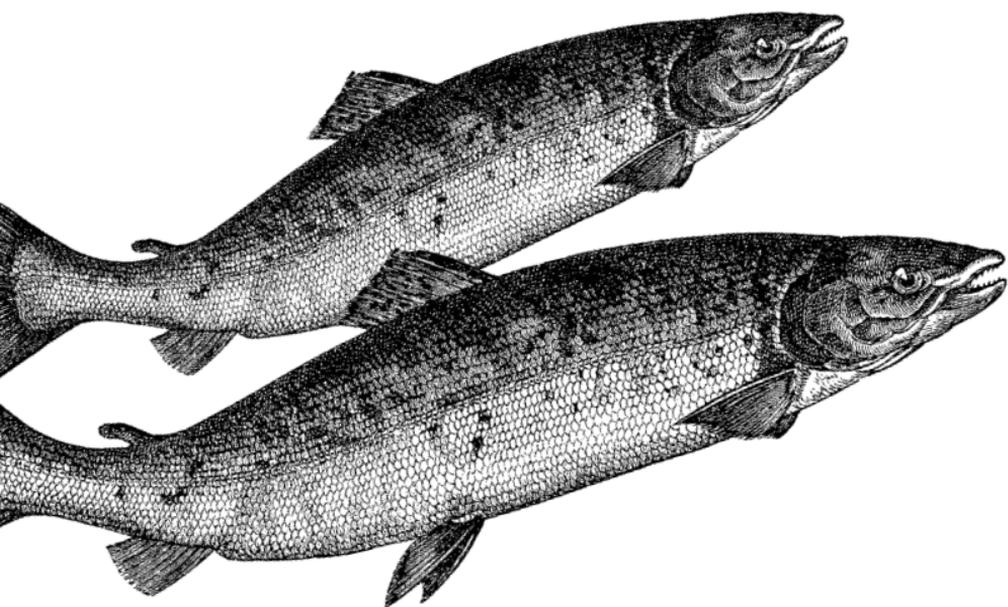


O'Reillys Taschenbibliothek

Deutsche Ausgabe
der 4. Auflage
Behandelt CSS2.1 & CSS3

CSS

kurz & gut



O'REILLY[®]

Eric A. Meyer

Übersetzung von Sascha Kersken

4. AUFLAGE

CSS
kurz & gut

Eric A. Meyer

*Deutsche Übersetzung
von Sascha Kersken*

O'REILLY®
Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

Die Informationen in diesem Buch wurden mit größter Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Verlag, Autoren und Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für eventuell verbliebene Fehler und deren Folgen.

Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen. Der Verlag richtet sich im Wesentlichen nach den Schreibweisen der Hersteller. Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Alle Rechte vorbehalten einschließlich der Vervielfältigung, Übersetzung, Mikroverfilmung sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Kommentare und Fragen können Sie gerne an uns richten:

O'Reilly Verlag GmbH & Co. KG

Balthasarstr. 81

50670 Köln

E-Mail: kommentar@oreilly.de

Copyright der deutschen Ausgabe:

© 2011 O'Reilly Verlag GmbH & Co. KG

1. Auflage 2001

2. Auflage 2005

3. Auflage 2009

4. Auflage 2011

Die Originalausgabe erschien 2011 unter dem Titel
CSS Pocket Reference, 4th Edition bei O'Reilly Media, Inc.

Die Darstellung von Lachsen im Zusammenhang mit dem Thema
CSS ist ein Wahrzeichen von O'Reilly Media, Inc.

Bibliografische Information Der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte bibliografische Daten
sind im Internet über <http://dnb.d-nb.de> abrufbar.

Übersetzung: Sascha Kersken, Köln

Lektorat: Inken Kiupel, Köln

Korrektorat: Tanja Feder, Bonn

Satz: Reemers Publishing Services GmbH, Krefeld (www.reemers.de)

Umschlaggestaltung: Michael Oreal, Köln

Produktion: Karin Driesen, Köln

Druck: fgb freiburger graphische betriebe; www.fgb.de

ISBN: 978-3-86899-144-4

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Vorwort	V
1 Grundlegende Konzepte	1
Stildefinitionen in HTML und XHTML einfügen	1
Der Aufbau einer Regel.....	5
Kommentare	6
Rangfolge der Stildefinitionen	7
Klassifizierung der Elemente	10
Darstellungsrollen der Elemente	11
Grundlagen der visuellen Darstellung	13
Floating	17
Positionierung.....	18
Tabellen-Layout	25
2 Werte	33
Schlüsselwörter	33
Farbwerte	34
Numerische Werte	36
Prozentwerte.....	36
Längenwerte	36
URLs.....	39
Winkel.....	39
Zeiten.....	40
Frequenzen.....	40
Strings.....	40

3	Selektoren	43
	Selektoren	43
	Struktur-Pseudoklassen	49
	Die Verneinungs-Pseudoklasse	56
	Interaktions-Pseudoklassen	57
	Pseudoelemente	61
	Media-Queries	64
4	Eigenschaften-Referenz	71
	Universalwerte	71
	Visuelle Medien	72
	Seitenbasierte Medien	214
	Akustische Medien	225
	Index	243

Cascading Style Sheets (CSS) ist der W3C-Standard zur visuellen Darstellung von Webseiten (der allerdings auch in anderen Umgebungen verwendet werden kann). Nach einer kurzen Einführung in die wichtigsten Konzepte von CSS finden Sie in diesem Buch eine alphabetische Referenz aller CSS3-Selektoren, gefolgt von einer alphabetischen Liste der CSS3-Eigenschaften.

Die im Buch behandelten Standards sind CSS2, CSS2.1 und CSS3. Da zum Zeitpunkt der Drucklegung dieses Buches nicht alle Browser alle CSS3-Eigenschaften und -Selektoren unterstützen, werden sie mit dem Hinweis »CSS3« hervorgehoben.

In diesem Buch verwendete Konventionen

In diesem Buch werden die folgenden typografischen Konventionen verwendet:

Kursiv

Kennzeichnet neue Begriffe, URLs, Dateinamen, Dateierweiterungen, Verzeichnisse, Befehle und Optionen sowie Programmnamen. Beispielsweise wird ein Pfad im Dateisystem als *C:\windows\system* dargestellt.

Nichtproportionalschrift

Wird verwendet, um den Inhalt von Dateien oder die Ausgabe von Befehlen zu kennzeichnen.

Nichtproportionalschrift kursiv

Wird verwendet, um Text darzustellen, der durch benutzerdefinierte oder durch den Kontext bestimmte Werte ersetzt werden soll.

Grundlegende Konzepte

Stildefinitionen in HTML und XHTML einfügen

Stildefinitionen können auf drei verschiedene Arten auf Dokumente angewendet werden; sie werden in den nachfolgenden Abschnitten besprochen.

Inline-Stildefinitionen

In HTML und XHTML können Stil-Informationen für ein einzelnes Element mit Hilfe des Attributs `style` angegeben werden. Der Wert eines `style`-Attributs ist ein Deklarationsblock (siehe den Abschnitt »Der Aufbau einer Regel« auf Seite 5) ohne geschweifte Klammern:

```
<p style="color: red; background: yellow;">Achtung!  
Dieser Text wird sehr auffällig dargestellt!</p>
```

Beachten Sie: Zum Zeitpunkt der Drucklegung dieses Buches ist es nicht möglich, in einem `style`-Attribut ein vollständiges Stylesheet unterzubringen. Nur der Inhalt eines einzelnen Deklarationsblocks kann als Wert eines `style`-Attributs verwendet werden. Beispielsweise ist es nicht möglich, Hover-Stildefinitionen (mittels `:hover`) in einem `style`-Attribut anzubringen, und auch `@import` kann in diesem Kontext nicht verwendet werden.

Typische XML-Dokumentsprachen (z.B. XHTML 1.0, XHTML 1.1 und SVG) unterstützen das `style`-Attribut zwar, aber es ist unwahrscheinlich, dass alle XML-Sprachen eine ähnliche Fähigkeit unter-

stützen werden. Aus diesem Grund, und weil es schlecht geschriebene Dokumente fördert, sollten Autoren das `style`-Attribut im Allgemeinen nicht verwenden.

Eingebettete Stylesheets

Mit Hilfe des `style`-Elements kann ein Stylesheet oben in einem HTML- oder XHTML-Dokument eingebettet werden; das Element muss innerhalb des `head`-Elements stehen:

```
<html><head><title>Stylen!</title>
<style type="text/css">
h1 {color: purple;}
p {font-size: smaller; color: gray;}
</style>
</head>
...
</html>
```

XML-Sprachen können eine entsprechende Fähigkeit zur Verfügung stellen oder auch nicht; überprüfen Sie daher sicherheits halber stets die Sprach-DTD.

Externe Stylesheets

Stildefinitionen können in einer separaten Datei aufgelistet werden. Der Hauptvorteil einer separaten Datei ist folgender: Das Sammeln häufig benutzter Stildefinitionen in einer einzelnen Datei ermöglicht die Aktualisierung aller Seiten, die dieses Stylesheet verwenden, durch das Bearbeiten eines einzelnen Stylesheets. Ein weiterer wichtiger Vorteil besteht darin, dass externe Stylesheets gecached werden, was bei der Reduzierung der Bandbreitennutzung helfen kann. Ein externes Stylesheet kann auf vier verschiedene Arten referenziert werden, die im Folgenden beschrieben werden.

Die `@import`-Direktive

Eine oder mehrere `@import`-Direktiven können am Anfang eines beliebigen Stylesheets platziert werden. In HTML- und XHTML-Dokumenten geschieht dies innerhalb eines eingebetteten Stylesheets:

```

<head><title>Mein Dokument</title>
<style type="text/css">
@import url(site.css);
@import url(navbar.css);
@import url(footer.css);
body {background: yellow;}
</style>
</head>

```

Beachten Sie, dass `@import`-Direktiven ganz oben (und gemäß der Spezifikation *nur* ganz oben) in einem beliebigen Stylesheet stehen können. Auf diese Weise könnte ein Stylesheet ein weiteres importieren, das wiederum ein drittes importieren könnte.

Das link-Element

In HTML- und XHTML-Dokumenten kann das Element `link` verwendet werden, um ein Stylesheet mit einem Dokument zu verknüpfen. Mehrere `link`-Elemente sind erlaubt. Das Attribut `media` kann verwendet werden, um ein Stylesheet auf ein oder mehrere Medien zu beschränken:

```

<head>
<title>Ein Dokument</title>
<link rel="stylesheet" type="text/css" href="basic.css"
      media="all">
<link rel="stylesheet" type="text/css" href="web.css"
      media="screen">
<link rel="stylesheet" type="text/css" href="paper.css"
      media="print">
</head>

```

Es ist auch möglich, alternative Stylesheets zu verknüpfen. Wenn alternative Stylesheets angegeben werden, ist es Sache des User Agents (oder des Autors), dem Benutzer ein Mittel an die Hand zu geben, eine der Alternativen zu wählen:

```

<head>
<title>Ein Dokument</title>
<link rel="stylesheet" type="text/css" href="basic.css">
<link rel="alternate stylesheet" title="Klassisch"
      type="text/css" href="altmodisch.css">
<link rel="alternate stylesheet" title="Futuristisch"
      type="text/css" href="jahr3000.css">
</head>

```

Während dies geschrieben wird, laden alle bekannten User Agents sämtliche verknüpften Stylesheets, ungeachtet dessen, ob der Benutzer sie je implementiert. Dies kann Auswirkungen auf Bandbreitennutzung und Serverlast haben.

Die Verarbeitungsanweisung `xml-stylesheet`

In XML-Dokumenten (wie beispielsweise XHTML-Dokumenten, die mit einem der Mime-Types `"text/xml"`, `"application/xml"` oder `"application/xhtml+xml"` gesendet werden) kann eine `xml-stylesheet`-Verarbeitungsanweisung verwendet werden, um ein Stylesheet mit einem Dokument zu verknüpfen. Jede `xml-stylesheet`-Verarbeitungsanweisung muss im Vorspann eines XML-Dokuments platziert werden. Es sind mehrere `xml-stylesheet`-Verarbeitungsanweisungen erlaubt. Das Pseudoattribut `media` kann verwendet werden, um ein einzelnes Stylesheet auf einen oder mehrere Medientypen zu beschränken:

```
<?xml-stylesheet type="text/css" href="basic.css"
  media="all"?>
<?xml-stylesheet type="text/css" href="web.css"
  media="screen"?>
<?xml-stylesheet type="text/css" href="paper.css"
  media="print"?>
```

Es ist auch möglich, mit Hilfe der Verarbeitungsanweisung `xml-stylesheet` alternative Stylesheets zu verknüpfen:

```
<?xml-stylesheet type="text/css" href="basic.css"?>
<?xml-stylesheet alternate="yes" title="Klassisch"
  type="text/css" href="altmodisch.css"?>
<?xml-stylesheet alternate="yes" title="Futuristisch"
  type="text/css" href="jahr3000.css"?>
```

HTTP-Link-Header

Die letzte (und bei weitem seltenste) Möglichkeit, ein externes Stylesheet mit Ihren Seiten zu verknüpfen, besteht darin, einen HTTP-Link-Header zu verwenden. Im Zusammenhang mit CSS handelt es sich dabei um eine Möglichkeit, die Wirkung eines `link`-Elements mit Hilfe von HTTP-Headern zu erzielen.

Wenn Sie eine Zeile wie die folgende zur `.htaccess`-Datei auf der Wurzelebene Ihres Servers hinzufügen, geschieht dies für alle Dateien der Site:

```
Header add Link
  "</style.css>;rel=stylesheet;type=text/css;media=all"
```

Da die Verwendung von `.htaccess` bekanntermaßen zu Performanceeinbußen führen kann, können Sie alternativ Ihre `httpd.conf`-Datei editieren, um dasselbe zu erreichen:

```
<Directory /usr/local/username/httpdocs>
Header add Link
  "</ style.css>;rel=stylesheet;type=text/css;media=all"
</Directory>
```

Dabei wird `/usr/local/username/httpdocs` durch den UNIX-Pfadnamen des tatsächlichen Stammverzeichnisses Ihrer Websites ersetzt.

Zum Zeitpunkt der Entstehung dieses Buchs wurden HTTP-Header nicht von allen User Agents unterstützt, insbesondere nicht von Internet Explorer und Safari. Deshalb ist dieses Verfahren auf Produktivumgebungen beschränkt, die andere User Agents verwenden, sowie auf gelegentliche Easter Eggs für Firefox- und Opera-Benutzer.

Der Aufbau einer Regel

Ein Stylesheet besteht aus einer oder mehreren Regeln, die beschreiben, wie Seitenelemente dargestellt werden sollen. Jede *Regel* besteht aus zwei grundlegenden Teilen: dem *Selektor* und dem *Deklarationsblock*. Abbildung 1-1 illustriert die Struktur einer Regel.

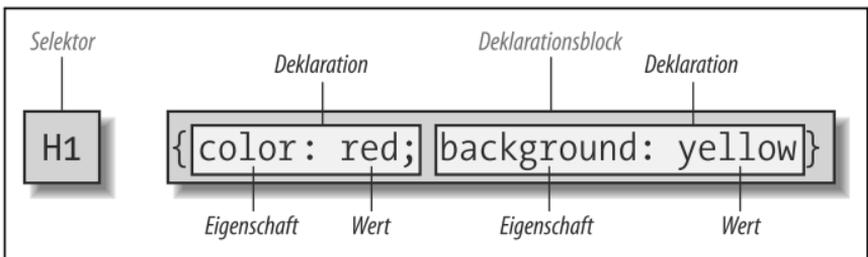


Abbildung 1-1: Aufbau einer Regel

Auf der linken Seite der Regel befindet sich der Selektor, der die Teile des Dokuments auswählt, auf die die Regel angewendet werden soll. Auf der rechten Seite der Regel haben wir den Deklarationsblock. Ein Deklarationsblock besteht aus einer oder mehreren *Deklarationen*; jede Deklaration ist eine Kombination aus einer *CSS-Eigenschaft* und einem *Wert* für diese Eigenschaft.

Der Deklarationsblock wird immer in geschweiften Klammern angeführt. Er kann mehrere Deklarationen enthalten; jede Deklaration muss durch ein Semikolon (;) abgeschlossen werden. Eine Ausnahme bildet die letzte Deklaration in einem Deklarationsblock, bei der das Semikolon optional ist.

Jede Eigenschaft, die einen bestimmten Stilparameter repräsentiert, wird durch einen Doppelpunkt von ihrem Wert getrennt. Bei Eigenschaftsnamen in CSS wird nicht zwischen Groß- und Kleinschreibung unterschieden. Zulässige Werte für eine Eigenschaft werden durch die Beschreibung der jeweiligen Eigenschaft definiert. Kapitel 4 liefert Details über mögliche Werte für CSS-Eigenschaften.

Kommentare

Das Einfügen von Kommentaren in CSS ist einfach. Sie beginnen mit `/*` und enden mit `*/`, wie folgt:

```
/* Dies ist ein Kommentar! */
```

Kommentare können sich über mehrere Zeilen erstrecken:

```
/* Dies ist ein Kommentar!  
   Dies ist eine Fortsetzung des Kommentars.  
   Und das hier auch. */
```

Sie können außerdem an jeder beliebigen Stelle innerhalb eines Stylesheets stehen, außer mitten in einem Token (Name oder Wert einer Eigenschaft):

```
h1/* Überschrift Stufe 1 */ {color /* Vordergrundfarbe */:  
  rgba(23,58,89,0.42) /* RGB + Deckkraft */;}
```

HTML-Kommentare (korrekte Bezeichnung: SGML-Kommentare, `<!--` zum Beispiel dieser `-->`) sind in Stylesheets erlaubt, um die Stildefinitionen vor Browsern zu verstecken, die so alt sind, dass sie

kein HTML 3.2 verstehen. Sie fungieren nicht als CSS-Kommentare; das heißt, dass alles, was von HTML-Kommentaren umschlossen wird, vom CSS-Parser gelesen und interpretiert wird.

Rangfolge der Stildefinitionen

Ein einzelnes HTML- oder XHTML-Dokument kann mehrere externe Stylesheets verknüpfen und importieren, ein oder mehrere eingebettete Stylesheets enthalten sowie Inline-Styles verwenden. In diesem Prozess ist es gut möglich, dass manche Regeln mit anderen in Konflikt geraten. CSS verwendet ein Verfahren namens *Kaskadierung*, um solche Konflikte zu lösen und schließlich einen Satz von Stildefinitionen auf das Dokument anzuwenden. Zwei wichtige Elemente der Kaskadierung sind *Spezifität* und *Vererbung*.

Berechnung der Spezifität

Spezifität beschreibt das Gewicht eines Selektors und der mit ihm verbundenen Deklarationen. Die folgende Tabelle fasst die Komponenten der Spezifitätsberechnung zusammen.

Art des Selektors	Beispiel	Spezifität
Universalselektor	*	0,0,0,0
Kombinator	+	
Element-Identifizier	div	0,0,0,1
Pseudoelement-Identifizier	::first-line	
Klassen-Identifizier	.warning	0,0,1,0
Pseudoklassen-Identifizier	:hover	
Attribut-Identifizier	[type="checkbox"]	
ID-Identifizier	#content	0,1,0,0
Inline-style-Attribut	style="color: red;"	1,0,0,0

Die Spezifitätswerte sind kumulativ; so hat etwa ein Selektor, der zwei Element-Identifizierer und einen Klassen-Identifizierer enthält (z.B. `div.aside p`), die Spezifität $0,0,1,2$. Spezifitätswerte sind in einer Rangfolge von rechts nach links angeordnet, daher hat ein Selektor mit 11 Element-Identifizierern $(0,0,0,11)$ eine niedrigere Spezifität als ein Selektor mit nur einem Klassen-Identifizierer $(0,0,1,0)$.

Die Direktive `!important` gibt einer Deklaration ein größeres Gewicht als Deklarationen ohne sie. Die Deklaration behält die Spezifität ihres Selektors bei, diese wird jedoch nur im Vergleich mit anderen `!important`-Deklarationen betrachtet.

Vererbung

Die Elemente in einem Dokument bilden eine baumartige Hierarchie, in der sich das Wurzelement an der Spitze befindet und der Rest der Dokumentstruktur sich darunter ausbreitet (wodurch das Ganze eigentlich eher wie ein Baumwurzelsystem aussieht). In einem HTML-Dokument bildet das Element `html` die Spitze des Baums, und die Elemente `head` und `body` entspringen daraus. Der Rest der Dokumentstruktur entspringt aus diesen Elementen. In einer solchen Struktur sind Elemente weiter unten im Baum Nachkommen ihrer Vorfahren, die sich weiter oben im Baum befinden.

CSS verwendet den Dokumentbaum für das Verfahren der *Vererbung*, bei der ein Stil, der auf ein Element angewendet wird, von dessen Nachkommen übernommen wird. Wenn beispielsweise das Element `body` den `color`-Wert `red` hat, wird dieser Wert den Dokumentbaum hinunter an die Elemente weitergereicht, die aus dem `body`-Element entspringen. Die Vererbung wird nur durch eine Stilregel unterbrochen, die sich direkt auf ein Element bezieht. Vererbte Werte haben überhaupt keine Spezifität (was *nicht* dasselbe ist wie eine Spezifität von null).

Beachten Sie, dass einige Eigenschaften nicht vererbt werden. Eine Eigenschaft definiert jeweils, ob sie vererbt wird. Einige Beispiele für nicht vererbte Eigenschaften sind `padding`, `border`, `margin` und `background`.

Kaskadierung

Die Kaskadierung ist das Verfahren, mit dem CSS Konflikte zwischen Stildefinitionen löst; es ist mit anderen Worten das Verfahren, mit dem ein User Agent beispielsweise entscheidet, welche Farbe ein Element erhalten soll, wenn zwei verschiedene Regeln darauf anwendbar sind und jede von ihnen versucht, eine andere Farbe festzulegen. Die Kaskadierung besteht aus folgenden Schritten:

1. Alle Deklarationen mit einem Selektor finden, der auf ein bestimmtes Element passt.
2. Alle Deklarationen, die auf das Element anwendbar sind, nach ihrem expliziten Gewicht sortieren. Die mit `!important` gekennzeichneten Regeln erhalten größeres Gewicht als diejenigen ohne diese Kennzeichnung. Außerdem werden alle Deklarationen, die auf das jeweilige Element passen, nach ihrer Herkunft sortiert. Es gibt drei Arten der Herkunft: Autor, Leser und User Agent. Unter normalen Umständen gewinnen die Stildefinitionen des Autors gegenüber denjenigen des Lesers. `!important`-Stildefinitionen des Lesers sind stärker als alle anderen Stile, einschließlich den `!important`-Stilen des Autors. Sowohl die Stildefinitionen des Autors als auch diejenigen des Lesers überschreiben die Standard-Stile des User Agents.
3. Alle Deklarationen, die auf das jeweilige Element anwendbar sind, nach ihrer Spezifität sortieren. Elemente mit einer höheren Spezifität haben größeres Gewicht als diejenigen mit einer niedrigeren.
4. Alle Deklarationen, die auf das jeweilige Element passen, nach ihrer Reihenfolge sortieren. Je später eine Deklaration in einem Stylesheet oder Dokument auftaucht, desto mehr Gewicht erhält sie. Deklarationen, die in einem importierten Stylesheet angeführt wurden, gelten als früher definiert als diejenigen in dem Stylesheet, das sie importiert, und sie haben ein niedrigeres Gewicht als diejenigen im importierenden Stylesheet.

Jegliche Darstellungshinweise, die aus Nicht-CSS-Quellen stammen (z.B. der Voreinstellungsdialog eines Browsers) erhalten dasselbe Gewicht wie die Standard-Stile des User Agents (siehe Schritt 2 oben).

Klassifizierung der Elemente

Allgemein ausgedrückt unterteilt CSS Elemente in zwei verschiedene Gruppen: *nicht ersetzte* und *ersetzte*. Wenngleich diese Gruppen ziemlich abstrakt erscheinen mögen, gibt es tatsächlich einige grundlegende Unterschiede in der Art und Weise, wie die beiden Typen von Elementen dargestellt werden. Diese Unterschiede werden detailliert in Kapitel 7 von CSS – Das umfassende Handbuch, deutsche Ausgabe der 3. Auflage (O'Reilly) untersucht.

Nicht ersetzte Elemente

Der größte Teil der HTML- und XHTML-Elemente besteht aus *nicht ersetzten Elementen*. Das bedeutet, dass ihr Inhalt vom User Agent in einer Box dargestellt wird, die das Element selbst erzeugt. Beispielsweise ist `Hallo Leute` ein nicht ersetztes Element, und der Text Hallo Leute wird vom User Agent dargestellt. Abschnitte, Überschriften, Tabellenzellen, Listen und fast alles andere in HTML und XHTML sind nicht ersetzte Elemente.

Ersetzte Elemente

Im Gegensatz dazu sind *ersetzte Elemente* diejenigen, deren Inhalt durch etwas ersetzt wird, das nicht direkt durch Dokumentinhalt dargestellt wird. Das bekannteste HTML-Beispiel ist das Element `img`, das durch eine Bilddatei ersetzt wird, die außerhalb des Dokuments selbst liegt. Tatsächlich hat `img` selbst keinen Inhalt, was wir durch die Betrachtung eines einfachen Beispiels sehen können:

```

```

Es gibt keinen Inhalt in diesem Element – nur einen Elementnamen und Attribute. Nur über das Ersetzen des fehlenden Elementinhalts durch Inhalt, der mit Hilfe anderer Mittel gefunden wird (in diesem Fall das Laden eines externen Bildes, das durch das Attribut `src` angegeben wird) kann das Element überhaupt dargestellt werden. Ein weiteres Beispiel ist das Element `input`, das durch einen Radio-

button, eine Checkbox oder ein Texteingabefeld ersetzt wird, je nach seinem Typ. Ersetzte Elemente erzeugen in ihrer Ausgabe ebenfalls Boxen.

Darstellungsrollen der Elemente

Außer in ersetzte oder nicht ersetzte Elemente unterteilt man Elemente in CSS3 auch in zwei grundlegende Arten von Darstellungsrollen: *Block-Elemente* und *Inline-Elemente*. Alle `display`-Werte in CSS3 gehören zu einer dieser beiden Kategorien. Es kann wichtig sein, zu wissen, zu welcher dieser Rollen eine Box gehört, da einige Eigenschaften nur auf jeweils eine von ihnen anwendbar sind.

Block-Elemente

Block-Elemente sind diejenigen, die (standardmäßig) die Breite des Inhaltsbereichs ihres Elternelements ausfüllen, so dass sich keine anderen Elemente neben ihnen befinden können. Block-Elemente bilden mit anderen Worten »Umbrüche« vor und hinter der Element-Box. Die bekanntesten Block-Elemente in HTML sind `p` und `div`. Ersetzte Elemente können Block-Elemente sein, sind jedoch üblicherweise keine.

Listeneinträge sind ein Sonderfall der Blockelemente. Sie verhalten sich in fast jeder Hinsicht genauso wie andere Block-Elemente, erzeugen aber zusätzlich eine Markierung – üblicherweise einen Listenpunkt bei Aufzählungen oder eine Zahl bei nummerierten Listen –, die an die Element-Box »angefügt« werden. Abgesehen von diesen Markierungen sind Listeneinträge mit anderen Block-Elementen identisch.

Die `display`-Werte, die Block-Boxen erzeugen, sind: `block`, `list-item`, `table`, `table-row-group`, `table-header-group`, `table-footer-group`, `table-row`, `table-column-group`, `table-column`, `table-cell`, `table-caption` und (zum Zeitpunkt des Schreibens dieses Buches) alle CSS-Advanced-Layout-Templates.

Inline-Elemente

Inline-Elemente sind solche, deren Element-Box innerhalb einer Textzeile erzeugt wird, wobei der Fluss dieser Zeile nicht unterbrochen wird. Das bekannteste Inline-Element ist vielleicht das Element `a` in HTML und XHTML. Weitere Beispiele sind `span` und `em`. Vor oder hinter diesen Elementen wird kein Umbruch erzeugt, so dass sie innerhalb des Kontextes eines anderen Elements erscheinen können, ohne dessen Darstellung zu unterbrechen.

Beachten Sie, dass CSS-Block- und Inline-Elemente zwar viel mit HTML- und XHTML-Block- und Inline-Elementen gemeinsam haben, es jedoch einen wichtigen Unterschied gibt. In HTML und XHTML können Block-Elemente keine Nachfahren von Inline-Elementen sein, während es in CSS keine Einschränkungen bezüglich der Verschachtelung von Darstellungsrollen gibt.

Die `display`-Werte, die Inline-Boxen erzeugen, sind: `inline`, `inline-block`, `inline-table` und `ruby`. Zum Zeitpunkt der Entstehung dieses Buches gab es noch keine explizite Definition dafür, dass die diversen spezielleren Ruby-Werte (z.B. `ruby-text`) ebenfalls Inline-Boxen erzeugen, aber dies scheint die wahrscheinlichste Entwicklung zu sein.

Run-In-Elemente

Einen Sonderfall bilden die durch `display: run-in` definierten *Run-In-Boxen*, die je nach Situation entweder eine Block- oder eine Inline-Box erzeugen können. Die Regeln, die das Ergebnis bestimmen, sind folgende:

1. Wenn das Run-In selbst eine Block-Box enthält, dann erzeugt es eine Block-Box.
2. Wenn dies nicht der Fall ist und auf das Run-In direkt ein gleichrangiges Block-Element folgt, das nicht gefloatet oder absolut positioniert ist, dann wird die Run-In-Box zur ersten Inline-Box der gleichrangigen Block-Box.
3. Wenn keine der beiden vorigen Bedingungen zutrifft, erzeugt das Run-In eine Block-Box.

Für den Fall, dass ein Run-In als erstes Inline-Element einer gleich-rangigen Block-Box eingefügt wird (siehe Regel 2 oben), erbt es *nicht* die Eigenschaftswerte dieser Block-Box. Stattdessen erbt es weiterhin von seinem Elternelement in der Struktur.

Grundlagen der visuellen Darstellung

CSS definiert Algorithmen für das Layout aller Elemente in einem Dokument. Diese Algorithmen bilden das Fundament der visuellen Darstellung in CSS. Es gibt zwei grundlegende Arten des Layouts mit unterschiedlichem Verhalten: Block-Layout und Inline-Layout.

Layout für Block-Elemente

Eine Block-Box erzeugt in CSS eine rechteckige Box, die als Element-Box bezeichnet wird und die Menge des von dem Element belegten Platzes definiert. Abbildung 1-2 zeigt die diversen Bestandteile einer Element-Box. Für eine Element-Box gelten die folgenden Regeln:

- Der Hintergrund der Element-Box erstreckt sich bis zur Außenkante des Rahmens und füllt so die Bereiche des Inhalts, des Innenabstands und des Rahmens. Wenn der Rahmen transparente Stellen hat (z.B. gepunktet oder gestrichelt ist), ist der Hintergrund an diesen Stellen sichtbar. Der Hintergrund erstreckt sich nicht auf den Außenabstand der Box. Etwaige Konturen werden im Bereich des Außenabstands dargestellt und beeinflussen das Layout nicht.
- Nur die Außenabstände sowie die `height-` und `width-`Werte einer Element-Box können auf `auto` gesetzt werden.
- Nur die Außenabstände können negative Werte erhalten.
- Die Stärke der Innenabstands- und Rahmen-Werte der Element-Box beträgt standardmäßig 0 (null) beziehungsweise `none`.
- Wenn `box-sizing` den Wert `content-box` hat (Standardwert), definiert die Eigenschaft `width` nur die Breite des Inhaltsbereichs; sämtliche Innenabstände, Rahmen oder Außenabstände werden hinzuaddiert.

- Wenn box-sizing den Wert border-box hat, definiert die Eigenschaft width die Gesamtbreite von Inhalt, Innenabständen und Rahmen; die jeweiligen Außenabstände werden hinzugefügt. Dasselbe gilt für height in Bezug auf die Höhe.

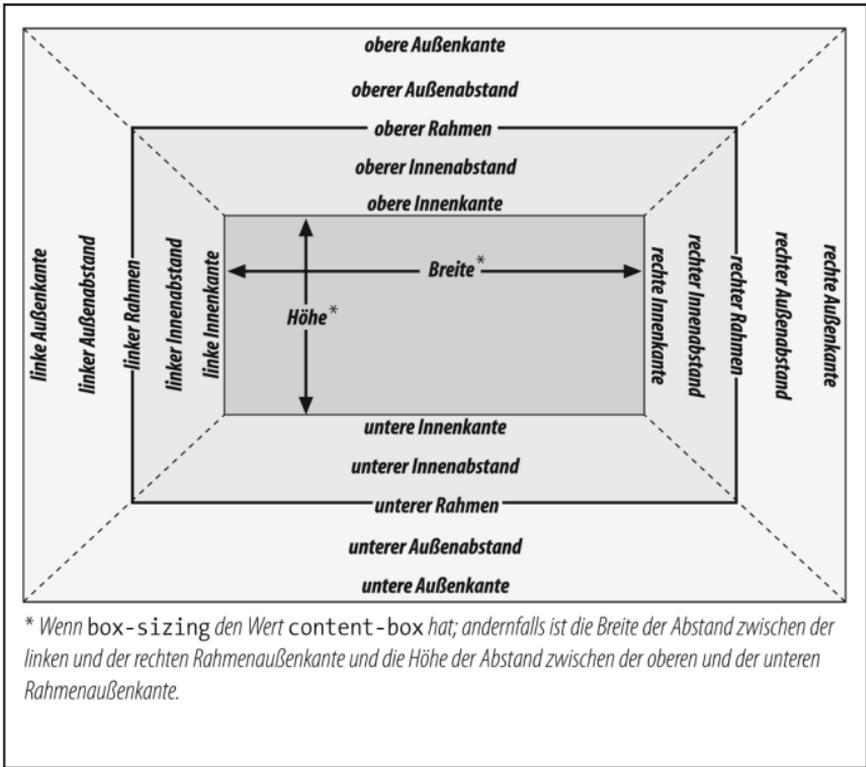


Abbildung 1-2: Details des Box-Modells

Layout für Inline-Elemente

Eine *Inline-Box* in CSS erzeugt eine oder mehrere rechteckige Boxen, je nachdem, ob sich die Inline-Box über mehrere Zeilen erstreckt. Für Inline-Boxen gelten folgende Regeln:

- Für die Eigenschaften left, right, top, bottom, margin-left, margin-right, margin-top und margin-bottom wird der Wert auto jeweils zu 0 (null) konvertiert.
- width und height gelten nicht für nicht ersetzte Inline-Boxen.

- Für ersetzte Inline-Boxen gelten die folgenden Regeln:
 - Wenn sowohl `width` als auch `height` den Wert `auto` haben und das Element eine immanente Breite besitzt (z.B. ein Bild), dann entspricht der Wert von `width` der immanenten Breite des Elements. Dasselbe gilt für `height`.
 - Wenn sowohl `height` als auch `width` den Wert `auto` haben und das Element keine immanente Breite, aber eine immanente Höhe und ein Layout-Seitenverhältnis besitzt, dann wird `width` auf das Produkt aus der immanenten Höhe und diesem Verhältnis gesetzt.
 - Wenn sowohl `height` als auch `width` den Wert `auto` haben und das Element keine immanente Höhe, aber eine immanente Breite und ein Layout-Seitenverhältnis besitzt, dann wird `height` auf das Produkt aus der immanenten Breite und diesem Verhältnis gesetzt.

Es gibt einige Regeln, die sogar noch undurchsichtiger sind als die letzten beiden. Sie sind jedoch zu umfangreich, um sie hier aufzuführen; siehe <http://w3.org/TR/css3-box/#inline-replaced> für Details.

Alle Inline-Elemente besitzen eine `line-height`, die maßgeblich bestimmt, wie die Elemente dargestellt werden. Die Höhe einer Textzeile wird unter Einbeziehung folgender Faktoren festgelegt:

Anonymer Text

Jede Zeichenkette, die nicht in einem Inline-Element enthalten ist. Beispielsweise sind in dem Markup

```
<p> Ich bin <em>so</em> glücklich!</p>
```

die Sequenzen »Ich bin« und »glücklich!« anonymer Text. Beachten Sie, dass auch die Leerzeichen Teil des anonymen Texts sind, denn ein Leerzeichen ist ein Zeichen wie jedes andere auch.

Em-Box

Die vom jeweiligen Zeichensatz definierte Em-Box, auch Zeichen-Box genannt. Die eigentlichen Zeichen können höher oder niedriger sein als ihre Em-Boxen; dies wird in Kapitel 5 von *CSS – Das umfassende Handbuch*, deutsche Ausgabe der 3. Auflage (O'Reilly) besprochen.

Der Inhaltsbereich

In nicht ersetzten Elementen kann der Inhaltsbereich die Box sein, die durch alle zusammengeführten Em-Boxen aller Zeichen des Elements beschrieben wird, oder aber die Box, die von allen Zeichen im Element beschrieben wird. Die CSS2.1-Spezifikation stellt es User Agents frei, eine der beiden Varianten zu wählen. Um den Text einfach zu halten, beschränken wir uns hier auf die Em-Box-Methode. In ersetzten Elementen entspricht der Inhaltsbereich der immanenten Höhe des Elements plus sämtlicher Außenabstände, Rahmen oder Innenabstände.

Der Durchschuss

Der Durchschuss ist die Differenz zwischen den Werten von `font-size` und `line-height`. Jeweils die Hälfte dieser Differenz wird auf die obere beziehungsweise untere Hälfte des Inhaltsbereichs angewendet. Diese Erweiterungen des Inhaltsbereichs heißen – wenig überraschend – Halb-Durchschuss. Der Durchschuss wird nur auf nicht ersetzte Elemente angewendet.

Die Inline-Box

Die durch die Addition von Inhaltsbereich und Durchschuss entstehende Box bezeichnet man als *Inline-Box*. Bei nicht ersetzten Elementen entspricht ihre Höhe dem Wert für `line-height`. Bei ersetzten Elementen entspricht die Höhe der Inline-Box der Höhe des Inhaltsbereichs, weil in diesem Fall kein Durchschuss zugerechnet wird.

Die Zeilen-Box

Die Höhe der Zeilen-Box entspricht dem Abstand zwischen dem höchsten und dem niedrigsten Punkt aller Inline-Boxen, die sich in einer Zeile befinden. Die Oberkante der Zeilen-Box schließt also bündig mit der Oberkante der höchsten Inline-Box ab, während die Unterkante der Zeilen-Box mit der Unterkante der niedrigsten Inline-Box bündig abschließt (siehe Abbildung 1-3).

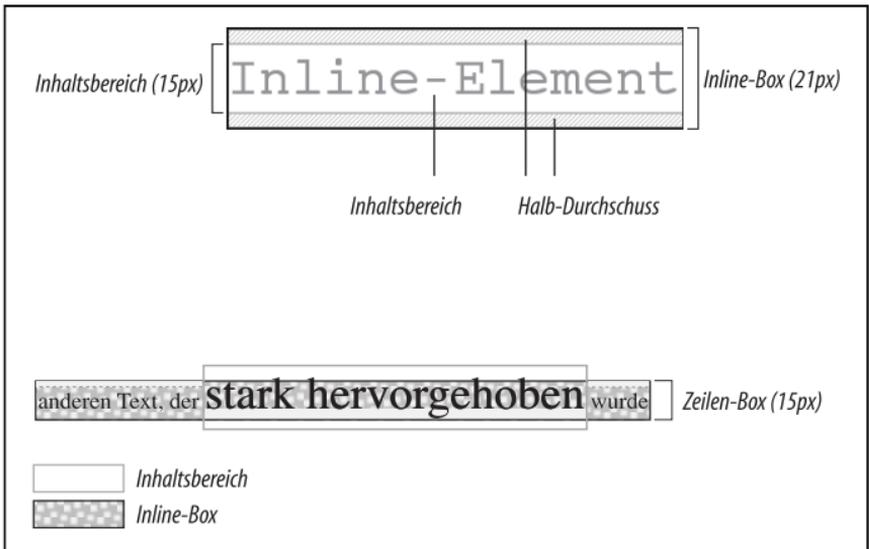


Abbildung 1-3: Details der Darstellung von Inline-Elementen

Floating

Floating ermöglicht es, ein Element links oder rechts von seinem umgebenden Block zu platzieren (dies ist das nächste Vorfahrenelement, das ein Block-Element ist), wobei der nachfolgende Inhalt das Element umfließt. Ein Floating-Element erzeugt automatisch eine Block-Box, ungeachtet des Box-Typs, den es erzeugen würde, wenn es nicht gefloatet wäre. Ein Floating-Element wird gemäß den folgenden Regeln platziert:

- Die seitlichen Außenkanten des Floats dürfen nicht über die seitlichen Innenkanten des umgebenden Blocks hinausragen.
- Die einander zugewandten Außenkanten zweier Floats, die beide links oder beide rechts angeordnet wurden, dürfen sich nicht überschneiden. Diese Regel gilt nicht, wenn das zweite Element direkt unter dem ersten steht.
- Die einander zugewandten Außenkanten zweier Floats (von denen eines links und eines rechts angeordnet wurde) dürfen sich nicht überschneiden.

- Die Oberkante eines Floats darf nicht über die Oberkante seines umgebenden Blocks hinausragen.
- Die Oberkante eines Floats darf nicht höher liegen als die Oberkante eines früheren Floats oder Block-Elements.
- Die Oberkante eines Floats darf nicht über die Oberkante einer vorangehenden Zeilen-Box mit Inhalt hinausragen.
- Steht links von einem links angeordneten Float ein weiterer Float, darf die rechte Außenkante des rechten Floats nicht über den Rand des umgebenden Elements hinausragen. Diese Regel gilt analog für Floats, die rechts angeordnet sind.
- Ein Float muss immer so weit oben wie möglich angeordnet werden.
- Ein links angeordneter Float muss so weit links wie möglich stehen. Das Gleiche gilt entsprechend für rechts angeordnete Floats. Eine höhere Position hat Vorrang gegenüber einer weiter rechts oder links befindlichen Position.

Positionierung

Wenn Elemente positioniert werden, kommt eine Reihe spezieller Regeln zur Anwendung. Diese Regeln steuern nicht nur den umgebenden Block des Elements, sondern legen auch dessen inneres Layout fest.

Arten der Positionierung

Statische Positionierung

Die Box für das Element wird angelegt wie üblich. Block-Elemente erzeugen eine rechteckige Box, die im Textfluss des Dokuments angeordnet wird. Inline-Boxen sorgen für die Erzeugung einer oder mehrerer Zeilenboxen, die im Fluss ihres Elternelements angeordnet werden.

Relative Positionierung

Die Elementbox wird um eine festgelegte Entfernung von ihrem ursprünglichen Ort verschoben. Als umgebender Block gilt der Bereich, den das Element eigentlich eingenommen hätte. Das