client zurück. Bei dem **Bearbeitungsvorgang für eine Anfrage** wird natürlich auch geprüft, ob die angefragte Ressource verfügbar ist (mit Beachtung von Rewrite-Regeln). Sobald die Ressource gefunden wurde, wird, falls es sich um ein ausführbares Programm / Skript handelt (z. B. ein CGI-Skript), das Programm / Skript durch eine **externe Software** (z. B. einen Perl- oder PHP-Interpreter) ausgeführt und das Resultat zurückgegeben. Das Resultat kann dann, wie wenn es sich um eine statische Ressource handelt, an den HTTP-Client zurückgesendet

werden. Beispiel für HTTP-Server sind der Apache HTTP Server und der Microsoft Internet Information Service (IIS). Webserver verfügen i. d. R. über Schnittstellen, die dazu verwendet werden, mit Fremdsoftware (wie z. B. einem PHP-Interpreter) zu kommunizieren.

Die Kommunikation zwischen Client und Server läuft über das **HTTP-Protokoll**. Dies ist ein textbasiertes Protokoll und baut auf dem Transportprotokoll TCP auf. Als Port wird i. d. R. 80 verwendet. Das HTTPS-Protokoll ist eine verschlüsselte Variante des HTTP-Protokolls, wobei SSL oder TLS als Verschlüsselungsprotokoll zum Einsatz kommt. Der Standardport für HTTPS ist 443.

### SQL-Server

Als SQL-Server bezeichnet man Server, welche **Datenbanken verwalten** können und dabei die Datenbanksprache SQL beherrschen. SQL-Server sind von unterschiedlichen Herstellern verfügbar. Zu den verbreitetsten SQL-Servern zählen MySQL, MariaDB, PostgreSQL und Microsoft SQL Server (kurz MSSQL). Ein SQL-Server (im Allgemeinen auch als Datenbankserver bezeichnet) ist ein Programm, welches den Zugriff auf Datenbanken über das Netzwerk (SQL-Client passend zum verwendeten SQL-Server notwendig) erlaubt. Bei SQL-Servern, wie Sie auf einem modernen Webserver zu finden sind, ist jedoch aus Sicherheitsgründen der Zugriff von „außen“ nicht gestattet, d. h. nur der Webserver selbst (z. B. der PHP-Interpreter, welcher ebenfalls auf dem Server läuft) kann auf den SQL-Server zugreifen. Ein wichtiger Teil eines Datenbankservers, mal abgesehen von den Daten, ist das **Datenbankmanagementsystem** (abgekürzt DBMS), welches als Schnittstelle zwischen den Daten und dem Anwender bzw. den Anfragen angesehen werden kann. Für die Verwaltung des SQL-Servers bzw. dessen Daten stehen verschiedene Frontends zur Verfügung. Beispiel dafür sind MySQL Workbench und phpMyAdmin.

### Mail-Server

Als Mail-Server wird ein Server bezeichnet, welcher **E-Mails speichert, annimmt und versendet**. Auch die **Verwaltung von E-Mail-Adressen, Postfächern, Filterregeln etc.** gehört zur Aufgabe eines Mail-Servers. Um die einzelnen Aufgaben erledigen zu können, kommen auch verschiedene Netzwerkprotokolle zum Einsatz: SMTP (*Simple Mail Transfer Protocol*, Server zum Annehmen von E-Mails und Client zum Senden bzw. Weiterleiten von E-Mails), POP3 (*Post Office Protocol v3*, Server, um Nachrichten auf einen Client herunterladen zu können), IMAP (*Internet Message Access Protocol*, Server, um einem Client ein Dateisystem für die E-Mails bereitzustellen). Diese Protokolle sind auch in verschlüsselten Varianten verfügbar: SMTPS, POP3S, IMAPS. Ein Beispiel für eine Mail-Server-Software ist Mercury. Auch der IIS von Microsoft enthält einen Mail-Server.

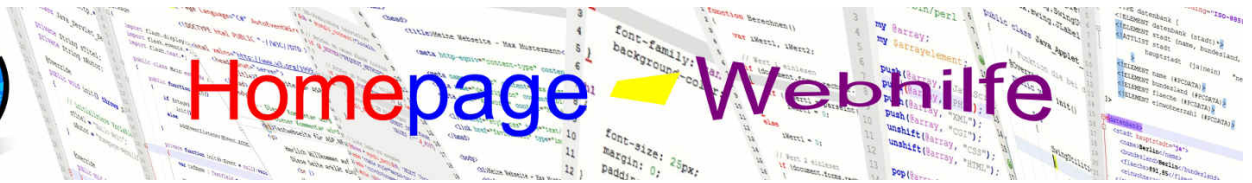
### DNS und Domain

Wenn Sie einen Webserver haben, auf welchem sich eine Website befindet, dann haben Sie i. d. R. auch einen Domainnamen. Ein „vollständiger“ Domainname (z. B. [www.homepage-webhilfe.de](http://www.homepage-webhilfe.de)), im Englischen auch als *Fully Qualified Domain Name* (FQDN) bezeichnet, ist im Prinzip nichts anderes als ein DNS-Name und somit ein Name, welcher sich aus der Zusammensetzung verschiedener Namesteile ergibt.

Zu jedem DNS-Namen gibt es mindestens eine IP-Adresse. Diese „Übersetzung“ eines Namens in eine IP-Adresse wird als **Namensauflösung** bezeichnet. Da die Kommunikation im Internet über IP-Adressen läuft und nicht über DNS-Namen, muss der Client, bevor er mit dem Webserver kommunizieren kann, den DNS-Namen in eine IP-Adresse auflösen. Die Vorteile von DNS-Namen im Gegensatz zu IP-Adressen sind, dass diese **leichter zu merken** sind (da Sie aus Zeichen oder sogar Wörtern bestehen, im Gegensatz zu IP-Adressen, die aus Zahlen bestehen) und **immer gleich bleiben** (IP-Adressen können sich z. B. durch Zwangstrennungen, einen Neustart oder eine Neuverteilung der Adressen durch den Provider ändern).

Die **Auflösung von DNS-Namen erfolgt schrittweise**. Dies liegt daran, dass das DNS-Namensverzeichnis hierarchisch aufgestellt ist. Der Auflösungsprozess beginnt beim letzten Teil des DNS-Namens und endet am ersten Teil (also von hinten nach vorne). Der erste Schritt bei der Domain [www.homepage-webhilfe.de](http://www.homepage-webhilfe.de) wäre also die Auflösung des Teils [.de](http://www.homepage-webhilfe.de). Dafür wendet sich ein DNS-Client zunächst an einen **Root-Namensserver**. Von diesen gibt es mehrere, die auf der ganzen Welt verteilt sind. Der Root-Namensserver gibt jetzt jedoch nicht die IP-Adresse der FQDN zurück, sondern die IP-Adresse von weiteren Namensservern, die für die Auflösung des DNS-Namens zuständig sind. In diesem Beispiel wären das die Server von DENIC, die die Top-Level-Domains (1. Ebene der DNS-Namen) [.de](http://www.homepage-webhilfe.de) verwalten. Dieser Vorgang wiederholt sich nun so lange, bis die IP-Adresse des Servers gefunden wurde.

<p><b>Über uns</b></p> <ul style="list-style-type: none"> <li>• Teamseite</li> <li>• Meinungen</li> <li>• Kontakt</li> <li>• Beratung</li> <li>• Impressum</li> <li>• Datenschutz</li> </ul>	<p><b>Community</b></p> <ul style="list-style-type: none"> <li>• Blog</li> <li>• Forum</li> <li>• News</li> </ul>	<p><b>Nachschlagewerk</b></p> <ul style="list-style-type: none"> <li>• Crashkurse</li> <li>• Glossar</li> <li>• FAQ</li> <li>• Karteikarten</li> <li>• E-Book</li> </ul>	<p>Benjamin Jung Krummstraße 9/3 73054 Eislingen</p> <p><b>Web:</b> <a href="https://www.homepage-webhilfe.de/">https://www.homepage-webhilfe.de/</a> <b>E-Mail:</b> <a href="mailto:info@homepage-webhilfe.de">info@homepage-webhilfe.de</a></p>
--	---	--	---



Sie befinden sich hier: [Homepage-Webhilfe](#) » [Weiterführendes](#) » [Suchmaschinenoptimierung](#)

## Suchmaschinenoptimierung



Die Suchmaschinenoptimierung (kurz SEO, engl. *Search Engine Optimization*) ist eines der **wichtigsten Maßnahmen für Websites, um diese im Ranking der Suchmaschinen weiter nach oben zu bringen**. Durch die Suchmaschinenoptimierung ist es also möglich, den Platz innerhalb der Suchrankings zu beeinflussen und dadurch die **Chancen auf neue Besucher** und „Beliebtheit“ zu erzielen.

Zur Suchmaschinenoptimierung gehören viele Techniken, Vorgehensweisen und Maßnahmen, welche genutzt werden können, um gute Plätze im Suchranking zu erlangen. In diesem Thema wollen wir einige dieser Möglichkeiten erläutern.

Es ist auch möglich, diese Technik zu missbrauchen, um somit sogar nicht relevante Seiten auf höhere Rankingplätze zu platzieren. Dies ist jedoch ein Verstoß gegen die **ethischen Regeln** der Suchmaschinenoptimierung und kann zum Ausschluss der kompletten Website aus dem Suchmaschinen-Index führen.

### Inhalt dieser Seite:

1. SEO für Webseiten
2. Weitere Optimierungen
3. Sitemap als Register

## SEO für Webseiten

Die wichtigste Suchmaschinenoptimierung ist die **Optimierung der einzelnen Seiten** einer Website (auch On-Page-Optimierung genannt). Die Webcrawler (auch Robots genannt) der Suchmaschinen, welche Ihre Webseiten untersuchen, um die Seiten im Index aufzunehmen und im Ranking einzuordnen, untersuchen dabei **sowohl die Struktur als auch den Inhalt Ihrer Seiten**. Diese beiden Faktoren sind also die primären Merkmale für die Einordnung im Ranking.

Die klassische Vorgehensweise ist das Setzen von Meta-Angaben (siehe [HTML-Tutorial](#)). Dazu gehören vor allem die Beschreibung (*description*) und die Schlüsselwörter (*keywords*), aber auch das ausdrückliche Erlauben zur Indexierung der Seite (*robots* mit dem Wert *index, follow*). In der Zwischenzeit werden diese Angaben jedoch nicht mehr von allen Suchmaschinen (u. a. auch Google) ausgewertet. Trotzdem sollten **die Meta-Angaben sowie der Titel nicht vernachlässigt werden**.

Wie bereits oben erwähnt, sind in der heutigen Zeit der Inhalt (engl. *content*) und die Struktur einer Seite die wichtigsten Faktoren für das Ranking. Die Webcrawler der Suchmaschinenanbieter arbeiten dabei mit komplexen Algorithmen, mit welchen die Seiten bewertet werden. Diese Algorithmen werden auch regelmäßig überarbeitet und verändert, um somit einen Missbrauch zu vermeiden. Sie sollten daher beim Formulieren Ihrer Inhalte auf **Qualität und Formatierung** (Überschriften, Absätze etc.) achten. Auch die **technischen Hintergründe**, wie z. B. der Aufbau und die Verschachtelung von HTML-Tags, die Angabe von *alt*-Attributen etc., sollten beachtet werden.

Da heutzutage **Smartphones und Tablets** von immer größerer Bedeutung werden, sollten Sie auch unbedingt darauf achten, dass Ihr Design auf unterschiedlichen Endgeräten und mit unterschiedlichen Browsern, Auflösungen und Seitenverhältnissen immer gleich gut aussieht (Stichwort [Responsives Webdesign](#)). Eine Seite mit responsivem Webdesign wird also immer höher gewertet.

## Weitere Optimierungen

Neben den bisher genannten Maßnahmen (On-Page-Optimierungen), die direkt auf den einzelnen Seiten angewendet werden, gibt es noch weitere Maßnahmen (Off-Page-Optimierungen), die **außerhalb der zu optimierenden Seite** durchgeführt werden.

Eine Maßnahme ist die **Platzierung von Backlinks**. Backlinks werden auf „fremden“ Websites platziert und verweisen auf die eigene Website. Solche Backlinks können Sie nur selten selbst platzieren. Meistens müssen Sie Personen finden, welche dazu bereit sind, einen solchen Backlink auf Ihrer Website zu setzen. Dies können z. B. Kunden oder (Geschäfts-)Partner sein. Eine ebenfalls beliebte Variante ist der **Linkaustausch**. Dabei tauschen sich zwei Personen gegenseitig einen Link aus und platzieren diesen auf Ihrer eigenen Website.

Des Weiteren werden in der Zwischenzeit sogar **soziale Netzwerke** beachtet. Dabei spielt es u. a. eine Rolle, ob der Betreiber selbst in sozialen Netzwerken aktiv ist und ob sich andere Nutzer über die Website austauschen (z. B. über Postings, Tweets und Kommentare).

Letztendlich wird natürlich auch das **Verhalten des Besuchers** analysiert: welche Links angeklickt werden, wie lange der Nutzer auf der Seite bleibt usw.. Auch die Anzahl der Klicks auf den Sucheintrag selbst werden bewertet.

## Sitemap als Register

Als Sitemap bezeichnet man eine **Übersicht der Webseiten auf einer Website bzw. einem Internetauftritt**. Da im Index der Suchmaschinen Milliarden von Seiten gelistet sind, würden die Webcrawler nur selten Ihre Website auf Änderungen prüfen. Daher wurde von Google eine Strategie bzw. ein Format (genannt Sitemaps-Protokoll) entwickelt, mit welchem die Suchanbieter über Änderungen „informiert“ werden können. Der Hauptbestandteil des **Sitemaps-Protokolls** ist ein auf XML basierendes Dateiformat, welches eine Sitemap (also die Seiten Ihrer Website) enthält. Dabei kann z. B. für jeden Eintrag ein Änderungsdatum hinterlegt werden. Weitere Suchanbieter wie z. B. MSN und Yahoo unterstützen das Sitemaps-Protokoll ebenfalls.

Das Wurzelement des XML-Format für das Sitemaps-Protokoll ist *urlset*. In diesem wird i. d. R. noch der Namensraum sowie ein Verweis zur XSD-Datei angegeben (siehe Beispiel unten). Die einzelnen Webseiten werden nun innerhalb des *urlset*-Elements durch das *url*-Element spezifiziert. In diesem können nun noch ein paar Elemente untergeordnet werden: *loc* (vollständige URL der Seite), *lastmod* (letzter Änderungszeitpunkt, Angabe im Format ISO 8601), *changefreq* (Änderungshäufigkeit der Seite, gültige Werte sind *always*, *hourly*, *daily*, *weekly*, *monthly*, *yearly* und *never*) und *priority* (Priorität (Priorität der Seite im Vergleich zu den anderen Links, numerischer Wert zwischen 0.0 und 1.0)). Das Element *loc* ist zwingend erforderlich. Hierzu nun ein Beispiel:

```

1 <?xml version="1.0" encoding="utf-8"?>
2
3 <urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd">
5   <url>
6     <loc>http://www.example.org/</loc>
7     <lastmod>2017-04-19T17:36:49+01:00</lastmod>
8     <changefreq>hourly</changefreq>
9     <priority>0.8</priority>
10  </url>
11 <url>
12   <loc>http://www.example.org/Produkte/</loc>
13   <lastmod>2017-04-11</lastmod>
14   <changefreq>weekly</changefreq>

```

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisingen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)



# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Weiterführendes](#) » [Suchmaschinenoptimierung](#)

```

14     </url>
15     <url>
16         <loc>http://www.example.org/Kontakt/</loc>
17         <lastmod>2017-04-01</lastmod>
18         <changefreq>never</changefreq>
19         <priority>0.3</priority>
20     </url>
21 </urlset>

```

**Wichtig:** Als Zeichenkodierung für Sitemaps muss UTF-8 verwendet werden. Des Weiteren ist das Format auf 50.000 URLs beschränkt.

Nachdem Sie eine solche Sitemaps-Datei erstellt haben und auf Ihrem Webserver an einem für alle Browser zugänglichen Ort (meistens das Wurzelverzeichnis) abgelegt haben, sollten Sie diese **bei den Suchanbietern (z. B. Google, Yahoo und Bing) einreichen**. Dafür müssen Sie bei den Anbietern i. d. R. ein kostenloses Konto erstellen. Danach können Sie über die angebotenen Tools (bei Google sind dies die Webmaster-Tools) die URL zur Sitemaps-Datei angeben und somit den Anbieter dazu auffordern, diese Sitemaps-Datei regelmäßig zu prüfen. Da Sie jedoch auch von anderen Suchanbietern (die Ihnen u. U. gar nicht bekannt sind) gelistet werden möchten, können Sie mit Hilfe der Datei `robots.txt` den **Pfad zu Ihrer Sitemap angeben**. Die Datei `robots.txt` ist eine einfache Textdatei, welche sich im Wurzelverzeichnis der Website befinden muss und zur Steuerung von Webcrawlern gilt. Der folgende Code zeigt, welche Zeile notwendig ist, um die Angabe für den Speicherort der Sitemaps-Datei anzugeben:

```
1 | Sitemap: http://www.example.org/sitemap.xml
```

**Übrigens:** Das Erstellen einer Sitemap kann manuell erfolgen oder über Tools von Drittanbietern. Ein Beispiel für ein solches Tool ist [phpSitemapNG](#).

**Wichtig:** Die Suchmaschinenoptimierung ist eine Daueraufgabe. Die Techniken entwickeln sich ständig weiter und die Schwerpunkte für die Bewertungskriterien von Suchmaschinen können sich im Laufe der Zeit ändern. Eine einmal optimierte Seite muss also immer wieder neu optimiert werden.

**Übrigens:** Die Anbieter von Suchmaschinen (z. B. Google) bieten oft Tools an, mit welchen das Nutzerverhalten und die Suchergebnisse aufgezeigt werden können. Die daraus gewonnenen Daten können sehr hilfreich für eine Suchmaschinenoptimierung sein.

## Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

## Community

- Blog
- Forum
- News

## Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eisligen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)

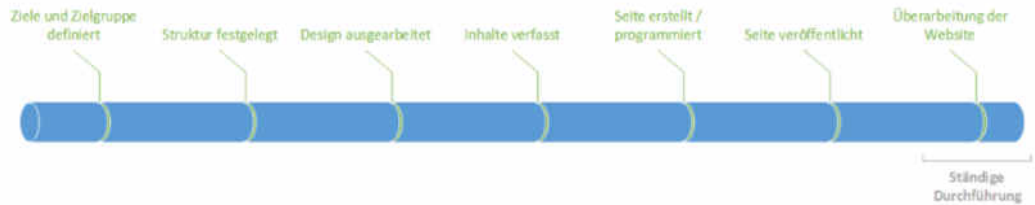


# Homepage-Webhilfe

Sie befinden sich hier: [Homepage-Webhilfe](#) » [Weiterführendes](#) » [Zeitschiene einer Website](#)

## Zeitschiene einer Website

Gerade Anfänger haben im Gebiet Websiteerstellung (egal ob Administrator, Designer oder Entwickler) **am Anfang viele Fragen** und wissen nicht wirklich, wo Sie anfangen sollen. Im [Kapitel Allgemein](#) von dieser Website haben wir deshalb versucht, Anfängern den Einstieg mit Hilfe eines **Konzepts für die Websiteerstellung** zu erläutern.



Im Kapitel Weiterführendes sind Sie nun **am Ende der regulären Kapitel** angelangt. Sie haben also die Planungsphasen, zu denen das Definieren der Ziele und Zielgruppe, das Ausarbeiten einer Struktur und das Festlegen des Erscheinungsbilds und Designs zählen, bereits hinter sich und haben vermutlich auch schon die Inhalte verfasst und Ihre Website erstellt bzw. programmiert. Sie stehen also jetzt gerade an dem Punkt, wo Sie Ihre Website an einigen Punkten vielleicht nochmals überarbeiten und nachbessern, bevor Sie diese dann ins Web stellen und somit für jeden zugänglich machen.

Natürlich möchten wir Ihnen hierzu zuallererst gratulieren, doch wir möchten Sie auch darauf hinweisen, dass **eine Website ständig überarbeitet werden sollte**. Eine solche Überarbeitung kann das Design, den Inhalt oder die technischen Hintergründe betreffen. Sie sollten sich also immer wieder informieren, ob es irgendwelche Neuerungen oder Änderungen gibt und versuchen, diese umzusetzen.

Wir hoffen, dass Sie viel Spaß auf unserer Website hatten und Sie viel dazu lernen konnten! Natürlich würden wir uns freuen, Sie weiterhin auf unserer Website begrüßen zu dürfen.

### Über uns

- Teamseite
- Meinungen
- Kontakt
- Beratung
- Impressum
- Datenschutz

### Community

- Blog
- Forum
- News

### Nachschlagewerk

- Crashkurse
- Glossar
- FAQ
- Karteikarten
- E-Book

Benjamin Jung  
Krummstraße 9/3  
73054 Eislungen



Web: <https://www.homepage-webhilfe.de/>  
E-Mail: [info@homepage-webhilfe.de](mailto:info@homepage-webhilfe.de)