O'REILLY®

5. Auflage
Behandelt CSS3

CSS

kurz & gut

O'REILLYS TASCHENBIBLIOTHEK

Eric A. Meyer

Übersetzung von Jørgen W. Lang

5. AUFLAGE

CSS

kurz & gut

Eric A. Meyer

Deutsche Übersetzung
von Jørgen W. Lang

O'REILLY®

Eric A. Meyer

Lektorat: Ariane Hesse
Übersetzung: Jørgen W. Lang
Korrektorat: Sibylle Feldmann, www.richtiger-text.de
Herstellung: Stefanie Weidner
Umschlaggestaltung: Michael Oréal, www.oreal.de
Satz: III-satz, www.drei-satz.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

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.

ISBN:
Print 978-3-96009-091-5
PDF 978-3-96010-282-3
ePub 978-3-96010-283-0
mobi 978-3-96010-284-7

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«. O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

5. Auflage

Wieblinger Weg 17
69123 Heidelberg

Authorized German translation of the English edition of CSS Pocket Reference, 5th Edition, ISBN 978-1-492-03339-4 is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

5 4 3 2 1 0

Inhalt

Vorwort

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 wesentlichen Konzepte von CSS finden Sie in diesem Buch eine alphabetische Referenz aller CSS3-Selektoren, gefolgt von einer alphabetischen Liste der CSS3-Eigenschaften.

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.

<Kursiv> Kursiv in spitzen Klammern
: Kennzeichnet Text, der durch benutzerdefinierte oder durch den Kontext bestimmte Werte zu ersetzen ist.

`Nichtproportionalschrift`
: Wird verwendet, um den Inhalt von Dateien oder die Ausgabe von Befehlen darzustellen.

Weitere Konventionen beziehen sich auf die Wertesyntax und werden zu Beginn des vierten Kapitels erläutert.

Verwendung von Codebeispielen

Dieses Buch soll Ihnen bei Ihrer Arbeit helfen. Im Allgemeinen dürfen Sie die Codebeispiele aus diesem Buch in Ihren Programmen und Ihrer Dokumentation verwenden. Hierfür müssen Sie keine Genehmigung von uns einholen, es sei denn, Sie übernehmen einen wesentlichen Teil des Codes. Wenn Sie beispielsweise ein Programm schreiben, dass mehrere Codestücke verwendet, ist keine Genehmigung von uns nötig. Wenn Sie Beispiele aus O'Reilly-Büchern verkaufen oder kommerziell vertreiben, ist dagegen eine Genehmigung nötig. Wenn Sie eine Frage beantworten oder Beispielcode aus diesem Buch zitieren, ist ebenfalls keine Genehmigung nötig. Wenn Sie einen wesentlichen Teil des Beispielcodes in Ihrer Produktdokumentation verwenden, ist eine Genehmigung nötig.

Eine Quellenangabe ist erwünscht, aber nicht zwingend. Üblicherweise enthält eine Quellenangabe die Nennung von Titel, Autor, Verlag und ISBN. Wenn Sie der Meinung sind, dass Ihre Verwendung von Codebeispielen nicht dem fairen Gebrauch entspricht oder nicht durch die oben formulierte Genehmigung abgedeckt ist, kontaktieren Sie uns bitte unter *permissions@oreilly.com*.

KAPITEL 1

Grundkonzepte

HTML mit Stildefinitionen versehen

Dokumente können auf drei verschiedene Arten mit Stildefinitionen versehen werden. Wie das funktioniert, wird in den folgenden Abschnitten erklärt.

Inline-Stile

In HTML können Stildefinitionen für ein bestimmtes Element anhand des `style`-Attributs angegeben werden. Der Wert des `style`-Attributs ist ein *Deklarationsblock*, allerdings ohne die geschweiften Klammern (Details hierzu finden Sie im Abschnitt »Struktur einer CSS-Regel« auf Seite 5):

```
<p style="color: red; background: yellow;">Aufgepasst!
Dieser Text soll auffallen!</p>
```

Beachten Sie, dass beim Schreiben dieses Buchs als Wert des `style`-Attributs nur ein einzelner Deklarationsblock verwendet werden kann. Außerdem können `:hover`-Stile oder `@import`-Anweisungen in einem `style`-Attribut *nicht* verwendet werden.

Obwohl typische XML-basierte Dokumentsprachen (z. B. SVG) das `style`-Attribut unterstützen, ist es ziemlich unwahrscheinlich, dass *alle* XML-Sprachen in absehbarer Zeit ähnliche Fähigkeiten besitzen werden. Aus diesem Grund – und weil die schlechtere Wartbarkeit meistens schwerer wiegt als der praktische Nutzen – sollten Sie das `style`-Attribut und damit Inline-Stile nach Möglichkeit nicht benutzen.

Eingebettete Stylesheets

Anhand des `style`-Elements kann ein Stylesheet direkt in ein HTML-Dokument eingebettet werden:

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

XML-basierte Sprachen stellen möglicherweise ähnliche Fähigkeiten bereit. Um sicherzugehen, sollten Sie daher immer die *Document Type Definition* (DTD) überprüfen.

Wie oben zu sehen, werden `style`-Elemente oft im `head`-Element platziert, auch wenn das nicht zwingend nötig ist. Gelegentlich werden Stylesheets aus Performancegründen auch am Ende eines Dokuments eingebettet.

Externe Stylesheets

Stildefinitionen können in einer separaten Datei gespeichert werden. Der Hauptvorteil liegt hierbei darin, dass bei der Aktualisierung häufig verwendeter Stile nur ein Stylesheet verändert werden muss. Der Nachteil besteht darin, dass es normalerweise effizienter ist, alle Stile (und Skripte) in ein HTML-Dokument einzubetten, um die Anzahl der Serverabfragen klein zu halten. Mit der zunehmenden Verwendung von HTTP/2 wird dieser Nachteil allerdings immer geringer.

Ein externes Stylesheet kann auf drei verschiedene Arten eingebunden werden:

@import-Direktive

Eine oder mehr `@import`-Direktiven können zu Beginn eines Stylesheets angegeben werden. Für HTML-Dokumente geschieht das am Anfang eines beliebigen Stylesheets:

```
<head><title>Mein Dokument</title>
<style type="text/css">
@import url(site.css);
@import url(navbar.css);
@import url(footer.css) screen and (min-width: 960px);
body {background: yellow;}
</style>
</head>
```

Beachten Sie, dass `@import`-Direktiven (nach der Spezifikation ausschließlich) am Anfang eines beliebigen Stylesheets stehen können. Das heißt, ein Stylesheet kann ein weiteres einbinden, das seinerseits ein drittes einbindet und so weiter.

link-Element

In HTML-Dokumenten kann das `link`-Element verwendet werden, um ein Stylesheet mit einem Dokument zu verbinden. Mehrere `link`-Elemente sind erlaubt. Anhand des `media`-Attributs kann ein Stylesheet auf eine oder mehrere Medienumgebungen beschränkt werden:

```
<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 and (max-width: 960px)">
<link rel="stylesheet" type="text/css" href="paper.css"
  media="print and (color-depth: 2)">
</head>
```

Außerdem ist es möglich, Stylesheet-Alternativen anzugeben. Diese werden allerdings nur von wenigen Browsern unterstützt. Beim Schreiben dieses Buchs haben die allermeisten bekannten Benutzerprogramme alle eingebundenen Stylesheets geladen, unabhängig davon, ob der Benutzer diese tatsächlich braucht.

xml-stylesheet-Verarbeitungsanweisung

In XML-Dokumenten (z.B. XHTML-Dokumenten, die mit dem MIME-Type `text/xml`, `application/xml` oder `application/xhtml+xml` übertragen werden) kann eine `xml-stylesheet`-Verarbeitungsanwei-

sung verwendet werden, um ein Stylesheet mit dem Dokument zu verbinden. Hierfür muss der Prolog des XML-Dokuments eine `xml-stylesheet`-Verarbeitungsanweisung enthalten. Mehrfache `xml-stylesheet`-Anweisungen sind erlaubt. Das Pseudoattribut `media` kann verwendet werden, um ein 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"?>
```

HTTP-Link-Header

Die letzte (und bei Weitem seltenste) Methode, ein externes Stylesheet in Ihre Seiten einzubinden, besteht in der Verwendung eines HTTP-`Link`-Headers. Dieser Ansatz verwendet HTTP-Header, um die Effekte eines `link`-Elements oder einer `import`-Direktive nachzubilden.

Um das für die gesamte Website zu aktivieren, muss eine Zeile wie die folgende zur *.htaccess*-Datei im Wurzelverzeichnis des Webservers (Document Root) hinzugefügt werden. In diesem Beispiel steht `/style.css` für den Serverpfad zum zu ladenden Stylesheet:

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

Alternativ zur Verwendung von *.htaccess*, die zu Performanceproblemen führen kann, können Sie auch die Datei *httpd.conf* editieren, um das Gleiche zu erreichen:

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

Hierbei muss `/usr/local/username/httpdocs` durch den tatsächlichen Unix-Pfadnamen für das Wurzelverzeichnis Ihrer Website und `/style.css` durch den Speicherort des Stylesheets innerhalb dieses Wurzelverzeichnisses ersetzt werden.

Beim Schreiben dieses Buchs wurde die HTTP-Header-Methode nicht von allen Browsern unterstützt. Insbesondere Internet Explorer und Safari haben hier Probleme. Daher ist diese Technik normalerweise auf Produktionsumgebungen beschränkt, die auf anderen Benutzerprogrammen und gelegentlichen »Easter Eggs« für Firefox und Opera basieren.

Struktur einer CSS-Regel

Ein Stylesheet besteht aus einer oder mehr *Regeln*, die beschreiben, wie Seitenelemente dargestellt werden sollen. Jede Regel setzt sich aus zwei wesentlichen Bausteinen zusammen: dem *Selektor* und dem *Deklarationsblock*. Abbildung 1-1 illustriert die Struktur einer Regel.

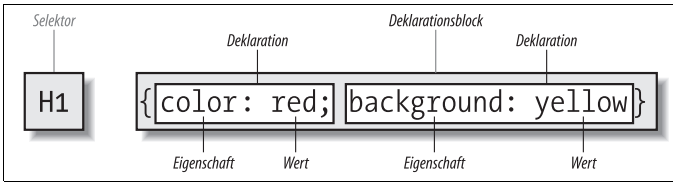

Abbildung 1-1: Struktur einer CSS-Regel

Auf der linken Seite der Regel befindet sich der Selektor, der die Teile des Dokuments auswählt, auf die die Regel angewandt werden soll. Selektoren können einzeln oder als kommagetrennte Liste angegeben werden. Um beispielsweise alle Überschriften der ersten drei Ebenen auszuwählen, würde die Selektorengruppe `h1, h2, h3` lauten. Auf der rechten Seite der Regel befindet sich der Deklarationsblock. Ein Deklarationsblock besteht aus einer oder mehreren *Deklarationen*; jede Deklaration besteht ihrerseits aus einer CSS-*Eigenschaft* und dem dazugehörigen *Wert*.

Der Deklarationsblock wird grundsätzlich mit geschweiften Klammern umgeben und kann mehrere Deklarationen enthalten. Jede Deklaration muss mit einem Semikolon (`;`) abgeschlossen werden. Die einzige Ausnahme bildet die letzte Deklaration eines Deklarati-

onsblocks. Hier ist das Semikolon optional (wird aber trotzdem empfohlen).

Jede Eigenschaft steht für einen bestimmten Darstellungsaspekt und wird durch einen Doppelpunkt (`:`) von ihrem Wert getrennt. Bei den Namen von Eigenschaften wird in CSS nicht zwischen Groß- und Kleinschreibung unterschieden. Erlaubte Werte für eine Eigenschaft werden durch die entsprechende Eigenschaftsbeschreibung definiert. Weitere Details zu erlaubten Werten für CSS-Eigenschaften finden Sie in Kapitel 4.

@-Regeln

Eine CSS-@-Regel (At-Regel) ist eine Anweisung oder ein Regelblock, der mit einem bestimmten Identifier beginnt, dem ein `@`-Zeichen vorangestellt wird. Dies sind:

`@charset`
: Erlaubt einem Autor, die Zeichenkodierung der Stile in einem Stylesheet festzulegen (z. B. `@charset "utf-8";`). Auf diese Weise können Autoren die Zeichenkodierung ihrer Stile auch dann definieren, wenn sie keine Kontrolle über die Kodierung der Datei oder des Systems haben, auf dem die Stile geschrieben werden. Werden mehrere `@charset`-Regeln deklariert, wird nur die erste verwendet. Die Regel *muss* in der ersten Zeile des Stylesheets stehen, in dem sie erscheint, ihr dürfen *keine* anderen Zeichen vorangestellt werden. Stylesheets, die in ein Dokument eingebettet werden, dürfen keine `@charset`-Regeln enthalten.

`@import`
: Erlaubt einem Autor, die Stile eines anderen Stylesheets einzubinden (siehe »@import-Direktive« auf Seite 2). Mehrfache `@import`-Regeln sind erlaubt. Sämtliche `@import`-Regeln *müssen* vor allen anderen Teilen des Stylesheets erscheinen mit Ausnahme von `@charset`.

`@namespace`
: Ermöglicht die Definition eines XML-Namensraums für die Verwendung in Selektoren (durch die Angabe von `@namespace`

`svg url(http://www.w3.org/2000/svg);` kann beispielsweise der Selektor `svg|a {color: black;}` benutzt werden, um `<a>`-Elemente in SVG-Dateien von `<a>`-Elementen in HTML-Dokumenten zu unterscheiden). Mehrfache `@namespace`-Regeln sind erlaubt. Jede `@namespace`-Regel *muss* vor allen anderen Teilen des Stylesheets stehen mit Ausnahme von `@charset`- und `@import`-Regeln.

Neben diesen Angaben gibt es eine Reihe bedingungsabhängiger `@`-Regeln. Hierzu gehören:

`@counter-style`
: Definiert Symbole und Zählmuster zur Verwendung in CSS-Zählern (»Counters«, z.B. die Nummerierung der Elemente einer geordneten Liste).

`@font-face`
: Definiert eine externe Schrift, die heruntergeladen und verwendet werden soll, inklusive der Identifier (Schriftnamen), die in anderen Stildefinitionen genutzt werden sollen. Dies ist Teil des oft als »Webfonts« oder »Custom Fonts« bezeichneten Systems.

`@keyframes`
: Definiert die Zustände verschiedener Schritte einer Animationssequenz, die unter einem einmaligen Identifier (Animationsnamen) gruppiert werden.

`@media`
: Definiert die Medientypen und Parameter, auf die ein bestimmter Block mit Stildefinitionen angewandt werden soll. Die Regel `@media (max-width: 600px)` legt beispielsweise fest, dass die Stile nur für Bildschirmbreiten unterhalb von 600 Pixeln gelten sollen. Dies ist der Schlüssel für »Responsive Web Design«.

`@supports`
: Legt fest, welche Merkmale ein Browser unterstützen muss, damit ein bestimmter Block mit Stildefinitionen angewandt wird. Die Formulierung `@supports (display: grid)` legt beispielsweise fest, dass die Stildefinitionen für einen Browser gelten sollen, der das CSS-Grid-System unterstützt.

Anfang 2018 existierte eine Reihe weiterer vorgeschlagener @-Regeln in unterschiedlichen Entwicklungsstadien. Hierzu gehören `@document`, `@font-feature-values`, `@page` und `@viewport`.

Kommentare

Das Einbetten von Kommentaren ist einfach. Sie werden durch die Zeichenfolge `/*` eingeleitet und mit `*/` wieder beendet, wie hier gezeigt:

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

Kommentare können sich über mehrere Zeilen erstrecken:

```
/* Dies ist ein Kommentar!
   Und hier geht der Kommentar weiter.
   Und hier auch. */
```

Kommentare können an beliebiger Stelle in einem Stylesheets eingesetzt werden, nur nicht innerhalb von Eigenschaftsnamen oder Werten:

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

HTML- (oder genauer SGML-)Kommentare `<!-- wie dieser -->` sind in Stylesheets erlaubt, um Stildefinitionen vor Browsern zu verstecken, die so alt sind, dass sie nicht einmal HTML 3.2 verstehen. Diese Kommentare funktionieren *nicht* als CSS-Kommentare. Das heißt, alles innerhalb der Kommentare ist für den CSS-Parser sichtbar und wird gegebenenfalls auch interpretiert.

Rangfolge der Stildefinitionen (Präzedenz)

Ein einzelnes HTML-Dokument kann mehrere externe Stylesheets einbinden und importieren, ein oder mehrere eingebettete Stylesheets enthalten sowie Inline-Stile verwenden. Dabei kann es leicht passieren, dass manche Regeln mit anderen in Konflikt geraten. CSS verwendet die sogenannte *Kaskade* (Cascade) – von der die Cascading Style Sheets übrigens ihren Namen haben –, um solche Konflikte zu lösen und schließlich einen Satz von Stildefinitionen auf das Dokument anzuwenden. Die zwei Hauptelemente der Kaskadierung sind Spezifität und Vererbung.

Berechnung der Spezifität

Die *Spezifität* beschreibt das »Gewicht« eines Selektors und der mit ihm verbundenen Deklarationen. Tabelle 1-1 zeigt, wie viel die einzelnen Teile eines Selektors zur Gesamtspezifität des Selektors beitragen.

Tabelle 1-1: Spezifität der verschiedenen Selektoren

Art des Selektors	Beispiel	Spezifität
Universal-Selektor Kombinator	`*` `+`	`0,0,0,0`
Element-Identifier Pseudoelement-Identifier	`div` `::first-line`	`0,0,0,1`

Klassen-Identifier
Pseudoklassen-Identifier
Attribut-Identifier
	`.warning`
	`:hover`
	`[type="checkbox"]`
		`0,0,1,0`

ID-Identifier	`#content`	`0,1,0,0`
Inline-style-Attribut	`style="color: red;"`	`1,0,0,0`

Die Spezifitätswerte sind kumulativ. Das heißt, ein Selektor, der zwei Element-Identifier und einen Klassen-Identifier enthält (z.B. `div.aside p`), hat die Spezifität `0,0,1,2`. Spezifitätswerte sind in einer Rangfolge von rechts nach links angeordnet, daher hat ein Selektor mit elf Element-Identifiern (`0,0,0,11`) eine niedrigere Spezifität als ein Selektor mit nur einem Klassen-Identifier (`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 eines Dokuments bilden eine baumartige Hierarchie, in der sich das Wurzelelement an der Spitze befindet und sich der Rest der Dokumentstruktur 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 wiederum diesen Elementen. In einer solchen Struktur bezeichnet man Elemente weiter unten im Baum als *Nachfahren* der Elemente, die sich weiter oben im Baum befinden.

CSS verwendet den Dokumentenbaum für das Verfahren der *Vererbung*, bei der ein Stil, der auf ein Element angewendet wird, von dessen Nachfahren übernommen (geerbt) wird. Hat das Element `body` z.B. den `color`-Wert `red`, wird dieser Wert den Dokumentbaum hinunter an die Elemente weitergereicht, die aus dem `body`-Element entspringen bzw. von ihm abstammen. Die Vererbung wird nur durch widersprüchliche Stilregeln unterbrochen, die sich direkt auf ein Element beziehen. Vererbte Werte haben überhaupt keine Spezifität (was *nicht* dasselbe ist wie eine Spezifität von null).

Beachten Sie, dass einige Eigenschaften nicht vererbt werden. Es ist für jede Eigenschaft separat definiert, ob sie vererbt wird. Einige Beispiele für nicht vererbbare Eigenschaften sind `padding`, `border`, `margin` und `background`.

Die Kaskade

Mithilfe der *Kaskade* werden in CSS Konflikte zwischen Stildefinitionen aufgelöst. Anhand dieses Verfahrens entscheidet ein Benutzerprogramm beispielsweise, welche Farbe ein Element erhält, falls zwei anwendbare Regeln versuchen, jeweils eine andere Farbe festzulegen. Die Kaskade funktioniert folgendermaßen:

1. Finde alle Deklarationen mit einem Selektor, der auf ein bestimmtes Element passt.
2. Sortiere alle Deklarationen, die auf ein bestimmtes Element anwendbar sind, nach explizitem Gewicht. Mit `!important` gekennzeichnete Regeln haben ein höheres explizites Gewicht als solche ohne diese Kennzeichnung.
3. Sortiere alle Deklarationen, die auf das jeweilige Element passen, nach *Herkunft*. Es gibt drei grundsätzliche Arten von Her-

kunft: Autor, Leser und Benutzerprogramm. Normalerweise gewinnen die Stildefinitionen des Autors gegenüber denjenigen des Lesers. `!important`-Stile des Lesers haben mehr Gewicht als alle anderen Stile, einschließlich der `!important`-Stile des Autors. Sowohl die Stildefinitionen des Autors als auch die des Lesers haben mehr Gewicht als die Standardstile des Browsers.

4. Sortiere alle Deklarationen, die auf das jeweilige Element anwendbar sind, nach ihrer *Spezifität*. Elemente mit einer höheren Spezifität haben größeres Gewicht als diejenigen mit einer niedrigeren.
5. Sortiere alle Deklarationen, die auf das jeweilige Element passen, nach ihrer *Reihenfolge*. Je später eine Deklaration im Stylesheet oder Dokument vorkommt, desto mehr Gewicht erhält sie. Deklarationen aus einem importierten Stylesheet werden behandelt, als seien sie vor allen Deklarationen des *importierenden* Stylesheets definiert worden.

Jegliche Darstellungshinweise, die nicht aus CSS-Quellen stammen (z. B. Browservoreinstellungen), erhalten dasselbe Gewicht wie die Standardstile des Browsers (siehe Schritt 2 oben).

Klassifizierung der Elemente

Einfach gesagt unterscheidet CSS zwischen zwei Arten von HTML-Elementen: *nicht ersetzte* und *ersetzte* Elemente. Auch wenn die Arten eher abstrakt erscheinen, gibt es grundlegende Unterschiede in ihrer Darstellung. Diese Unterschiede werden detailliert in Kapitel 7 von *CSS: The Definitive Guide*, 4th Edition (O'Reilly) behandelt.

Nicht ersetzte Elemente

Die meisten HTML-Elemente sind *nicht ersetzte* Elemente. Ihr Inhalt wird vom Benutzerprogramm in einer vom Element erzeugten Box dargestellt, beispielsweise: `<span>Moin</span>`. Der enthaltene Text `Moin` wird direkt vom Browser ausgegeben. Absätze, Überschriften, Tabellenzellen, Listen und fast alle anderen HTML-Elemente sind nicht ersetzte Elemente.

Ersetzte Elemente

Im Gegensatz dazu wird bei *ersetzten* Elementen der Inhalt durch etwas ausgetauscht, das nicht direkt im Dokument enthalten ist. Das bekannteste Beispiel ist das HTML-Element `img`. Anstelle des Elements wird eine Bilddatei angezeigt, die sich nicht im HTML-Dokument selbst befindet (es sei denn, sie wurde anhand eines `data:`-URI eingebunden). Tatsächlich hat das `img`-Element keinen eigenen Inhalt, sondern nur einen Namen und Attribute:

```
<img src="howdy.gif" alt="Moin">
```

Das Element kann nur dargestellt werden, indem der fehlende Inhalt durch etwas ersetzt wird, das auf andere Weise gefunden werden kann (in diesem Fall das im `src`-Attribut angegebene Bild). Ansonsten hätte das Element keine Darstellung. Ein anderes Beispiel ist das `input`-Element, das je nach Typ durch einen Radiobutton, ein Ankreuzfeld oder ein Texteingabefeld ersetzt wird. Ersetzte Elemente erzeugen bei der Ausgabe ebenfalls eine Box.

Darstellungsrollen

Zusätzlich zur Unterscheidung zwischen ersetzten und nicht ersetzten Elementen gibt es in CSS zwei grundsätzliche Darstellungsrollen: *Blockelemente* und *Inline-Elemente*. Alle `display`-Werte in CSS gehören zu einer dieser beiden Kategorien. Es kann wichtig sein, zu wissen, zu welcher der beiden grundsätzlichen Kategorien eine Box gehört, da sich einige Eigenschaften nur auf einen der beiden Typen beziehen.

Blockelemente

Blockelemente sind solche, bei denen sich die Box (standardmäßig) über die gesamte Breite des Inhaltsbereichs des Elternelements erstreckt. Links und rechts davon können keine anderen Elemente stehen. Blockelemente erzeugen sozusagen vor und nach der Elementbox einen »Zeilenumbruch«. Die bekanntesten HTML-Blockelemente sind `p` und `div`. Ersetzte Elemente können Blockelemente sein, sind es in der Regel aber nicht.

Einen Sonderfall der Blockelemente bilden Listeneinträge (`li`). Neben dem Verhalten von Blockelementen erzeugen Listeneinträge ein Aufzählungszeichen, typischerweise einen Punkt für ungeordnete (`ul`) und eine Zahl für geordnete Listen (`ol`), das der Elementbox üblicherweise vorangestellt wird. Abgesehen vom Aufzählungszeichen sind Listeneinträge mit anderen Blockelementen identisch.

Anfang 2018 erzeugten folgende `diplay`-Werte eine Blockbox: `block`, `list-item`, `table`, `table-row-group`, `table-header-group`, `table-footer-group`, `table-column-group`, `table-row`, `table-column`, `table-cell`, `table-caption`, `flex` und `grid`.

Inline-Elemente

Bei *Inline-Elementen* wird die Elementbox normalerweise auf einer Textzeile erzeugt, die den Textfluss nicht unterbricht. Das bekannteste Inline-Element ist vermutlich das HTML-Element `a`. Andere Beispiele sind `span` und `em`. Diese Elemente erzeugen vor und nach ihrer Box keinen Umbruch, sodass sie im Inhalt eines anderen Elements erscheinen können, ohne dessen Darstellung zu stören.

Auch wenn Block- und Inline-Elemente in CSS eine Menge mit den Block- und -Inline-Elementen in HTML gemeinsam haben, gibt es einen wichtigen Unterschied: In HTML können Blockelemente nicht in Inline-Elementen enthalten sein, während es in CSS keinerlei Beschränkung dahin gehend gibt, wie die Darstellungsrollen ineinander verschachtelt sind.

Folgende `display`-Werte erzeugen Inline-Boxen: `inline`, `inline-block`, `inline-table` und `ruby`. Beim Schreiben dieses Buchs war nicht ausdrücklich definiert, dass die verschiedenen Ruby-Werte (z. B. `ruby-text`) ebenfalls Inline-Boxen erzeugen. Allerdings ist das die wahrscheinlichste Entwicklung.

Grundlagen der visuellen Darstellung

CSS definiert Algorithmen für das Layout aller Elemente eines Dokuments. Sie bilden das Fundament der visuellen Darstellung in CSS. Es gibt zwei grundsätzliche Layoutarten: *Blocklayout* und *Inline-Layout*.

Blocklayout

Eine Blockbox erzeugt in CSS einen rechteckigen Bereich, der als *Elementbox* bezeichnet wird und beschreibt, wie viel Platz von einem Element beansprucht wird. Abbildung 1-2 zeigt die Bestandteile der Elementbox.

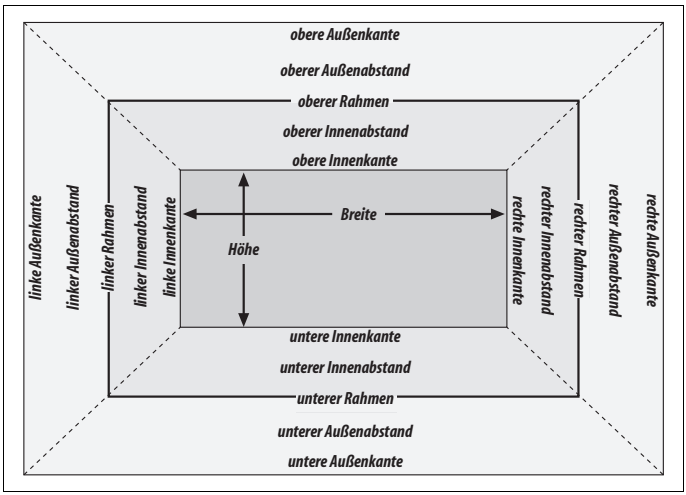

Abbildung 1-2: Das vollständige Boxmodell

Für eine Elementbox gelten die folgenden Regeln:

- Standardmäßig erstreckt sich der Hintergrund des Elements bis zur Außenkante des Rahmens und füllt damit die Bereiche von Inhalt, Innenabstand (`padding`) und Rahmen. (Dieses Verhalten kann mit der Eigenschaft `background-clip` verändert werden.) Besitzt der Rahmen transparente Bereiche (z. B. weil er gepunktet oder gestreift ist), ist der Hintergrund in diesen Bereichen sichtbar. Der Hintergrund erstreckt sich nicht in die Bereiche des Außenabstands (`margin`). Umrisse (`outline`), die im Bereich des Außenabstands gezeichnet werden, haben keinen Einfluss auf das Layout.

- Nur die Eigenschaften `margin`, `height` und `width` einer Elementbox dürfen den Wert `auto` erhalten.
- Nur Außenabstände (`margin`) können negative Werte erhalten.
- Der Standardwert für die Rahmenbreiten der Elementbox ist `0` (null). Der Standardwert für den Rahmenstil (`border-style`) ist `none`.
- Hat die Eigenschaft `box-sizing` den Wert `content-box` (Standardwert), definiert die Eigenschaft `width` nur die Breite des Inhaltsbereichs. Die Werte für Innen- und Außenabstände sowie Rahmen werden gegebenenfalls hinzuaddiert. Das Gleiche gilt für die Eigenschaft `height` in Bezug auf die Höhe.
- Hat `box-sizing` den Wert `padding-box`, definiert die Eigenschaft `width` die Gesamtbreite von Inhalt und Innenabstand. Rahmen und Außenabstände werden hinzuaddiert. Das Gleiche gilt `height` in Bezug auf die Höhe.
- Hat `box-sizing` den Wert `border-box`, definiert die Eigenschaft `width` die Gesamtbreite von Inhalt, Innenabstand und Rahmen. Außenabstände werden hinzuaddiert. Das Gleiche gilt für `height` in Bezug auf die Höhe.

Inline-Layout

Ein Inline-Element erzeugt in CSS eine oder mehrere rechteckige Bereiche, die als *Inline-Boxen* bezeichnet werden. Folgende Regeln gelten für Inline-Boxen:

- Die Eigenschaften `width` und `height` können für nicht ersetzte Inline-Boxen nicht verwendet werden.
- Für die Eigenschaften `left`, `right`, `top`, `bottom`, `margin-left`, `margin-right`, `margin-top` und `margin-bottom` wird der Wert `auto` in `0` (null) konvertiert.
- Für ersetzte Inline-Boxen gelten folgende Regeln:
  - Haben sowohl `height` als auch `width` den Wert `auto` und besitzt das Element eine intrinsische Breite (z.B. wenn ein Bild seine Breite selbst definiert), entspricht der Wert von

`width` der intrinsischen Breite des Elements. Das Gleiche gilt entsprechend für `height`.

- Haben sowohl `height` als auch `width` den Wert `auto` und besitzt das Element keine intrinsische Breite, aber eine intrinsische Höhe und ein Layoutseitenverhältnis, erhält `width` die intrinsische Höhe multipliziert mit dem Seitenverhältnis.
- Haben sowohl `height` als auch `width` den Wert `auto` und besitzt das Element keine intrinsische Höhe, aber eine intrinsische Breite und ein Layoutseitenverhältnis, erhält `height` die intrinsische Breite dividiert durch das Seitenverhältnis.

Es gibt ein paar Regeln, die sogar noch undurchsichtiger sind als die letzten zwei. Details hierzu finden Sie unter Dokumentation zum CSS Box-Modell (*http://w3.org/TR/css3-box/#inline-replaced*).

Alle Inline-Elemente besitzen einen Wert für `line-height` (Zeilenhöhe), der einen großen Einfluss auf die Darstellung der Elemente hat. Die Höhe einer Textzeile wird unter Berücksichtigung der folgenden Faktoren bestimmt:

Anonymer Text
: Eine beliebige Zeichenkette, die nicht in einem Inline-Element enthalten ist. Nehmen wir beispielsweise das folgende Markup:

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

Hier sind die Zeichenfolgen »Ich bin« und » glücklich!« anonymer Text. Übrigens: Das Leerzeichen vor »glücklich« ist ein Zeichen wie jedes andere und daher Teil des anonymen Texts.

Em-Box
: Der Raum, den der Großbuchstabe M des jeweiligen Zeichensatzes für sich beansprucht, wird auch als Zeichenbox bezeichnet. Die tatsächlichen Glyphen können größer oder kleiner als ihre Em-Box sein, wie in Kapitel 5 von *CSS: The Definitive Guide*, 4th Edition besprochen wird. In CSS wird die Größe der Em-Box durch die Eigenschaft `font-size` definiert.

Inhaltsbereich
: Für nicht ersetzte Elemente kann der Inhaltsbereich die Box sein, die durch die verbundenen Em-Boxen aller Zeichen eines Elements beschrieben wird. Ansonsten ist der Inhaltsbereich die Box, die von allen im Element enthaltenen Glyphen beschrieben wird. In CSS 2.1 und später können Benutzerprogramme selbst entscheiden, welches Verfahren sie verwenden. Der Einfachheit halber verwenden wir in diesem Buch die Em-Box-Definition. Bei ersetzten Elementen entspricht der Inhaltsbereich der intrinsischen Höhe des Elements zuzüglich möglicher Innen- und Außenabstände sowie Rahmen.

Durchschuss
: Der Durchschuss ist die Differenz zwischen den Werten für `font-size` und `line-height`. Jeweils die Hälfte dieses Unterschieds wird der oberen und unteren Hälfte des Inhaltsbereichs hinzugerechnet. Diese Erweiterungen des Inhaltsbereichs heißen – wenig überraschend – *Halb-Durchschuss*. Der Durchschuss wird nur auf nicht ersetzte Elemente angewandt.

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.

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