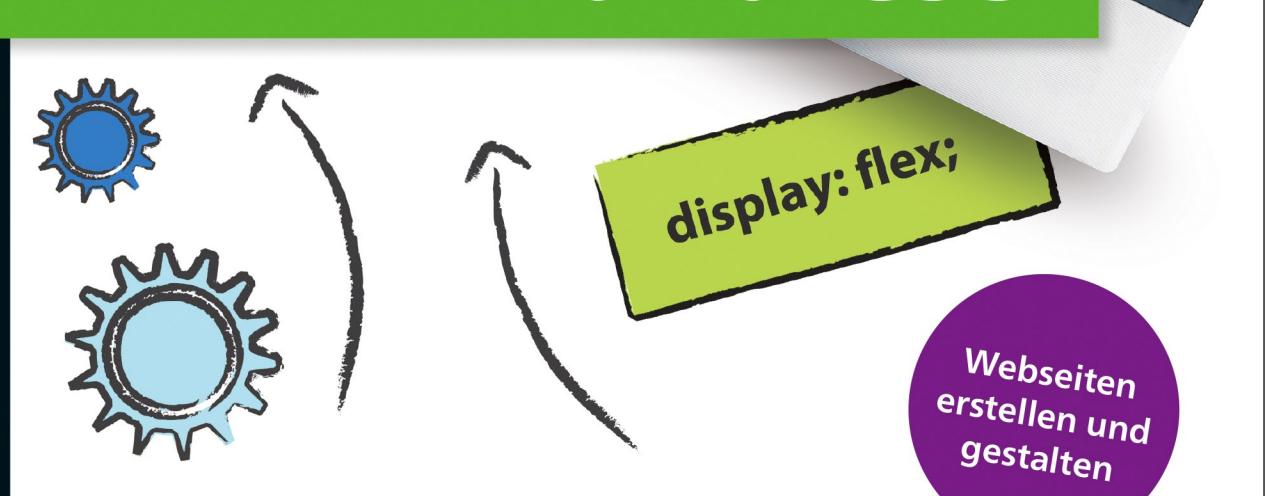




Einstieg in HTML und CSS



- ▶ Alle wichtigen HTML-Elemente und CSS-Eigenschaften
- ▶ Moderne Layouts mit Flexbox und CSS Grid
- ▶ Responsive Webseiten und mobile Navigation



Alle Beispielprojekte zum Download



Rheinwerk
Computing

Kapitel 2

HTML kennenlernen: Die erste Webseite erstellen

Worin Sie erfahren, dass alle Webseiten aus rechteckigen Kästchen bestehen, die mit HTML erstellt werden.

Die Themen im Überblick:

- ▶ Webseiten bestehen aus rechteckigen Kästchen, Seite 48
- ▶ HT-M-L: die »HyperText Markup Language«, Seite 49
- ▶ Jede Webseite hat ein HTML-Grundgerüst, Seite 50
- ▶ Der <!doctype> und das Stammelement <html>, Seite 54
- ▶ HTML-Elemente können Attribute haben, Seite 55
- ▶ Der <head> enthält wichtige Infos über die Webseite, Seite 56
- ▶ Bitte nicht verkleinern: <meta name="viewport" ...>, Seite 59
- ▶ Der sichtbare Bereich der Webseite steht in <body>, Seite 61
- ▶ Der Kopfbereich der Webseiten: <header>, Seite 63
- ▶ Entwicklerwerkzeuge im Browser: HTML analysieren, Seite 64
- ▶ Eine sehr kurze Geschichte von HTML, Seite 65
- ▶ Auf einen Blick, Seite 67

HTML ist eine vergleichsweise einfache Sprache und wird vielleicht gerade deshalb manchmal nicht wirklich ernst genommen, aber die Gestaltung von Webseiten beginnt mit soliden HTML-Kenntnissen.

In diesem Kapitel lernen Sie HTML kennen und erstellen die Startseite für eine kleine Übung Webseite, die im Laufe des Buches Schritt für Schritt weiterentwickelt wird.

2.1 Webseiten bestehen aus rechteckigen Kästchen

Webseiten bestehen aus rechteckigen Kästchen, die im Browserfenster übereinander, nebeneinander und ineinander gestapelt werden und auf Englisch *box* genannt werden.

Abbildung 2.1 zeigt die Startseite der Übungswebsite am Ende des Buches. Die rechteckigen Kästchen wurden mit einer Rahmenlinie sichtbar gemacht. Je eher Sie sich an den Gedanken gewöhnen, dass Webseiten aus Rechtecken bestehen, desto leichter wird Ihnen das Gestalten von Webseiten fallen. Everything is a box. Alles Runde ist entweder Trick, Grafik oder beides.

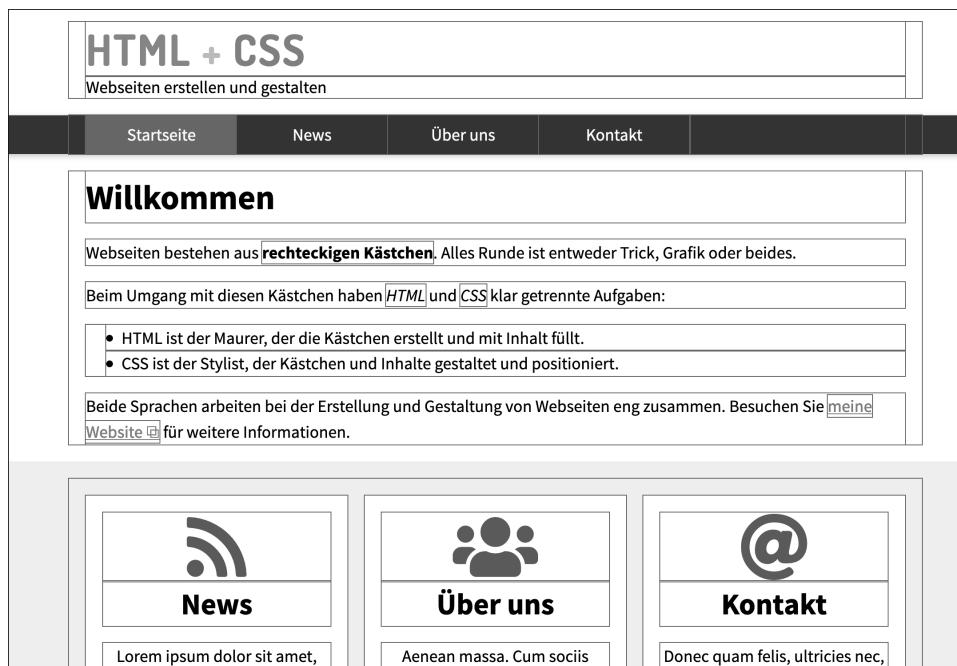


Abbildung 2.1 Die fertige Startseite mit sichtbar gemachten Kästchen

Beim Umgang mit diesen rechteckigen Kästchen haben HTML und CSS klar getrennte Aufgaben:

- ▶ HTML strukturiert die Webseite und erstellt die Kästchen.
- ▶ CSS gestaltet die Kästchen und deren Inhalte.

Das Zusammenspiel dieser beiden Sprachen ist Thema dieses Buches, und los geht es mit HTML. CSS kommt im nächsten Kapitel.

2.2 HT-M-L: die »HyperText Markup Language«

Die Abkürzung HTML steht für *HyperText Markup Language*, was übersetzt so viel heißt wie *Sprache zur Markierung von Hypertext*. Das ist zwar korrekt, aber nicht sehr aussagekräftig, und deshalb folgt hier eine etwas verständlichere Erklärung dieser vier Buchstaben.

2.2.1 HT wie Hypertext – Text mit Hyperlinks

Hypertext ist ganz normaler Text mit *Hyperlinks*. Das World Wide Web besteht aus Milliarden von Webseiten, die durch diese Links miteinander verbunden sind. Dadurch entsteht bildlich gesprochen ein weltweites, fein gesponnenes Gewebe von Webseiten, oder etwas prosaischer ausgedrückt:

Hyperlinks sind die Fäden, mit denen das World Wide Web gesponnen wird.

Hyperlinks sind also das Besondere am Web, und das *HT* im HTML besagt lediglich, dass man damit Hyperlinks erstellen kann.

2.2.2 M wie Markup – Text mit <tags>

Markup wird meist mit »Auszeichnung« übersetzt, und das können Sie sich wie in einem Supermarkt vorstellen: *Ware auszeichnen* bedeutet so viel wie *Etiketten an die Ware kleben*.

Typisch für HTML sind die in spitzen Klammern stehenden *Tags* (*tähgs* gesprochen), was auf Deutsch *Etikett* heißt. Mit diesen Etiketten zeichnen Sie den Text aus:

< p > Dieser Text ist ein Absatz. < /p >

Die Tags < p > und < /p > sagen dem Browser, dass der Text dazwischen ein ganz normaler Fließtextabsatz ist. *p* ist kurz für *paragraph*, auf Deutsch *Absatz*.

2.2.3 L wie Language – Vokabeln und Grammatikregeln

Das *L* steht für *Language*. HTML ist eine Sprache, und dementsprechend gibt es Vokabeln wie *Elemente*, *Tags* oder *Attribute* und Grammatikregeln zu deren Einsatz – das alles will gelernt und zum Teil sehr genau umgesetzt werden.

Das Wichtigste lernen Sie in diesem Buch, und für alles andere gibt es im Web mehr als genug Referenzen. Einige Links dazu finden Sie in Abschnitt 1.6, »Referenzen und Nachschlagewerke zu HTML und CSS«.

2.2.4 Der Unterschied zwischen »Element« und »Tag«

Da die Begriffe *Element* und *Tag* im Alltag oft Verwirrung stiften, möchte ich den Unterschied kurz erläutern.

Die Namen der HTML-Elemente sind Abkürzungen für einen englischen Begriff. Das Element für einen Absatz heißt wie gesehen schlicht und einfach *p*, kurz für *paragraph*.

Anfang und Ende eines HTML-Elements werden im Quelltext durch *Tags* markiert. Für einen Absatz lautet das Anfangs-Tag `<p>` und das Ende-Tag `</p>`. Abbildung 2.2 zeigt ein kleines Beispiel.

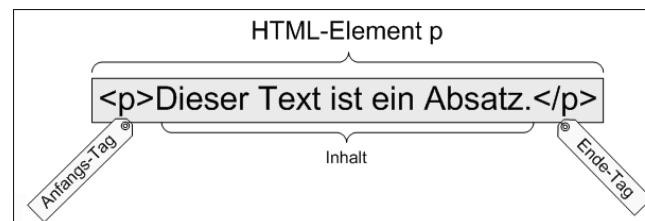


Abbildung 2.2 Ein Element besteht aus Anfangs-Tag, Inhalt und Ende-Tag.

Das HTML-Element *p* besteht also aus drei Teilen:

1. dem Anfangs-Tag `<p>`
2. dem Ende-Tag `</p>`
3. dem Inhalt zwischen den beiden Tags

Alle drei Teile zusammen bilden das *Element*. Die *Tags* stehen in spitzen Klammern, das *Element* hingegen heißt schlicht und einfach *p*, ohne spitze Klammern. Dieser Unterschied wird beim Gestalten per CSS wichtig, wenn es um das Selektieren von HTML-Elementen geht.

2.3 Jede Webseite hat ein HTML-Grundgerüst

In diesem Abschnitt erstellen Sie das HTML-Grundgerüst für die Startseite der Übungswebsite.

Drei Empfehlungen für Datei- und Ordnernamen im Web

Vorab drei Empfehlungen für die Namen der Ordner, HTML-Dateien, Stylesheets und Grafikdateien einer Website:

- ▶ Kleinschreibung
- ▶ keine Leerstellen
- ▶ keine Umlaute oder sonstige Sonderzeichen

Wenn Sie diese Empfehlungen beherzigen, ersparen Sie sich eine Menge möglicher Probleme.

2.3.1 Die Datei »index.html« im Editor erstellen und speichern

In diesem Abschnitt erstellen Sie einen Übungsordner und eine Datei namens *index.html*. Den Namen für den Ordner können Sie selbst wählen, der Name für die Startseite ist festgelegt:

- ▶ Eine Startseite hat den Vornamen *index*.
- ▶ HTML-Dateien haben den Familiennamen *.html*.

Im folgenden Kasten erstellen Sie eine Datei namens *index.html*.

Übungswebsite: Die Startseite »index.html« erstellen und speichern

1. Erstellen Sie irgendwo auf Ihrer Festplatte einen Übungsordner.
2. Starten Sie einen Editor, und erstellen Sie eine neue Datei.
3. Speichern Sie diese Datei im Übungsordner als *index.html*.
4. Fertig, und weiter geht's.

2.3.2 Eine gute Angewohnheit: <!-- Kommentare -->

Es ist eine gute Angewohnheit, den Quelltext von Anfang an mit Kommentaren zu versehen. Kommentare stehen im Quelltext, erscheinen aber nicht auf der Webseite im Browser. Kommentare haben zwei Funktionen:

- ▶ Dokumentieren. Als Gedächtnisstütze, damit Sie auch morgen noch wissen, was Sie sich heute dabei gedacht haben.
- ▶ Auskommentieren. Um Quelltextpassagen testweise vor dem Browser zu verstecken, ohne sie zu löschen.

In HTML beginnt ein Kommentar mit `<!--` (kleiner als, Ausrufezeichen und zwei Bindestriche) und endet mit `-->` (zwei Bindestriche und größer als):

```
<!-- Dieser Text ist ein HTML-Kommentar. -->
```

Listing 2.1 Ein HTML-Kommentar

Wenn der Browser die Zeichenfolge `<!--` sieht, weiß er, dass ein Kommentar anfängt und er den Text bis zum Kommentarenden `-->` nicht im Browserfenster darstellen soll.

HTML-Kommentare dürfen *nicht verschachtelt* werden. Innerhalb eines Kommentars darf also kein weiterer Kommentar stehen.

Kommentare bleiben im Quelltext sichtbar

Denken Sie beim Verfassen von Kommentaren daran, dass diese zwar nicht im Browserfenster erscheinen, aber doch im Quelltext stehen und dass jeder Besucher sich den Quelltext ansehen kann.

2.3.3 Das HTML-Grundgerüst für die Startseite erstellen

Der Quelltext einer jeden Webseite besteht aus vier Abschnitten:

1. Ganz am Anfang, in der allerersten Zeile, steht der `doctype`.
2. Das Stammelement `html` enthält nur die Elemente `head` und `body`.
3. Der Vorspann `head` hat Elemente wie `meta` und `title`.
4. `body` enthält den Inhalt der Webseite für die Besucher.

Diese vier Abschnitte bilden zusammen das HTML-Grundgerüst, das einer Webseite wie ein Skelett eine Struktur gibt und sie im Innersten zusammenhält.

Listing 2.2 zeigt den Quelltext für das Grundgerüst der Startseite auf einen Blick, wobei Sie zwischen `<body>` und `</body>` statt einem Absatz mit »Hallo!« auch gerne etwas anderes schreiben können:

```
<!doctype html>
<html lang="de">

  <head>
    <meta charset="utf-8">
    <title>Startseite - Einführung in HTML + CSS</title>
  </head>
```

```
<body>
  <p>Hallo!</p>
</body>
```

```
</html>
```

Listing 2.2 Das HTML-Grundgerüst für die Startseite

Im folgenden Kasten erstellen Sie das Grundgerüst für die Startseite der Übungswebsite.

Übungswebsite: Das HTML-Grundgerüst für die Startseite erstellen

1. Öffnen Sie die Datei `index.html` im Editor.
2. Erstellen Sie in der leeren Datei das in Listing 2.2 gezeigte HTML-Grundgerüst.
3. Speichern Sie die Datei im Editor.
4. Öffnen Sie die Startseite im Browser.

Abbildung 2.3 zeigt die Startseite nach diesen Schritten im Browser. Der Seitentitel erscheint oben im Browser-Tab, der Text zwischen `<body>` und `</body>` im inneren Bereich des Browserfensters.



Abbildung 2.3 Die Startseite mit `<title>` im Tab und `<p>` im Browserfenster

Sie werden den größten Teil Ihrer Zeit mit dem Erstellen und Gestalten von HTML-Elementen im `body` verbringen, aber zunächst möchte ich Ihnen die einzelnen Teile des Grundgerüsts kurz vorstellen.

Quelltext ordnen und übersichtlich schreiben

Für uns Menschen empfiehlt es sich, den Quelltext möglichst übersichtlich zu schreiben und zum Beispiel hierarchische Abhängigkeiten mit Einrückungen zu verdeutlichen. Unübersichtlich wird der Quelltext später ganz von allein.

Dem Browser hingegen ist das egal. Für ihn könnte der gesamte Quelltext in einer einzigen Zeile stehen. Sogenannten *Whitespace* (Leerstellen, Tabstopps und Zeilenumbrüche) ignoriert er. Ihn interessieren nur die in spitzen Klammern stehenden Tags.

2.4 Der `<!doctype>` und das Stammelement `<html>`

Das in Listing 2.2 gezeigte Grundgerüst beginnt mit dem doctype und dem Stammelement html. Los geht es mit dem doctype.

2.4.1 Die Dokumenttyp-Definition `<!doctype html>`

Die *Dokumenttyp-Definition*, kurz doctype, muss in der allerersten Zeile des Dokuments stehen:

```
<!doctype html>
```

Listing 2.3 Der »doctype« steht in der allerersten Zeile

Groß- und Kleinschreibung spielt für das Wort doctype keine Rolle. `<!DOCTYPE html>` ist also auch erlaubt.

Diese Zeile sagt dem Browser, dass es sich um ein Dokument vom Typ HTML handelt und dass der Quelltext mit einem Element namens html beginnt. Der doctype muss wie gesagt in der ersten Zeile des Quelltextes stehen. Er sorgt dafür, dass der Browser den Quelltext dem gültigen HTML-Standard gemäß umsetzt.

Früher war der »doctype« länger. Viel länger.

Falls Sie sich schon mal mit HTML beschäftigt haben, kommt Ihnen der doctype vielleicht sehr kurz vor. Früher war er viel länger und sah zum Beispiel so aus:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Heute reicht ein simples `<!doctype html>`. Die Browser verstehen das.

2.4.2 Das Stammelement: `<html>` und `</html>` umschließen den Quelltext

Mit html folgt das bereits im doctype angekündigte Stammelement, das die beiden Elemente head und body enthält:

```
<html lang="de">
  <head> ... </head>
  <body> ... </body>
</html>
```

Listing 2.4 Das Stammelement html enthält den »head« und den »body«.

Vor dem Anfangs-Tag `<html>` steht nur der doctype, nach dem Ende-Tag `</html>` kommt nichts mehr.

2.5 HTML-Elemente können Attribute haben

Das Anfangs-Tag des Stammelements enthält mit `lang="de"` eine Angabe, die bisher noch nicht erklärt wurde:

```
<html lang="de">
```

Listing 2.5 `<html>` mit dem Attribut lang und dem Wert »de«

Mit `lang="de"` definieren Sie die natürliche Sprache, in der der Inhalt der Webseite geschrieben ist. Diese Angabe ist für Maschinen bei der korrekten Verarbeitung des Textes sehr nützlich: Browsern hilft es bei der Silbentrennung, Screenreadern bei der korrekten Aussprache und Suchmaschinen bei der Analyse und Bewertung der Suchbegriffe.

Der Zusatz lang, kurz für *language*, ist ein sogenanntes *Attribut*, dem als Wert das Kürzel für die Sprache zugewiesen wird, in diesem Fall "de" für Deutsch.

Folgende Aufzählung enthält die wichtigsten Regeln zu Attributen auf einen Blick:

- ▶ Attribute stehen immer im *Anfangs-Tag*, das Ende-Tag ändert sich dadurch nicht.
- ▶ Fast alle Attribute haben einen *Wert*.
- ▶ Nach dem Namen des Attributs folgt *ohne* Leerzeichen ein Gleichheitszeichen und in Anführungsstrichen ein Wert: `lang="de"`.
- ▶ Zwischen Attributname, Gleichheitszeichen, Anführungsstrichen und Wert ist *wirklich kein Leerzeichen*.

- Wenn in einem Anfangs-Tag mehrere Attribute stehen, werden diese durch eine Leerstelle voneinander getrennt. Die Reihenfolge der Attribute spielt keine Rolle.

Mit dem Attribut »lang« kann man auch andere Elemente auszeichnen

lang ist ein globales Attribut und kann in fast allen HTML-Elementen verwendet werden. Auf einer überwiegend deutschsprachigen Webseite könnte man damit zum Beispiel einen englischen Absatz wie folgt markieren:

```
<p lang="en">This paragraph is in English.</p>
```

Das hilft zum Beispiel einem Screenreader bei der korrekten Aussprache.

Kapitel 3

CSS kennenlernen: Die erste Webseite gestalten

Worin Sie die bisher erstellten HTML-Elemente per CSS gestalten und sehen, wie man den Quelltext im Browser-Entwicklertool analysieren kann.

Die Themen im Überblick:

- ▶ Jeder Browser hat ein fest eingebautes Stylesheet, Seite 69
- ▶ Das HTML für <body> als schematische Darstellung, Seite 70
- ▶ Das erste eigene Stylesheet: »style.css«, Seite 72
- ▶ Die erste CSS-Regel: Hintergrundfarbe für <body>, Seite 73
- ▶ Den Kopfbereich <header> selektieren und gestalten, Seite 76
- ▶ Wichtige Vokabeln: Der Aufbau einer CSS-Regel, Seite 78
- ▶ Entwicklerwerkzeuge im Browser: CSS analysieren, Seite 79
- ▶ Eine sehr kurze Geschichte von CSS, Seite 80
- ▶ Auf einen Blick, Seite 81

CSS, kurz für *Cascading Style Sheets*, ist eine Sprache, die speziell zur Gestaltung von HTML-Elementen erfunden wurde. In diesem Kapitel erstellen Sie ein erstes Stylesheet, verbinden es mit der HTML-Datei und gestalten diese mit den ersten CSS-Regeln.

3.1 Jeder Browser hat ein fest eingebautes Stylesheet

Abbildung 3.1 zeigt die bisher noch recht übersichtliche Webseite *index.html* aus Kapitel 2, »HTML kennenlernen: Die erste Webseite erstellen«. Die von den HTML-Elementen *h1* und *p* erzeugten rechteckigen Kästchen wurden in der Abbildung mit einer Rahmenlinie sichtbar gemacht. Beachten Sie, dass die Kästchen sich über die gesamte Breite des Browserfensters erstrecken, von ganz links bis ganz rechts.



Abbildung 3.1 Der aktuelle Stand der Webseite

HTML-Elemente dienen zur Strukturierung von Webseiten und haben kein Aussehen. Wenn man sich die Startseite aber im Browser anschaut, ist diese durchaus bereits ein bisschen gestaltet:

- ▶ Die Seite hat einen weißen Hintergrund und schwarze Schrift.
- ▶ Die Überschrift ist fett und hat eine Schriftgröße von 32px.
- ▶ Der Absatz steht unterhalb der Überschrift, hat eine normale Strichstärke und eine Schriftgröße von 16px.
- ▶ Der Text hat eine *Serifen*-Schriftart mit kleinen Verzierungen an den Buchstaben.
- ▶ Beide Elemente haben nach oben und unten ein bisschen Abstand.

Verantwortlich für diese grundlegende Gestaltung ist eine Mischung aus den Browsereinstellungen und einem fest eingebauten Browser-Stylesheet. Wenn der Browser eine h1-Überschrift sieht, denkt er: »Mmmh, eine wichtige Überschrift. Hier steht nirgendwo, wie das aussehen soll, also mach ich den Text mal groß und fett.« Durch das Browser-Stylesheet ist der HTML-Quelltext im Browserfenster ohne weitere Gestaltung einigermaßen lesbar.

Von Ihnen selbst geschriebene CSS-Regeln überschreiben das Browser-Stylesheet, aber für alle Elemente und Eigenschaften, die Sie nicht gestalten, gelten weiterhin die Browsereinstellungen und das CSS aus dem Browser-Stylesheet. In Abschnitt 4.7, »Ein kurzer Blick auf das Browser-Stylesheet«, lernen Sie es noch genauer kennen.

3.2 Das HTML für <body> als schematische Darstellung

»Kenne dein HTML« ist das oberste Motto beim Gestalten von Webseiten, denn ohne die HTML-Struktur zu kennen, kann man im CSS nicht viel machen.

Zur Erinnerung daher ein kurzer Blick auf das HTML für body auf *index.html* (Listing 3.1):

```
<body>
  <header>
    <h1>HTML + CSS</h1>
    <p>Webseiten erstellen und gestalten</p>
  </header>
  <nav> </nav>
  <main> </main>
  <footer> </footer>
</body>
```

Listing 3.1 Die aktuelle HTML-Struktur

HTML-Elemente werden am Bildschirm als ineinander verschachtelte rechteckige Kästchen dargestellt, und besonders am Anfang ist es manchmal hilfreich, sich die Struktur des Quelltextes mit einer einfachen Zeichnung zu visualisieren. Schematisch dargestellt sieht dieser Quelltext so aus wie in Abbildung 3.2.

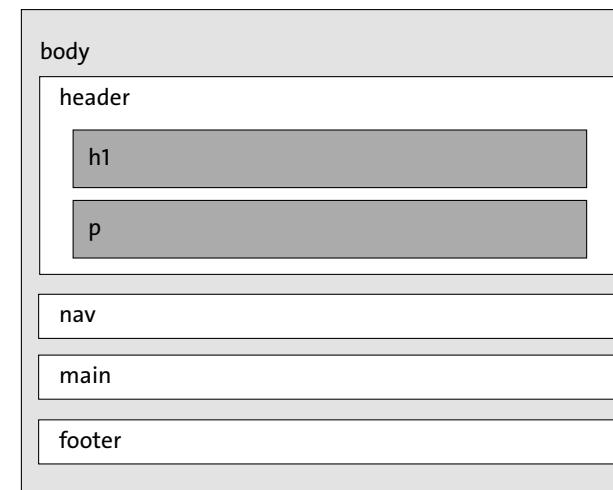


Abbildung 3.2 Schematische Darstellung des Quelltextes der Startseite

Jede von den HTML-Elementen generierte Box hat diverse Eigenschaften wie zum Beispiel die Hintergrund- oder die Schriftfarbe, und diese Eigenschaften können Sie per CSS gestalten.

Die Namen der CSS-Eigenschaften sind englische Begriffe in amerikanischer Schreibweise. So heißt die Eigenschaft zur Gestaltung der Hintergrundfarbe `background-color` und nicht `background-colour`.

In diesem Kapitel gestalten Sie die beiden Kästchen für `body` und `header` inklusive Überschrift und Absatz. Die Bereiche `nav`, `main` und `footer` enthalten noch keinen Inhalt und sind daher unsichtbar, aber das wird sich im Laufe der nächsten Kapitel ändern.

3.3 Das erste eigene Stylesheet: »style.css«

In diesem Abschnitt erstellen Sie im Übungsordner ein leeres Stylesheet und verbinden es dann mit der Beispieleseite `index.html`.

3.3.1 Schritt 1: Einen Unterordner und ein Stylesheet erstellen

Im ersten Schritt erstellen Sie einen Unterordner namens `css` und speichern darin ein Stylesheet `style.css`.

Übungswebsite: Ein Stylesheet und einen CSS-Kommentar erstellen

1. Wechseln Sie im Finder oder Explorer in den Übungsordner, in dem Sie die Startseite `index.html` gespeichert haben.
2. Erstellen Sie einen Unterordner namens `css`.
3. Erstellen Sie in Ihrem Editor eine leere Datei.
4. Speichern Sie die Datei als `style.css` im Unterordner `css`.

Der Dateiname `style.css` ist für ein Stylesheet weit verbreitet, aber nicht vorgeschrieben. Sie könnten also auch einen anderen Dateinamen wählen, solange er die Endung `.css` hat und den üblichen Empfehlungen für Dateinamen auf Webseiten entspricht (Kleinschreibung, keine Leerstellen, keine Sonderzeichen).

3.3.2 Schritt 2: HTML-Datei und CSS-Datei verbinden mit <link>

In diesem Abschnitt fügen Sie im `head` der Webseite ein HTML-Element mit dem Namen `link` ein, das dem Browser sagt, wo er die CSS-Datei findet, und Webseite und Stylesheet so miteinander verbindet:

```
<head>
  <meta charset="utf-8">
```

```
<title>Startseite - Einstieg in HTML + CSS</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
```

```
<link href="css/style.css" rel="stylesheet">
```

```
</head>
```

Listing 3.2 Das Element »link« verbindet HTML und CSS

Die beiden Attribute im `link`-Element haben folgende Bedeutung:

- `href` gibt den Pfad zu einer Datei an, und `style.css` liegt im Unterordner `css`.
- `rel` ist kurz für *relation (Beziehung)*. `rel="stylesheet"` bedeutet: »Die verknüpfte Datei ist ein Stylesheet.«

Durch diese Anweisung weiß der Browser, wo er die Gestaltungsanweisungen für die im Quelltext stehenden HTML-Elemente finden kann. Im folgenden Kasten fügen Sie auf der Startseite der Übungswebsite ein `link`-Element hinzu.

Übungswebsite: Startseite und Stylesheet per »link« verbinden

1. Öffnen Sie die Startseite `index.html` in Ihrem Editor.
2. Lassen Sie die vorhandenen Elemente am Anfang des Dokuments unverändert, und fügen Sie vor `</head>` ein paar leere Zeilen ein.
3. Verknüpfen Sie wie in Listing 3.2 gezeigt die Datei `index.html` per `link`-Element mit dem Stylesheet `style.css`.
4. Speichern Sie die Seite, und betrachten Sie sie in einem Browser.

Im Browserfenster hat sich nach diesen Schritten nichts geändert, aber es gibt jetzt eine Verbindung zwischen der Webseite `index.html` und dem Stylesheet `style.css`.

3.4 Die erste CSS-Regel: Hintergrundfarbe für <body>

In diesem Abschnitt definieren Sie ein paar Farben für die Webseite, aber bevor es damit losgeht, noch eine kurze Unterbrechung mit einem Werbespot für CSS-Kommentare.

3.4.1 Auch in CSS eine gute Angewohnheit: /* Kommentare */

Genau wie in HTML sind Kommentare auch in CSS eine mehr als gute Angewohnheit. Sie können sie sowohl für eigene Notizen als auch zum vorübergehenden Auskommern gebrauchen. CSS-Kommentare sehen anders aus als HTML-Kommentare und stehen zwischen /* und */:

```
/* Stylesheet für die Übungswebsite aus "Einstieg in HTML + CSS" */
```

Listing 3.3 Ein Kommentar in CSS

Den Schrägstrich und das Sternchen erhalten Sie auf der Tastatur mit **Shift** + **/** bzw. **Shift** + *****. Auf einem Ziffernblock geht es noch einfacher:

- Der Schrägstrich **/** ist das Symbol für »geteilt durch« (Division).
- Das Sternchen ***** ist das Malzeichen (Multiplikation) daneben.

CSS-Kommentare dürfen wie HTML-Kommentare *nicht verschachtelt* werden. Innerhalb eines Kommentars darf also kein weiterer Kommentar stehen.

Übungswebsite: Einen Kommentar in »style.css« speichern

1. Öffnen Sie das Stylesheet *style.css* im Editor.
2. Erstellen Sie am Anfang der Datei wie in Listing 3.3 gezeigt einen CSS-Kommentar.
Der Text ist beliebig.
3. Speichern Sie die Datei.

3.4.2 Hintergrund- und Schriftfarbe für <body> ändern

Vor der Gestaltung der Eigenschaften müssen Sie dem Browser sagen, welches Element Sie gestalten möchten, und dazu schreiben Sie einfach dessen Namen hin. Die sichtbare Seite wird mit <body> und </body> begrenzt, der Name des Elements ist also *body*. Ohne spitze Klammern. Als Farbwert nutzen Sie in diesem Kapitel zunächst standardisierte englische Farbnamen. In diesem Abschnitt soll der Hintergrund der Startseite eine andere Farbe bekommen, und in CSS sieht das so aus:

```
body {
  background-color: floralwhite;
  color: black;
}
```

Listing 3.4 Hintergrund- und Schriftfarbe für die ganze Seite

Die CSS-Regel aus Listing 3.4 besteht aus folgenden Einzelteilen:

- *body* selektiert das zu gestaltende HTML-Element.
- Die Hintergrundfarbe gestalten Sie mit der CSS-Eigenschaft *background-color*, und als Farbwert wird *floralwhite* (Blütenweiß) verwendet.
- Die Eigenschaft für die Schriftfarbe heißt *color*, und *schwarz* wird die Schrift mit *black*.

Die Reihenfolge der Anweisungen zwischen den geschweiften Klammern spielt übrigens keine Rolle. Sie könnten also auch zuerst die Schrift- und dann die Hintergrundfarbe definieren. Im folgenden Kasten speichern Sie diese CSS-Regel in *style.css*.

Übungswebsite: Schrift- und Hintergrundfarbe für <body>

1. Öffnen Sie das Stylesheet *style.css* im Editor.
2. Erstellen Sie nun unter dem Kommentar am Anfang der Datei die Anweisungen in Listing 3.4.
3. Speichern Sie das Stylesheet, und betrachten Sie die Webseite *index.html* (nicht das Stylesheet) in einem Browser.

Abbildung 3.3 zeigt, dass sich die Hintergrundfarbe im Browserfenster nach diesen Schritten geändert hat.

HTML + CSS

Webseiten erstellen und gestalten

Abbildung 3.3 Die Seite mit einer hellen Hintergrundfarbe für <body>

Farben definieren in CSS

In CSS gibt es noch diverse andere Möglichkeiten zur Definition von Farbwerten, die Sie später in Kapitel 12, »Gestalten per CSS: Farben und Einheiten«, kennenlernen, aber Farbnamen sind erst einmal am einfachsten.

Kapitel 6

HTML-Elemente für Bilder, Audio und Video

Worin Sie HTML-Elemente zur Einbindung von Bildern, Audiodateien und Videos kennenlernen und diese mit ein paar CSS-Regeln flexibel machen.

6

Die Themen im Überblick:

- ▶ Über Grafikformate im Web: JPEG, GIF, PNG und SVG, Seite 121
- ▶ Ein Bild als Logo einbinden mit ``, Seite 123
- ▶ Bilder mit flexibler Breite: »max-width: 100%«, Seite 127
- ▶ Abbildungen beschriften: `<figure>` und `<figcaption>`, Seite 130
- ▶ Audiodateien einbinden mit `<audio>`, Seite 133
- ▶ Bewegte Bilder einbinden mit `<video>`, Seite 135
- ▶ Auf einen Blick, Seite 138

In diesem Kapitel sehen Sie, wie man in HTML Bilder auf Webseiten einfügt und für hochauflösende Bildschirme und unterschiedlich breite Viewports optimiert. Danach lernen Sie die HTML-Elemente zum Einfügen von Audio- und Videodateien auf Webseiten kennen.

6.1 Über Grafikformate im Web: JPEG, GIF, PNG und SVG

Ein Bild sagt nicht nur mehr als tausend Worte, im Web lädt es oft auch länger, denn jedes Kilobyte muss vom Server zum Besucher übertragen werden. Die Dateigröße von Bildern ist im Web also wichtig, und Formate mit Dateikompression wie JPEG, GIF und PNG sind daher am besten geeignet, im Gegensatz zu nicht komprimierten Dateiformaten wie BMP oder TIFF.

Die folgende Aufzählung fasst das Wichtigste zusammen:

- ▶ *JPEG* hat die Endung `.jpg` oder `.jpeg` und ist das Standardformat für Fotos. JPEG ist immer ein Kompromiss zwischen der Qualität des Bildes und der Größe der Datei.

Beim Speichern von JPEG-Dateien kann man die gewünschte Qualitätsstufe einstellen, und die besten Ergebnisse liefert eine Kompressionsrate von 60 bis 80 %.

- *GIF* hat die Endung *.gif*, ist der Veteran unter den Grafikformaten und wird manchmal noch für Logos verwendet. GIF kann nur 256 Farben darstellen, aber eine davon kann man als transparent, d. h. als durchsichtig festlegen. Beliebt ist GIF durch die Möglichkeit, mehrere Bilder in einer Datei zu speichern und nacheinander darzustellen. Diese GIF-Animationen findet man aber eher in Social Media als auf Webseiten.
- *PNG* hat die Endung *.png* und kommt in zwei Varianten: *PNG8* kann wie GIF nur 256 Farben speichern, *PNG24* hingegen wie JPEG über 16 Millionen. PNG bietet die Möglichkeit, Bereiche transparent erscheinen zu lassen. Das ist zum Beispiel ideal für die Darstellung von Fotos mit Freistellungen vor einem Hintergrund. Für normale Fotos ist JPEG oft die bessere Wahl, da es effektiver komprimiert.

Alle diese Formate sind *Rastergrafiken*, bei denen in der Datei nur Pixel gespeichert werden. Rastergrafiken kann man schlecht vergrößern, da dann die Pixel sichtbar und die Bilder unscharf werden.

Eine Besonderheit ist das Format *SVG (Scalable Vector Graphics)*, das wie der Name andeutet *Vektorgrafiken* bereitstellt. SVG-Dateien enthalten mathematisch berechnete Formen und Pfade (*Vektoren*), die erst zur Darstellung am Bildschirm in Pixel umgerechnet werden. Als Vektorformat kann man SVG sehr gut vergrößern oder verkleinern (*skalieren*), und die Bilder bleiben auch auf modernen, hochauflösenden Bildschirmen gestochen scharf. In Abschnitt 25.1, »Flexible Icons: Skalierbare Symbole mit SVG«, lernen Sie das Format näher kennen.

Tabelle 6.1 zeigt die wichtigsten Anwendungsgebiete dieser Dateiformate auf einen Blick.

Anwendungsgebiet	Bildformat
Normale Fotos	JPEG
Logos und Icons	PNG, GIF oder SVG
Bilder und Fotos mit Transparenz	GIF oder PNG

Tabelle 6.1 Anwendungsgebiete für Bildformate auf einen Blick

Im Zweifelsfall speichern Sie ein Bild einfach in mehreren Varianten und wählen dann die mit dem besten Kompromiss zwischen Bildqualität und Dateigröße.

6.2 Ein Bild als Logo einbinden mit ``

Das Element zum Einfügen einer Bilddatei auf Webseiten heißt `img`, kurz für *image (Bild)*, und in diesem Abschnitt ersetzen Sie den Text für die `h1`-Überschrift mit einem Logo.

6.2.1 Das Element `` und seine wichtigsten Attribute

`img` hat kein Ende-Tag, aber diverse Attribute, die Informationen über die Bilddatei enthalten.

Das folgende Listing zeigt ein Beispiel:

```

```

Listing 6.1 Das Element `` und einige Attribute

Wichtig zu verstehen ist zunächst einmal, dass der Browser die Bilddatei noch nicht hat, wenn er den Quelltext analysiert:

- Um das Bild darstellen zu können, muss er die angegebene Datei erst einmal vom Webserver holen.
- Bis zum Eintreffen der Bilddatei hat der Browser nur die Informationen, die in den Attributen von `img` stehen.

Der Quelltext zum Einbinden eines Bildes ist also wichtig, auch wenn er später im Browserfenster nicht erscheint, und die wichtigsten Attribute sind `src`, `alt`, `width` und `height`:

- `src="bilddatei.jpg"`

Das erste und wichtigste Attribut ist `src`, was für *Source* steht und *Quelle* heißt. Das Attribut enthält den Namen der Bilddatei und die Wegbeschreibung dorthin. Steht dort nur ein Dateiname, liegt die Datei im selben Ordner wie die Webseite.

- `alt="Alternativer Text"`

Die Eingabe eines *alternativen* Textes ist Pflicht. Dieser Text wird im Browserfenster angezeigt, wenn das Bild *nicht* oder *noch nicht* dargestellt werden kann. Möchten Sie aus irgendeinem Grund keinen alternativen Text angeben, schreiben Sie einfach `alt=""`.

- `width=""` und `height=""`

Die Attribute `width` und `height` teilen dem Browser die Abmessungen der Bilddatei mit, also wie viele Pixel sie breit bzw. hoch ist. So kann der Browser beim Erstellen der

Webseite den Platz für das Bild schon einplanen, *bevor* er die Bilddatei tatsächlich erhalten hat.

Das Element `img` erzeugt im Browserfenster keinen Zeilenumbruch, sondern fließt wie ein Inline-Element einfach in der Zeile mit.

6.2.2 Ein Logo auf der Übungswebsite einfügen mit

In diesem Abschnitt ergänzen Sie die Übungsseite um ein einfaches Logo, das in der `h1`-Überschrift anstelle des Textes »HTML + CSS« eingebunden wird. Das Logo ist eine transparente PNG-Datei, und Sie finden sie in den Übungsdateien.

Listing 6.2 zeigt den Quelltext zum Einbinden der Bilddatei. Die Attribute zu `img` stehen der Übersichtlichkeit halber jeweils in einer eigenen Zeile untereinander. Der Quelltext wird dadurch leichter lesbar, aber im Editor können Sie auch einfach alles in einer Zeile schreiben:

```
<h1>
  
</h1>
```

Listing 6.2 Ein Bild als Logo in einer Überschrift

Im folgenden Kasten setzen Sie dieses Listing für die Übungswebsite um.

Übungswebsite: Ein Bild als Logo einfügen

1. Wechseln Sie im Finder oder Explorer in den Übungsordner, in dem Sie die Startseite `index.html` gespeichert haben.
2. Erstellen Sie einen Unterordner namens `bilder`.
3. Kopieren Sie die Datei `html+css-logo.png` aus den heruntergeladenen Übungsdateien in den Ordner `bilder`.
4. Entfernen Sie den Text »HTML + CSS« aus der `h1`-Überschrift.
5. Binden Sie zwischen `<h1>` und `</h1>` wie in Listing 6.2 gezeigt das Logo ein. Das `img`-Element kann dabei auch in einer Zeile stehen.
6. Speichern Sie die Seite, und betrachten Sie sie in einem Browser.

Die Beispieleseite sieht jetzt im Browser ungefähr so aus wie in Abbildung 6.1.

HTML + CSS

Webseiten erstellen und gestalten

Startseite News Über uns Kontakt

Abbildung 6.1 Die Beispieleseite mit einem Bild als Logo

6.2.3 Hochauflösende Bildschirme benötigen größere Bilder

Vor einigen Jahren wäre das Einbinden von Bildern an dieser Stelle erledigt, aber inzwischen ist die Sache nicht mehr ganz so einfach wie früher. Alle modernen Smartphones und Tablets und immer mehr Laptops und Desktop-Computer haben sogenannte *hochauflösende Bildschirme*. Apple nennt sie *Retina*, bei anderen Herstellern haben sie andere Bezeichnungen.

Die Pixel auf diesen Bildschirmen sind kleiner als auf traditionellen Displays, und die *Pixeldichte* ist so hoch, dass das menschliche Auge die einzelnen Bildpunkte aus einem normalen Betrachtungsabstand nicht mehr unterscheiden kann.

Damit Schrift und Bilder durch die kleineren Pixel nicht schrumpfen, haben sich die Entwickler einen Trick ausgedacht und mehrere *Gerätepixel* zu logischen Pixeln zusammengefasst. Fazit: Auf traditionellen Bildschirmen ist ein Pixel auch wirklich ein Pixel, auf hochauflösenden Displays nicht.

Während vektorbasierte Schrift auf diesen Bildschirmen gestochen scharf wirkt, gibt es bei Bildern in Rasterformaten wie JPEG, PNG oder GIF ein Problem. Damit die Bilder nicht kleiner dargestellt werden, vergrößert der Browser die vorhandenen Pixel, wodurch die Bilder unscharf wirken.

Abbildung 6.2 zeigt, dass das im vorherigen Abschnitt eingebundene Logo auf einem hochauflösenden Bildschirm etwas verschwommen wirkt. Zum Vergleich sehen Sie darunter, wie es aussehen sollte.



HTML + CSS
HTML + CSS

Abbildung 6.2 Das Logo – oben etwas verschwommen, unten scharf

In Abschnitt 25.3, »Unterschiedliche Bilder je nach Pixeldichte«, erfahren Sie, wie man unterschiedliche Bilddateien ausliefern kann, in diesem Abschnitt nutzen Sie einen simplen, aber effektiven Trick und verwenden einfach eine doppelt so große Grafik: Damit das im Browserfenster 222 Pixel breite Logo auch auf hochauflösenden Bildschirmen scharf bleibt, binden Sie im HTML eine 444 Pixel breite Grafikdatei ein.

Im folgenden Kasten setzen Sie diese Lösung für die Übungswebsite um, und danach ist das Logo auch auf hochauflösenden Bildschirmen scharf.

Übungswebsite: Ein Logo für hochauflösende Bildschirme einbinden

1. Kopieren Sie die Datei `html+css-logo-444.png` in den Ordner `bilder` unterhalb des Übungsordners mit der Startseite `index.html`.
2. Suchen Sie im Quelltext das Element `img` zur Einbindung des Logos.
3. Ändern Sie das Attribut `src` so, dass die Datei `html+css-logo-444.png` eingebunden wird.
4. Speichern Sie die Seite, und betrachten Sie sie in einem Browser.

6.2.4 Fine-Tuning: Die Abstände um Logo und Slogan anpassen

Der Abstand zwischen dem Logo in der `h1`-Überschrift und dem Absatz ist auf der Beispieleite eigentlich etwas zu groß. In diesem Abschnitt zeige ich Ihnen, wie Sie diese vom Browser-Stylesheet vorgegebenen Abstände mit eigenem CSS überschreiben. Listing 6.3 enthält dazu zwei einfache Regeln:

```
header h1 {
  margin-bottom: 0;
}
header p {
  margin-top: 0;
}
```

Listing 6.3 Außenabstände von Logo und Slogan anpassen

Die erste Regel selektiert die `h1`-Überschrift und entfernt mit der Eigenschaft `margin-bottom` den Außenabstand nach unten, die zweite wählt den Absatz darunter aus und entfernt mit der Eigenschaft `margin-top` den Außenabstand nach oben. Im Folgenden binden Sie die Regeln aus Listing 6.3 auf der Übungswebsite ein.

Übungswebsite: Außenabstände von Logo und Slogan anpassen

1. Öffnen Sie das Stylesheet `style.css` im Editor.
2. Ergänzen Sie am Ende des Stylesheets die CSS-Regeln aus Listing 6.3.
3. Speichern Sie das Stylesheet, und betrachten Sie die Webseite in einem Browser.

Abbildung 6.3 zeigt, dass der Abstand zwischen Logo, Slogan und Navigation sich geändert hat, und der Kopfbereich sieht jetzt etwas kompakter aus. Mehr zu Eigenschaften wie `margin` erfahren Sie in Kapitel 16, »Das Box-Modell für Block- und Inline-Boxen«.

HTML + CSS

Webseiten erstellen und gestalten

Startseite News Über uns Kontakt

Willkommen

Abbildung 6.3 Logo und Slogan mit angepassten Außenabständen

6.3 Bilder mit flexibler Breite: »max-width: 100%«

In diesem Abschnitt sehen Sie, wie Sie Bilder mit einer einfachen CSS-Regel dazu überreden, nicht breiter zu werden als das umgebende HTML-Element. In den Übungsdateien finden Sie dazu im Ordner zu diesem Kapitel die Datei `bilder-flexibel-einbinden.html`.

6.3.1 Das Problem: Pixelbilder haben eine feste Breite

Abbildung 6.4 zeigt ein ganz normales Foto im Browser. Dieses Bild wird mit dem folgenden HTML eingebunden:

```

```

Listing 6.4 Ein Foto mit »img« einbinden

Solange das Browserfenster breiter ist als das Bild, sieht alles okay aus, aber in einem schmalen Viewport passt sich das Bild nicht an. Stattdessen wird es rechts abgeschnitten, sodass es nicht ganz zu sehen ist (Abbildung 6.5).



Abbildung 6.4 Ein ganz normales Foto im Browser



Abbildung 6.5 In einem schmalen Viewport wird das Foto rechts abgeschnitten.

6.3.2 Die Lösung: Flexible Bilder mit »max-width: 100%«

Schöner wäre es, wenn das Bild sich von der Breite her an den zur Verfügung stehenden Platz anpassen würde, und die Lösung benötigt nur eine einzige CSS-Regel:

```
img {
  max-width: 100%;
  height: auto;
}
```

Listing 6.5 Die CSS-Regel zum flexiblen Einbinden von Bildern

Abbildung 6.6 zeigt, dass sich das Foto auf der Beispelseite mit dieser CSS-Regel der Breite des Viewports anpasst und ganz zu sehen ist. Der Benutzer kann es bei Bedarf so groß zoomen, wie er möchte.



Abbildung 6.6 Das Bild passt sich der Breite des Viewports flexibel an.

Weil die Übungsdatei `bilder-flexibel-einbinden.html` nicht mit einer Stylesheet-Datei verbunden ist, erstellen Sie zum Speichern dieser CSS-Regel im `head` der HTML-Datei mit den Tags `<style>` und `</style>` einen Style-Block. Das ist ein spezieller Bereich zum Speichern von CSS-Regeln, die dann nur für diese HTML-Datei gelten.

Mit diesem Style-Block sieht der `head` der Übungsdatei so aus:

```
<head>
<!-- andere Elemente im head wie meta und title -->
<style>
```

```

img {
  max-width: 100%;
  height: auto;
}
</style>
</head>

```

Listing 6.6 Eine CSS-Regel in einem Style-Block gilt nur für diese Seite.

Dieser Trick funktioniert so:

- ▶ HTML-Elemente passen sich von Haus aus dem Viewport flexibel an.
- ▶ Die Anweisung `max-width: 100%` weist mit `img` eingefügten Bildern eine maximale Breite von 100 % zu. Im Klartext: Das Bild darf nicht breiter werden als das umgebende Element, in diesem Falle `body`.
- ▶ Wenn also `body` schmäler wird, schrumpft auch das Bild.
- ▶ Die Anweisung `height: auto` besagt, dass die Höhe des Bildes automatisch berechnet werden soll. So bleiben die Proportionen des Bildes erhalten, und es wird nicht verzerrt dargestellt.

6.4 Abbildungen beschriften: <figure> und <figcaption>

Die Elemente `figure` und `figcaption` dienen zur Beschriftung von Abbildungen aller Art. In diesem Abschnitt beschriften Sie das im vorherigen Abschnitt eingebundene Foto, in den folgenden Abschnitten dann auch noch Audio- und Videodateien.

6.4.1 Ein Foto mit einer Beschriftung: <figure> und <figcaption> im Einsatz

In HTML gibt es mit `figure` und `figcaption` zwei semantische Elemente zur Beschriftung von Abbildungen. Die Semantik lässt zum Beispiel Suchmaschinen verstehen, dass Abbildung und Beschriftung zusammengehören:

- ▶ Das Element `figure` umgibt die Abbildung und die Beschriftung, und beide werden zu einer Einheit.
- ▶ Das Element `figcaption` enthält den Text zur Beschriftung der Abbildung.

Das folgende Listing zeigt ein Foto mit einer Beschriftung:

```

<figure>
  
  <figcaption>Blick auf das Treppenviertel in Blankenese</figcaption>
</figure>

```

Listing 6.7 <figure> und <figcaption> im Einsatz

Dieses Beispiel sieht im Browser etwa so aus wie in Abbildung 6.7. Die Beschriftung steht unter dem Bild, aber rechts und links gibt es eine Einrückung, die gleich im nächsten Abschnitt entfernt wird.



Abbildung 6.7 Foto mit Beschriftung und Einrückung links und rechts

6.4.2 Die Einrückung von <figure> entfernen und die Beschriftung zentrieren

Die Einrückung links und rechts vom Bild entsteht dadurch, dass das Element `figure` vom Browser-Stylesheet rechts und links einen Außenabstand von 40px bekommt. Das ist vielleicht nett gemeint, stört aber eigentlich fast immer, und mit einer einfachen CSS-Regel entfernen Sie diese Einrückung.

Im folgenden Listing zentrieren Sie zusätzlich noch in `figure` enthaltene Inhalte und Inline-Elemente, sodass Bild und Beschriftung mittig untereinanderstehen:

```
figure {  
  text-align: center;  
  margin-left: 0;  
  margin-right: 0;  
}
```

Listing 6.8 <figure> bitte ohne Außenabstand und den Inhalt zentrieren

Falls Sie auch die Überschrift zentrieren möchten, fügen Sie noch die Regel `h1 { text-align: center; }` hinzu. Dieses CSS speichern Sie im Style-Block der Übungsdatei, und danach sieht die Webseite so aus wie in Abbildung 6.8.



Abbildung 6.8 Das fertig beschriftete Bild

Bilder per CSS gestalten

Wie Sie Bilder per CSS gestalten und mit Schlagschatten, runden Ecken und anderen Effekten versehen, erfahren Sie in Kapitel 16, »Das Box-Modell für Block- und Inline-Boxen«.

Kapitel 14

Der Browser und das CSS: Kaskade, Vererbung oder Standardwert

Worin Sie erfahren, warum die Kaskade in CSS so wichtig ist und was es mit Vererbung und Standardwert auf sich hat.

14

Die Themen im Überblick:

- ▶ Die Kaskade: Wichtigkeit, Spezifität und Reihenfolge, Seite 271
- ▶ Die Kaskade im Browser analysieren, Seite 277
- ▶ Nichts gefunden? Vererbung oder Standardwert, Seite 279
- ▶ Überblick: Kaskade, Vererbung oder Standardwert, Seite 281
- ▶ Auf einen Blick, Seite 283

In diesem kurzen, aber wichtigen Kapitel erfahren Sie, was es mit der namensgebenden *Kaskade* der *Cascading Style Sheets* auf sich hat und welche Rolle *Vererbung* und *Standardwert* dabei spielen.

Dieses Kapitel ist eher theoretischer Natur, und wenn Sie gerade keine Lust auf Theorie haben, können Sie es ruhig erst einmal überspringen. Falls Sie in Ihren Stylesheets allerdings vor scheinbar unlösbaren Phänomenen stehen, kommen Sie zurück, und lesen Sie sich dieses Kapitel ganz in Ruhe durch. Es ist die Antwort auf viele Rätsel.

14.1 Die Kaskade: Wichtigkeit, Spezifität und Reihenfolge

Sobald Stylesheets ein bisschen länger werden, gibt es für die CSS-Eigenschaften eines Elements fast zwangsläufig mehrere sich widersprechende Anweisungen. Die Frage ist, wie der Browser in solchen Konfliktfällen entscheidet, und die Antwort ist die *Kaskade*, die den *Cascading Style Sheets* ihren Namen gegeben hat.

14.1.1 Die Kaskade hilft dem Browser, genau eine Anweisung zu finden

Bei der Gestaltung eines Elements sammelt der Browser zunächst für jede CSS-Eigenschaft alle relevanten Anweisungen und schreibt sie auf einen metaphorischen Zettel.

Wenn es für eine Eigenschaft mehrere Anweisungen gibt, durchläuft er auf der Suche nach einem eindeutigen Wert einen dreistufigen Entscheidungsprozess namens Kaskade:

1. Zunächst sortiert der Browser nach *Wichtigkeit*. Er schaut, ob es eine Anweisung `!important` gibt. Gibt es genau eine, nimmt er den Wert aus dieser Anweisung und beendet die Kaskade.
2. Gibt es keine oder mehrere Anweisungen mit `!important`, prüft er die *Spezifität*. Hat genau eine Anweisung mehr Punkte als alle anderen, ist die Kaskade beendet.
3. Haben mehrere Anweisungen dieselbe Punktzahl, geht es um die *Reihenfolge* im CSS, und da kann es kein Unentschieden geben. Der Browser nimmt den Wert aus der zuletzt notierten Anweisung.

Ziel dieser Kaskade ist es, genau *eine* Anweisung zu finden, die der Browser dann anwendet.

14.1.2 Die Ausgangssituation: Das Beispiel im Überblick

Das HTML der in diesem Abschnitt gezeigten Übungsdatei ist sehr übersichtlich und besteht nur aus einer Überschrift und zwei Absätzen, von denen der erste die Klasse `intro` hat:

```
<h1>Die Kaskade</h1>
<p class="intro">Lorem ipsum dolor sit amet, ... </p>
<p>Donec quam felis, ultricies nec, ...</p>
```

Listing 14.1 Eine Überschrift und zwei Absätze

Schriftgröße und Zeilenabstand werden mit den Eigenschaften `font-size` und `line-height` entsprechend den vom Browser vorgegebenen Formatierungen für Absätze gestaltet. Diese Vorgaben stammen aus den Browserinstellungen und dem Browser-Stylesheet und bleiben normalerweise im Verborgenen. Zur Veranschaulichung werden sie im folgenden Listing gezeigt:

```
/* Grundlegende Formatierung vom Browser */
p {
  font-size: 1rem;
  line-height: normal;
}
```

Listing 14.2 Die Standardformatierung für Absätze

Nur mit diesen Browser-Styles sieht das HTML so aus wie in Abbildung 14.1.

Die Kaskade

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim.

Abbildung 14.1 Die Ausgangssituation

14.1.3 Beispiel Nr. 1: Die Reihenfolge entscheidet.

Im ersten Beispiel werden die Vorgaben vom Browser mit einer CSS-Regel überschrieben, die den Zeilenabstand auf das Anderthalbfache der Schriftgröße erhöht:

```
/* Grundlegende Formatierung vom Browser */
p {
  font-size: 1rem;
  line-height: normal;
}
/* Eigene Anweisungen überschreiben die Vorgaben vom Browser */
p {
  line-height: 1.5;
```

Listing 14.3 Der Zeilenabstand wird auf 1,5 erhöht.

Für den Zeilenabstand bekommt der Browser jetzt zwei Anweisungen:

- Die Vorgabe vom Browser lautet `line-height: normal`.
- In Listing 14.3 steht `line-height: 1.5`.

Um herauszufinden, welche Anweisung er nehmen soll, durchläuft der Browser blitzschnell die Kaskade:

1. Es gibt keine Anweisung mit dem Zusatz `!important`.
2. Die Selektoren haben die gleiche Spezifität.
3. Die Reihenfolge im Quelltext ergibt `line-height: 1.5`.

Abbildung 14.2 zeigt, dass die zweite Anweisung gewinnt und der Zeilenabstand für die Absätze erhöht wird. Dass die Reihenfolge im Quelltext zählt und weiter unten notierte Anweisungen gewinnen, finden die meisten Menschen intuitiv richtig und wenig überraschend.

Die Kaskade

```
 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

 Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim.
```

Abbildung 14.2 Die Reihenfolge zählt, und der Zeilenabstand wird erhöht.

Bei gleicher Wichtigkeit und Spezifität zählt die Reihenfolge

Bei gleicher Wichtigkeit und Spezifität zählt wirklich die *Reihenfolge* der Anweisungen im CSS: Wenn Sie die beiden Regeln in Listing 14.3 umdrehen, ist der Zeilenabstand *normal*. Probieren Sie es ruhig einmal aus.

14.1.4 Beispiel Nr. 2: Die Spezifität ist wichtiger als die Reihenfolge.

Im zweiten Beispiel wird das bisherige CSS um eine Regel für die Klasse `intro` ergänzt. Die Schriftgröße wird darin auf `1.25rem` erhöht:

```
 /* Grundlegende Formatierung vom Browser */
p {
  font-size: 1rem;
  line-height: normal;
}
```

```
 /* Eigene Anweisungen überschreiben die Vorgaben vom Browser */
p {
  line-height: 1.5;
}
p.intro {
  font-size: 1.25rem;
}
```

Listing 14.4 Die Schriftgröße für den ersten Absatz wird erhöht.

Abbildung 14.3 zeigt, dass wie erwartet die Schrift im ersten Absatz mit der Klasse `intro` mit diesem CSS etwas größer wird.

Die Kaskade

```
 Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Aenean commodo ligula eget dolor. Aenean massa. Cum
sociis natoque penatibus et magnis dis parturient montes,
nascetur ridiculus mus.
```

```
 Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla
consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec,
vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae,
justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras
dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend
tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim.
```

Abbildung 14.3 Die Schriftgröße für die Klasse »intro« wird erhöht.

Bemerkenswert ist, dass in diesem Fall die *Spezifität* entscheidend ist, und *nicht* die Reihenfolge der Regeln. Die Kaskade funktioniert in diesem Beispiel so:

1. Keine der Anweisungen hat `!important`.
2. Treffer. Der Selektor `p.intro` gewinnt mit 11 Punkten gegen das `p` mit nur 1 Punkt. Kaskade beendet.
3. Die Reihenfolge spielt keine Rolle mehr.

Der Grund für die größere Schrift im ersten Absatz ist die Spezifität der Klasse `p.intro` und *nicht* die Reihenfolge der CSS-Regeln. Auch wenn die Regel mit `font-size: 1.25rem` ganz am Anfang steht, bleibt die Schrift im ersten Absatz größer:

```
p.intro {
  font-size: 1.25rem;
}
```

```

p {
  font-size: 1rem;
  line-height: normal;
}
p {
  line-height: 1.5;
}

```

Listing 14.5 Die Reihenfolge verliert gegen die Spezifität.

Und an dieser Stelle entfernt sich die Erwartung der CSS schreibenden Autoren häufig von der Realität der Kaskade:

- ▶ Das Prinzip der *Reihenfolge* ist auch ohne CSS-Kenntnisse intuitiv verständlich.
- ▶ Das Konzept der *Spezifität* hingegen muss man gelernt und verstanden haben, um es nachvollziehen und anwenden zu können.

Merke: *Spezifität* ist wichtiger als die *Reihenfolge* im Quelltext.

Halten Sie die Spezifität so niedrig wie möglich

Spezifität ist also wichtiger als die Reihenfolge im Stylesheet, und das ist auch der Grund, warum man die Spezifität der Selektoren möglichst niedrig halten sollte. IDs und Inline-Styles haben mit 100 bzw. 1000 Punkten eine Spezifität, die nur schwierig zu übertrumpfen ist.

14.1.5 Beispiel Nr. 3: »!important« gewinnt immer.

Die Verwendung von `!important` ist wie gesagt so, als ob Sie den Browser anschreien. Das folgende Beispiel zeigt, dass `!important` die Stufen 2 und 3 der Kaskade außer Kraft setzt:

```

p {
  font-size: 1rem !important;
  line-height: normal;
}

p {
  line-height: 1.5;
}

```

```

p.intro {
  font-size: 1.25rem;
}

```

Listing 14.6 Die Schriftgröße für den ersten Absatz wird erhöht.

Durch `!important` werden *Spezifität* und *Reihenfolge* komplett ignoriert.

1. Nur die erste Anweisung hat `!important`. Fertig. Kaskade beendet.
2. Der Selektor `p.intro` würde gegen den Typselektor `p` von der Spezifität her gewinnen, aber das zählt nicht mehr.
3. Auch die Reihenfolge spielt in diesem Falle keine Rolle mehr.

Der Browser findet zwar alle Anweisungen für die `font-size` von Absätzen, aber bei der Sortierung trifft er schon in der ersten Stufe der Kaskade eine Entscheidung. Abbildung 14.4 zeigt, dass die Schriftgröße für alle Absätze `1rem` ist.

Die Kaskade

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim.

Abbildung 14.4 »!important« gewinnt gegen Spezifität und Reihenfolge.

14.2 Die Kaskade im Browser analysieren

In einem modernen Desktop-Browser wie Chrome oder Firefox können Sie die Auswirkungen der Kaskade sehr schön analysieren und nachvollziehen.

14.2.1 Die Kaskade im Entwicklerwerkzeug des Firefox

Abbildung 14.5 zeigt die Übungsseite im Firefox. Im INSPEKTOR ist der Absatz mit der Klasse `intro` markiert. Rechts daneben sehen Sie die CSS-Regeln aus Listing 14.6, und zwar in umgekehrter Reihenfolge, von unten nach oben.

The screenshot shows the Firefox developer tools' 'Stile' (Style) panel. It displays the CSS rules for the 'intro' class. The first rule, 'p.intro { font-size: 1.25rem; }', has a strikethrough line through it. The second rule, 'p.intro { font-size: 1rem !important; }', has an underline underneath 'font-size'. The third rule, 'p { line-height: 1.5; }', is shown with its original value. The 'Regeln' (Rules) tab is selected.

Abbildung 14.5 Die Kaskade im Entwicklerwerkzeug des Firefox

Im Rahmen der Kaskade überschriebene Anweisungen werden durchgestrichen:

- ▶ Für die Schriftgröße steht in Listing 14.6 das `!important` in der ersten Regel hinter `font-size: 1rem`.
- ▶ Etwas weiter oben ist `font-size: 1.25rem` durchgestrichen, weil es durch das `!important` überschrieben wird.

Beim Zeilenabstand ist es ähnlich:

- ▶ Ganz unten ist `line-height: normal` durchgestrichen, denn die Anweisung wird im Rahmen der Kaskade von einer folgenden Anweisung überschrieben.
- ▶ Und zwar von `line-height: 1.5` direkt darüber, mit dem der Zeilenabstand im Absatz gestaltet wird.

Zum Analysieren der Kaskade sind die Entwicklerwerkzeuge der Browser also eine große Hilfe. Falls beim Gestalten etwas nicht so klappt wie erwartet, hilft es oft, das CSS im Inspektor zu analysieren und sich anzuschauen, wie der Browser das CSS anwendet.

14.2.2 Der Trick mit dem Trichter: Anweisungen nach Eigenschaft filtern

Manchmal sieht man im CSS den Wald vor lauter Bäumen nicht, und dann hilft ein kleiner Trick, die Übersichtlichkeit etwas zu erhöhen:

- ▶ Hinter im Rahmen der Kaskade überschriebenen Anweisungen sehen Sie im Firefox ein kleines Trichtersymbol.
- ▶ Ein Klick darauf aktiviert einen Filter für diese Eigenschaft.
- ▶ Die gefilterten Anweisungen werden optisch hervorgehoben.
- ▶ Alle anderen Eigenschaften werden ausgeblendet.

Abbildung 14.6 zeigt die CSS-Regeln mit aktiviertem Filter für `font-size`. Um den Filter wieder auszuschalten, klicken Sie rechts im Eingabefeld für den Filter auf das x im dunklen Kreis.

The screenshot shows the Firefox developer tools' 'Stile' (Style) panel with a blue background applied to the 'font-size' column. The 'font-size' filter is active. The first rule, 'p.intro { font-size: 1.25rem; }', has a strikethrough line through it. The second rule, 'p.intro { font-size: 1rem !important; }', has an underline underneath 'font-size'. The third rule, 'p { line-height: 1.5; }', is shown with its original value. The 'Regeln' (Rules) tab is selected.

Abbildung 14.6 Ein Klick auf das Trichtersymbol aktiviert einen Filter.

Kapitel 19

Media Queries und responsives Webdesign

Worin Sie Media Queries und deren Möglichkeiten kennenlernen.

Die Themen im Überblick:

- ▶ »Getting responsive«: Das Web wird flexibel, Seite 383
- ▶ Medientypen definieren das Ausgabemedium, Seite 384
- ▶ Media Queries = Medientypen + Medieneigenschaften, Seite 387
- ▶ Media Queries und der »Meta-Viewport«, Seite 391
- ▶ Media Queries und der richtige Breakpoint, Seite 393
- ▶ Mehrspaltiger Text mit dem »Multi-column Layout«, Seite 394
- ▶ Auf einen Blick, Seite 396

In diesem Kapitel sehen Sie, was es mit *Responsive Web Design* auf sich hat, und lernen die `@media`-Regel kennen, mit der Sie *Medientypen* und *Media Queries* definieren können: *Medientypen* beschränken die Ausgabe auf ein bestimmtes Medium wie `screen` oder `print`, *Media Queries* sind ein wichtiges Werkzeug zur Erstellung responsiver Webseiten.

19.1 »Getting responsive«: Das Web wird flexibel

Der Begriff *Responsive Web Design* stammt aus einem im Mai 2010 erschienenen Artikel des Webdesigners Ethan Marcotte:

- ▶ alistapart.com/article/responsive-web-design

Marcotte beschreibt darin drei Zutaten für responsives Webdesign:

- ▶ flexible, prozentbasierte Seitenlayouts
- ▶ flexible Bilder
- ▶ Media Queries

Vereint ermöglichen diese drei Techniken es bereits in 2010, dass eine Webseite auf ihre Umgebung reagieren und sich zum Beispiel der Breite des Browserfensters anpassen konnte.

Heute wird der Begriff *responsiv* unabhängig von seiner ursprünglichen Bedeutung für alle Arten von anpassungsfähigen Webseiten genutzt, auch wenn sie keine Media Queries haben oder die Layouts nicht prozentbasiert sind, weil sie auf modernen Techniken wie Flexbox oder Grid basieren.

19.2 Medientypen definieren das Ausgabemedium

Mit @media kann man verschiedene Medientypen definieren. Mit *Medientyp* ist das Ausgabemedium für die Webseite gemeint.

19.2.1 Die Medientypen in der Übersicht

Webseiten werden typischerweise mit dem Medientyp *screen* an einem Bildschirm angezeigt, und das ist mit Abstand der wichtigste Medientyp. Eine Alternative wäre zum Beispiel *print* für eine Druckversion.

Tabelle 19.1 zeigt eine Übersicht der Medientypen.

Medientyp	Beschreibung
all	Für alle Ausgabemedien (Standardwert)
screen	Für Bildschirme aller Art und Größe
print	Für Drucker und die Druckvorschau am Bildschirm, wenn Dokumente seitenweise ausgegeben werden

Tabelle 19.1 Medientypen auf einen Blick

Wichtig zu wissen ist, dass, solange nichts anderes definiert wurde, immer der Medientyp *all* gilt.

Neben *screen* und *print* gibt es noch einen Medientyp namens *speech*, der bisher aber von fast keinem Browser unterstützt wird. Screenreader gehören übrigens zum Medientyp *screen*. Sie lesen eine Webseite zwar vor, aber die Sprachausgabe basiert auf dem visuellen Layout einer Seite.

Viele weitere Medientypen gelten als veraltet

Früher gab es noch zahlreiche andere Medientypen wie *tv*, *projection*, *handheld*, *braille*, *embossed* oder *aural*, die aber allesamt als veraltet (engl. *deprecated*) gelten und nicht mehr genutzt werden sollten. Wenn Sie mehr dazu wissen möchten:

► drafts.csswg.org/mediaqueries-4/#media-type

19.2.2 Eine Druckversion für die Übungswebsite mit »@media print«

Die Gestaltung der Druckausgabe geschieht im CSS mit folgender Anweisung:

```
@media print {
    /* CSS-Regeln für die Druckversion */
}
```

Listing 19.1 Die Syntax für @media print

Alle Regeln innerhalb der darauf folgenden geschweiften Klammern werden nur für den Ausdruck der Seiten verwendet. Das folgende Listing zeigt ein einfaches Beispiel:

```
/* =====
   print.css
   Einfache Druckversion für die Übungswebsite
===== */

@media print {
    * {
        box-shadow: none !important;
        text-shadow: none !important;
    }

    html { background-color: white; color black; }

    .site-nav, .site-footer { display: none; }

} /* Ende @media print */
```

Listing 19.2 Das Stylesheet für eine einfache Druckversion

Sie können diese Druckversion natürlich beliebig erweitern, aber hier zunächst eine kurze Erläuterung von Listing 19.2:

- Der Universalselektor * selektiert alle Elemente einer Seite.
- Die Anweisungen zur Entfernung der Schatten haben den Zusatz !important, damit sie auf jeden Fall gelten.
- Für das Stammelement html wird schwarze Schrift auf weißem Grund definiert.
- Navigation und Fußbereich werden mit display:none ausgeblendet, da Navigationslinks in einer Druckversion nicht sinnvoll sind.
- Der Kommentar /* Ende @media print */ dient nur zur Erinnerung, damit die schließende geschweifte Klammer nicht versehentlich gelöscht wird.

Im folgenden Kasten erstellen Sie eine Druckversion für die Übungswebsite.

Übungswebsite: Eine Druckversion mit »@media print«

1. Starten Sie Ihren Editor, und erstellen Sie eine neue Datei.
2. Speichern Sie die Datei als *print.css* im Unterordner *css*.
3. Fügen Sie das CSS aus Listing 19.2 ein.
4. Speichern Sie das Stylesheet, und öffnen Sie *style.css* im Editor.
5. Fügen Sie unterhalb der bestehenden Anweisungen eine @import-Anweisung für *print.css* ein: `@import url("print.css");`
6. Speichern Sie das Stylesheet.

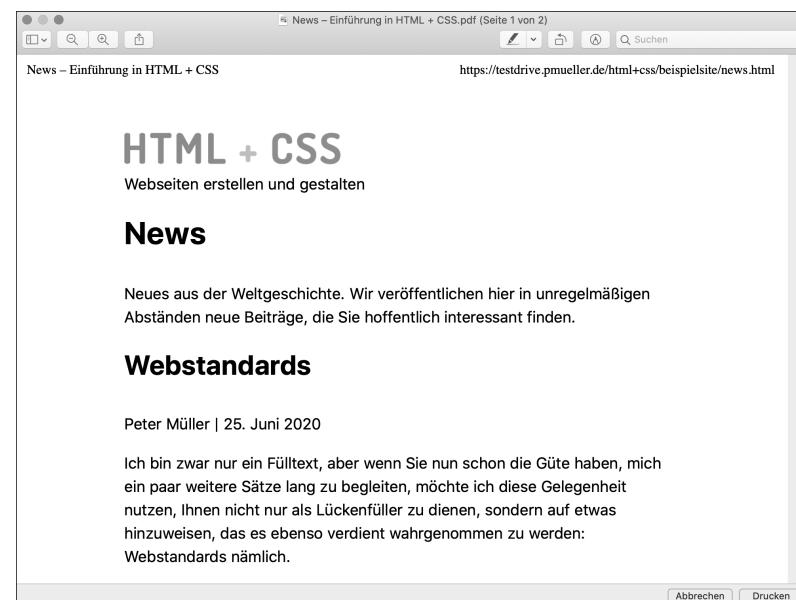


Abbildung 19.1 Die Print-Version der Seite »News«

Unter Windows hat Firefox zum Testen der Druckversion im Menü DATEI eine DRUCKVORSCHAU, unter macOS können Sie im Menü DATEI mit dem Befehl DRUCKEN... ein PDF erzeugen und sich das Ergebnis mit IN VORSCHAU ÖFFNEN direkt anzeigen lassen. Abbildung 19.1 zeigt die Druckversion der Seite News. Die Navigation wird nicht angezeigt.

19.3 Media Queries = Medientypen + Medieneigenschaften

Mit CSS3 wurden Media Queries eingeführt, was wörtlich übersetzt *Medienabfrage* heißt. Diese Abfragen ermöglichen es, einen Medientyp wie *screen* mit bestimmten *Medieneigenschaften* (engl. *media features*) zu kombinieren.

Medieneigenschaften sind zum Beispiel Dinge wie die Viewportbreite, die Bildschirmauflösung oder die Orientierung eines Geräts, also ob es gerade im Hoch- oder im Querformat gehalten wird.

Sie können diese Eigenschaften abfragen und anhand von definierten Bedingungen beim Gestalten der Seiten darauf reagieren, sodass die Seiten sich der Umgebung anpassen.

Mit einer Media Query können Sie einem Browser zum Beispiel sagen, dass er bestimmte CSS-Regeln nur anwenden soll, wenn der Viewport eine bestimmte Mindestbreite hat.

19.3.1 Die Syntax: »@media Medientyp and (Eigenschaft: Wert)«

Media Queries werden meist mit einer @media-Regel in einem Stylesheet definiert. Im folgenden Listing werden die Regeln zwischen den geschweiften Klammern vom Browser nur angewendet, wenn die Seite auf einem Bildschirm ausgegeben wird und der Viewport mindestens 600px breit ist:

```
@media screen and (min-width: 600px) { /* CSS-Regeln */ }
```

Listing 19.3 Media Query für eine Mindestbreite von 600px

Die Syntax ist recht einfach:

- Nach @media kommt der Medientyp *screen*.
- Danach folgt ein *and*, das den Medientyp und die in ganz normalen Klammern stehende Bedingung miteinander verknüpft.

- In den Klammern steht ein Ausdruck wie `min-width: 600px`, der für eine *Eigenschaft* wie `min-width` einen *Wert* wie `600px` definiert.

Die Abfrage der Eigenschaft `min-width` bezieht sich, wie gesagt, auf die Mindestbreite des Viewports im Browser, *nicht* auf die Breite des Bildschirms und *auch nicht* auf die Breite des umgebenden Elements.

19.3.2 Ein Beispiel: Eine Media Query zur Änderung der Hintergrundfarbe

Ein kleines Beispiel zeigt, wie Media Queries funktionieren. Im folgenden Listing bekommt `body` ab einer Viewportbreite von `600px` eine andere Hintergrund- und Schriftfarbe:

```
body {
  font-family: sans-serif; line-height: 1.5;
  background: white; color: black;
  padding: 1rem;
}
@media screen and (min-width: 600px) {
  body {
    background-color: steelblue;
    color: white;
  }
}
```

Listing 19.4 Media Queries für verschiedene Hintergrundfarben

Die erste CSS-Regel steht außerhalb der Media Query und gilt immer. Innerhalb der Media Query werden Hintergrund- und Schriftfarbe für `body` geändert, Schriftart, Zeilenabstand und `padding` bleiben hingegen unverändert.

Abbildung 19.2 zeigt dieses Beispiel im Firefox mit der Funktion BILDSCHIRMGRÖSSEN TESTEN, die Sie im Menü WEB-ENTWICKLER finden:

- Oben hat der Viewport eine Breite von `375px`. Der Hintergrund ist weiß und die Schrift schwarz.
- Unten ist der Viewport `600px` breit. Der Hintergrund ist wie in der Media Query definiert `steelblue`, und die Schrift wird weiß.

Alle anderen Eigenschaften bleiben unverändert.



Abbildung 19.2 Bei einer Viewportbreite von `600px` ändern sich die Farben.

Die Umbruchstelle bei einer Media Query nennt man »Breakpoint«

Den Punkt, an dem sich die Darstellung durch eine Media Query ändert, nennt man im CSS-Jargon *Breakpoint*. In diesem Abschnitt wurde also bei `600px` ein *Breakpoint* definiert. Mehr zur Definition von Breakpoints erfahren Sie in Abschnitt 19.5.

19.3.3 Weitere Beispiele für Media Queries

Man kann als Bedingung auch eine maximale Breite angeben. Wenn der Viewport maximal `599px` breit sein soll, lautet die Media Query wie folgt:

```
@media screen and (max-width: 599px) {
  /* CSS-Regeln */
```

Listing 19.5 Medientyp mit Media Query für eine maximale Breite

Auch eine Kombination von mehreren Eigenschaften ist problemlos möglich:

```
@media screen and (min-width:320px) and (max-width:767px) {
  /* CSS-Regeln */
```

Listing 19.6 Media Query mit zwei Bedingungen

Die innerhalb dieser Abfrage definierten CSS-Regeln sind nur gültig, wenn der Viewport zwischen 320 und 767 Pixel groß ist.

Sie können auch mehrere Media Queries mit verschiedenen Breakpoints nacheinander nutzen:

```
body { background: white; color: black; padding: 1rem; }
```

```
@media screen and (min-width: 600px) {  
  body { background-color: steelblue; color: white; }  
}
```

```
@media screen and (min-width: 1200px) {  
  body { background-color: darkblue; color: white; }  
}
```

Listing 19.7 Mehrere Media Queries mit verschiedenen Breakpoints

In einem schmalen Viewport ist der Hintergrund white, ab einer Breite von 600px wird er steelblue und ab 1200px dann darkblue.

19.3.4 Die wichtigsten Medieneigenschaften im Überblick

In den meisten Fällen werden Sie mit Media Queries die Breite eines Viewports testen, aber es gibt auch andere Abfragen. Tabelle 19.2 zeigt die wichtigsten Eigenschaften auf einen Blick.

Medieneigenschaft	Beschreibung
width	Viewport-Breite inklusive Rollbalken; meist als min-width oder max-width abgefragt.
height	Viewport-Höhe, meist als min-height bzw. max-height
orientation	Querformat landscape, Hochformat portrait
resolution	Auflösung des Bildschirms

Tabelle 19.2 Die wichtigsten Eigenschaften für Media Queries

Auf einen Blick

TEIL I Webseiten, HTML und CSS

- | | | |
|---|---|----|
| 1 | Wissenswertes über Webseiten | 35 |
| 2 | HTML kennenlernen: Die erste Webseite erstellen | 47 |
| 3 | CSS kennenlernen: Die erste Webseite gestalten | 69 |

TEIL II HTML (mit einer Prise CSS)

- | | | |
|----|---|-----|
| 4 | HTML-Elemente für Überschriften, Text und Listen | 85 |
| 5 | Hyperlinks – das Besondere am Web | 103 |
| 6 | HTML-Elemente für Bilder, Audio und Video | 121 |
| 7 | HTML-Elemente zur Strukturierung von Webseiten und Inhalten | 139 |
| 8 | Weitere HTML-Elemente zur Auszeichnung von Inhalten | 159 |
| 9 | HTML-Elemente zum Erstellen von Formularen | 179 |
| 10 | HTML-Elemente zum Erstellen von Tabellen | 201 |
| 11 | Von der Webseite zur Website | 213 |

TEIL III CSS – Grundlagen

- | | | |
|----|---|-----|
| 12 | Gestalten per CSS: Farben und Einheiten | 237 |
| 13 | Die wichtigsten Selektoren und Spezifität | 253 |
| 14 | Der Browser und das CSS: Kaskade, Vererbung oder Standardwert | 271 |
| 15 | Schrift und Text gestalten per CSS | 285 |
| 16 | Das Box-Modell für Block- und Inline-Boxen | 307 |
| 17 | Boxen gestalten per CSS | 329 |
| 18 | Ordnung halten: Stylesheets organisieren | 357 |

TEIL IV CSS – Layout

- | | | |
|----|--|-----|
| 19 | Media Queries und responsives Webdesign | 383 |
| 20 | Der Flow und die Eigenschaft »position« | 399 |
| 21 | Schweben und schweben lassen: »float« | 413 |
| 22 | Gestalten per Flexbox: Das »CSS Flexible Box Layout« | 423 |
| 23 | Eine responsive Navigation erstellen | 443 |
| 24 | Gestalten mit Raster: Das »CSS Grid Layout« | 459 |
| 25 | Flexible Icons und responsive Bilder | 483 |

Inhalt

Materialien zum Buch	26
Geleitwort	27
Vorwort	29

TEIL I Webseiten, HTML und CSS

1 Wissenswertes über Webseiten 35

1.1 Webseiten sehen bei jedem Benutzer anders aus	35
1.2 Webseiten bestehen aus Quelltext	36
1.3 Quelltext besteht aus HTML, CSS und JavaScript	37
1.3.1 Der Inhalt: HTML ist nicht hübsch, aber flexibel	38
1.3.2 Das Styling: CSS gestaltet das HTML	39
1.4 Webseiten werden von einem Browser dargestellt	40
1.4.1 Die bekanntesten Browser: Chrome, Firefox, Safari, Edge und Co	40
1.4.2 Viele Browser sind miteinander verwandt	41
1.4.3 Besonderheiten: Browser unter iOS und Internet Explorer	41
1.5 Editoren zum Schreiben und Bearbeiten von Quelltext	42
1.6 Referenzen und Nachschlagewerke zu HTML und CSS	43
1.6.1 SelfHTML – das deutschsprachige Urgestein	43
1.6.2 Das »Mozilla Developer Network« (MDN) – mostly English	44
1.6.3 Anlaufstelle für Fragen zur Browserunterstützung: »caniuse.com«	45
1.7 Auf einen Blick	46

2 HTML kennenlernen: Die erste Webseite erstellen 47

2.1 Webseiten bestehen aus rechteckigen Kästchen	48
2.2 HT-M-L: die »HyperText Markup Language«	49
2.2.1 HT wie Hypertext – Text mit Hyperlinks	49
2.2.2 M wie Markup – Text mit <tags>	49

2.2.3	L wie Language – Vokabeln und Grammatikregeln	49
2.2.4	Der Unterschied zwischen »Element« und »Tag«	50
2.3	Jede Webseite hat ein HTML-Grundgerüst	50
2.3.1	Die Datei »index.html« im Editor erstellen und speichern	51
2.3.2	Eine gute Angewohnheit: <!-- Kommentare -->	51
2.3.3	Das HTML-Grundgerüst für die Startseite erstellen	52
2.4	Der <!doctype> und das Stammelement <html>	54
2.4.1	Die Dokumenttyp-Definition <!doctype html>	54
2.4.2	Das Stammelement: <html> und </html> umschließen den Quelltext	55
2.5	HTML-Elemente können Attribute haben	55
2.6	Der <head> enthält wichtige Infos über die Webseite	56
2.6.1	Die Angabe des Zeichensatzes: <meta charset="utf-8">	56
2.6.2	Der Seitentitel steht zwischen <title> und </title>	57
2.6.3	Die Seitenbeschreibung mit <meta name="description">	58
2.7	Bitte nicht verkleinern: <meta name="viewport" ...>	59
2.7.1	Mobile Browser stellen Webseiten verkleinert dar	59
2.7.2	Responsive Webseiten sollen nicht verkleinert werden	59
2.7.3	Der Meta-Viewport gehört mit ins HTML-Grundgerüst	60
2.8	Der sichtbare Bereich der Webseite steht in <body>	61
2.9	Der Kopfbereich der Webseiten: <header>	63
2.10	Entwicklerwerkzeuge im Browser: HTML analysieren	64
2.11	Eine sehr kurze Geschichte von HTML	65
2.11.1	Das W3C definierte die Standards: HTML 1 bis 4 und XHTML	65
2.11.2	W3C und WHATWG: HTML5 und »HTML – Living Standard«	66
2.12	Auf einen Blick	67
3	CSS kennenlernen: Die erste Webseite gestalten	69
3.1	Jeder Browser hat ein fest eingebautes Stylesheet	69
3.2	Das HTML für <body> als schematische Darstellung	70
3.3	Das erste eigene Stylesheet: »style.css«	72
3.3.1	Schritt 1: Einen Unterordner und ein Stylesheet erstellen	72

3.3.2	Schritt 2: HTML-Datei und CSS-Datei verbinden mit <link>	72
3.4	Die erste CSS-Regel: Hintergrundfarbe für <body>	73
3.4.1	Auch in CSS eine gute Angewohnheit: /* Kommentare */	74
3.4.2	Hintergrund- und Schriftfarbe für <body> ändern	74
3.5	Den Kopfbereich <header> selektieren und gestalten	76
3.5.1	Hintergrund- und Schriftfarbe für <header> ändern	76
3.5.2	Etwas Abstand zwischen Text und Rand einfügen mit »padding«	77
3.6	Wichtige Vokabeln: Der Aufbau einer CSS-Regel	78
3.7	Entwicklerwerkzeuge im Browser: CSS analysieren	79
3.8	Eine sehr kurze Geschichte von CSS	80
3.9	Auf einen Blick	81

TEIL II HTML (mit einer Prise CSS)

4	HTML-Elemente für Überschriften, Text und Listen	85
4.1	Überschriften strukturieren den Inhalt: <h1> bis <h6>	85
4.1.1	HTML kennt sechs Ebenen für Überschriften	86
4.1.2	Eine <h2>-Überschrift im Inhaltsbereich einfügen	87
4.2	Absätze und Hervorhebungen: <p>, , 	88
4.2.1	Absätze mit <p> und Hervorhebungen mit und 	88
4.2.2	Fließtext auf der Übungswebsite einfügen	88
4.2.3	HTML-Elemente verschachteln – zuerst geöffnet, zuletzt geschlossen	89
4.3	Webseiten in unterschiedlich großen Viewports testen	90
4.4	Listen erstellen mit , und 	92
4.4.1	Aufzählungen: Ungeordnete Listen mit und 	92
4.4.2	Nummerierungen: Geordnete Listen mit und 	93
4.5	Listen verschachteln: Eine Liste in einer Liste	94
4.6	Beschreibungslisten mit <dl>, <dd> und <dt>	96
4.6.1	Das HTML für eine Beschreibungsliste	96
4.6.2	Eine einfache Gestaltung für eine Beschreibungsliste	97

4.7	Ein kurzer Blick auf das Browser-Stylesheet	98
4.7.1	Das Browser-Stylesheet des Firefox heißt »html.css«	98
4.7.2	Nicht hübsch, aber lesbar – was ein Browser-Stylesheet macht	99
4.7.3	Die CSS-Regel für eine <code><h1></code> -Überschrift aus dem Browser-Stylesheet	100
4.8	Über Blockelemente, Inline-Elemente und »display«	100
4.8.1	Blockelemente werden so breit, wie es geht	100
4.8.2	Inline-Elemente werden nur so breit wie ihr Inhalt	101
4.8.3	Block- und Inline-Elemente im Browser-Entwicklertool	101
4.9	Auf einen Blick	102
5	Hyperlinks – das Besondere am Web	103
5.1	Das Standardverhalten von Hyperlinks	103
5.2	Anatomie eines Hyperlinks: <code>Linktext</code>	104
5.3	Hyperlinks in neuem Tab oder Fenster öffnen	106
5.4	Eine Navigation ist eine Liste mit Links	107
5.5	Eine grundlegende Gestaltung für die Navigation	109
5.5.1	Schritt 1: Die Listenelemente nebeneinanderstellen	109
5.5.2	Schritt 2: Den Navigationsbereich und die Liste gestalten	111
5.5.3	Schritt 3: Die Links und den Linktext gestalten	112
5.6	Im Fußbereich einen Link »Nach oben« einfügen	114
5.6.1	Schritt 1: Das HTML für einen Link nach oben auf derselben Seite	114
5.6.2	Schritt 2: Eine grundlegende Gestaltung für den Footer und den Link	115
5.7	Besondere Links: Dateien, E-Mail und Telefon	117
5.7.1	Hyperlinks auf Dateien, die keine Webseiten sind: PDF & Co	117
5.7.2	Links auf E-Mail-Adressen	118
5.7.3	Links auf Telefonnummern	118
5.8	Auf einen Blick	119

6	HTML-Elemente für Bilder, Audio und Video	121
6.1	Über Grafikformate im Web: JPEG, GIF, PNG und SVG	121
6.2	Ein Bild als Logo einbinden mit <code></code>	123
6.2.1	Das Element <code></code> und seine wichtigsten Attribute	123
6.2.2	Ein Logo auf der Übungswesite einfügen mit <code></code>	124
6.2.3	Hochauflösende Bildschirme benötigen größere Bilder	125
6.2.4	Fine-Tuning: Die Abstände um Logo und Slogan anpassen	126
6.3	Bilder mit flexibler Breite: »max-width: 100%«	127
6.3.1	Das Problem: Pixelbilder haben eine feste Breite	127
6.3.2	Die Lösung: Flexible Bilder mit »max-width: 100%«	128
6.4	Abbildungen beschriften: <code><figure></code> und <code><figcaption></code>	130
6.4.1	Ein Foto mit einer Beschriftung: <code><figure></code> und <code><figcaption></code> im Einsatz	130
6.4.2	Die Einrückung von <code><figure></code> entfernen und die Beschriftung zentrieren	131
6.5	Audiodateien einbinden mit <code><audio></code>	133
6.5.1	Audioformate, Browserunterstützung und Audioplayer	133
6.5.2	Die Einbindung von Sound-Dateien mit <code><audio></code>	133
6.5.3	Audiodateien beschriften mit <code><figure></code> und <code><figcaption></code>	134
6.6	Bewegte Bilder einbinden mit <code><video></code>	135
6.6.1	Videoformate und Browserunterstützung im Überblick	135
6.6.2	Die Einbindung von Videodateien mit <code><video></code>	136
6.6.3	Flexible Videos per CSS mit »max-width: 100%«	137
6.7	Auf einen Blick	138
7	HTML-Elemente zur Strukturierung von Webseiten und Inhalten	139
7.1	Die semantischen Strukturelemente auf einen Blick	140
7.2	Kopfbereiche auszeichnen: <code><header></code>	140
7.2.1	Das Element <code><header></code> kann auf einer Seite mehrfach vorhanden sein	140
7.2.2	Den Kopfbereich auf der Übungswesite um eine Klasse erweitern	142

7.3	Navigationsbereiche erstellen mit <nav>	143
7.3.1	<nav> für die Site-Navigation auf der Übungswebsite	143
7.3.2	Den Navigationsbereich auf der Übungswebsite um eine Klasse erweitern	144
7.3.3	<nav> kann in der HTML-Struktur auch an anderen Positionen stehen	145
7.4	Der Hauptinhalt einer Webseite steht in <main>	146
7.4.1	Das Element <main> für den Hauptinhalt einer Webseite	146
7.4.2	Den Inhaltsbereich der Übungswebsite um eine Klasse erweitern	147
7.5	Fußbereiche auszeichnen: <footer>	147
7.5.1	Der Fußbereich <footer> auf der Übungswebsite	147
7.5.2	Den Fußbereich auf der Übungswebsite um eine Klasse erweitern	148
7.6	Inhaltliche Abschnitte erstellen: <section>	148
7.7	In sich geschlossene, eigenständige Blöcke: <article>	151
7.7.1	Grundlegende Gestaltung für den Abschnitt und die Infoboxen	153
7.8	Bereiche mit zusätzlichen Informationen: <aside>	154
7.9	Elemente semantisch neutral gruppieren: <div>	156
7.10	Auf einen Blick	158

8	Weitere HTML-Elemente zur Auszeichnung von Inhalten	159
8.1	Zitate auszeichnen mit <blockquote> und <cite>	159
8.1.1	Das HTML für Blockzitate: <blockquote> und <cite>	160
8.1.2	Ein Blockzitat mit einem <footer> für die Quellenangabe	161
8.1.3	Eine einfache Gestaltung für ein Zitat	162
8.2	Einen Zeilenumbruch erzwingen mit
	163
8.3	Kontaktinformationen auszeichnen mit <address>	163
8.3.1	Eine Kontaktadresse auszeichnen mit <address>	164
8.3.2	Eine grundlegende Gestaltung für eine Kontaktadresse	164
8.4	Änderungen am Text dokumentieren: und <ins>	165
8.4.1	Das HTML für Änderungen am Text	165
8.4.2	Eine einfache Gestaltung für Änderungen am Text	166

8.5	Zeitangaben für Menschen und Maschinen: <time>	167
8.5.1	Datumsangaben mit <time>	167
8.5.2	Die Uhrzeit mit <time>	168
8.6	Kurz vorgestellt: , <hr> und <small>	169
8.6.1	 ist ein semantisch neutrales Inline-Element	170
8.6.2	<hr> markiert einen inhaltlichen Bruch innerhalb eines Abschnitts	170
8.6.3	Das sprichwörtliche Kleingedruckte mit <small>	170
8.7	Ausklappbare Inhalte: <details> und <summary>	171
8.7.1	Das HTML für ausklappbare Inhalte: <details> und <summary>	171
8.7.2	Eine grundlegende Gestaltung für <details> und <summary>	172
8.8	Weitere Inline-Elemente in der Übersicht	174
8.9	Know-how: Zeichensätze und Sonderzeichen	175
8.9.1	UTF-8: Wissenswertes über Zeichensätze	175
8.9.2	Die Kodierung von Sonderzeichen in HTML	176
8.10	Auf einen Blick	178

9	HTML-Elemente zum Erstellen von Formularen	179
9.1	Formulare dienen zur Interaktion mit den Besuchern	179
9.2	Das Element <form> definiert ein Formular	180
9.3	Einzelige Eingabefelder mit <input> und <label>	181
9.3.1	Ein einziges Eingabefeld für Text: <input type="text">	182
9.3.2	Die Beschriftung eines Formularfeldes mit <label>	182
9.3.3	Ein Eingabefeld für E-Mail-Adressen: <input type="email">	184
9.3.4	Pflichtfelder definieren: das Attribut »required«	185
9.4	Mehrzeilige Eingabefelder mit <textarea> und <label>	186
9.5	Kontrollkästchen und Optionsfelder	187
9.5.1	Eckige Kontrollkästchen mit <input type="checkbox">	187
9.5.2	Runde Optionsfelder mit <input type="radio">	188
9.6	Formularfelder gruppieren: <fieldset> und <legend>	189
9.7	Ein Button zum Abschicken der Formulardaten	190
9.8	Ein DSGVO-kompatibles Kontaktformular erstellen	192
9.8.1	Schritt 1: Das HTML für die Eingabefelder	193

9.8.2	Schritt 2: DSGVO-Einverständnis per Kontrollkästchen	194
9.8.3	Schritt 3: Eine grundlegende Gestaltung für das Formular	195
9.8.4	Schritt 4: Beschriftung und Formularfelder ausrichten	196
9.9	Auf einen Blick	198
10	HTML-Elemente zum Erstellen von Tabellen	201
10.1	Eine einfache HTML-Tabelle: <table>, <tr> und <td>	201
10.2	Tabellenüberschriften stehen in <th>	203
10.3	Tabellen strukturieren: <thead>, <tbody> und <tfoot>	204
10.4	Zellen verbinden mit »colspan« und »rowspan«	205
10.5	HTML-Tabellen erstellen und gestalten – ein Beispiel	206
10.5.1	Schritt 1: Das HTML für die Beispieltabelle	207
10.5.2	Schritt 2: Eine grundlegende Gestaltung für die Beispieltabelle	209
10.5.3	Schritt 3: Zwischenraum kontrollieren mit »border-spacing« und »border-collapse«	210
10.6	Auf einen Blick	211
11	Von der Webseite zur Website	213
11.1	Fine-Tuning für die Startseite	213
11.1.1	Eine Klasse für die Seite: <body class="startseite">	214
11.1.2	»Sie sind hier«: Den aktuellen Menüpunkt hervorheben	214
11.1.3	Im Footer: Links zu Impressum und Datenschutz einfügen	216
11.2	Das HTML überprüfen mit dem HTML-Validator	218
11.3	Die Seiten »News«, »Über uns« und »Kontakt« erstellen	220
11.3.1	Die Seite »News« erstellen und anpassen	220
11.3.2	Die Seiten »Über uns« und »Kontakt« erstellen und anpassen	222
11.4	Die Seite »News« mit Inhalt füllen	224
11.4.1	Einen neuen Abschnitt hinzufügen: <section class="beitagsliste">	224
11.4.2	Einen Bereich mit Linklisten erstellen: <aside class="linklisten">	226
11.4.3	Eine grundlegende Gestaltung für die Inhalte der Seite »News«	227

11.5	Ein Bild auf der Seite »Über uns« einfügen	228
11.6	Kontaktdaten und Formular für die Seite »Kontakt«	229
11.6.1	Den Abschnitt »Kontaktdaten« hinzufügen	230
11.6.2	Einen Abschnitt mit einem Kontaktformular hinzufügen	231
11.7	Die Seiten »Impressum« und »Datenschutz«	233
11.8	Auf einen Blick	234

TEIL III CSS – Grundlagen

12	Gestalten per CSS: Farben und Einheiten	237
12.1	Überblick: Webseiten gestalten per CSS	237
12.2	CSS kann an drei Stellen definiert werden	238
12.2.1	Externes Stylesheet: CSS-Regeln in einer eigenen Datei	238
12.2.2	»Style-Block«: CSS-Regeln mit <style> im <head> einer Webseite	239
12.2.3	»Inline-Styles«: Anweisungen mit dem Attribut »style« im Element	239
12.2.4	Die empfohlene Vorgehensweise: CSS-Datei, <style> und style=" "	240
12.3	Farbnamen in CSS: Einfach, aber nicht sehr flexibel	240
12.4	Hexadezimale Farbangaben: #rrggb	241
12.4.1	Der Aufbau eines hexadezimalen Farbwertes	242
12.4.2	Die hexadezimale Kurzschreibweise: #rgb	242
12.4.3	Übersicht: Einige Farbnamen und ihre HEX-Werte	243
12.4.4	HEXen und blaufarben: Farbnamen auf der Übungswebsite ändern	243
12.5	Farben definieren mit rgb() und rgba()	244
12.6	Werkzeuge und Websites zur Arbeit mit Farben	246
12.6.1	Firefox hat in den Entwicklerwerkzeugen einen Farbwähler	246
12.6.2	Ausführliche Farbauswahl in den Entwicklerwerkzeugen der Browser	247
12.7	Wichtige Einheiten: »px«, »em«, »rem«, »%« & Co.	248
12.7.1	Die Einheit »px«: Ein Pixel ist nicht immer ein Pixel	249
12.7.2	Die Einheit »em« hat verschiedene Berechnungsgrundlagen	250

12.7.3	Die Einheit »rem« entspricht der Standardschriftgröße des Browsers	251
12.7.4	Die Einheit »%« (Prozent) ist meist relativ zum Elternelement	251
12.8	Auf einen Blick	252

13 Die wichtigsten Selektoren und Spezifität

13.1	Einfache Selektoren: Elemente, Gruppierung und »*«	253
13.1.1	»Der Name der Kiste« – einfache Elementselektoren	254
13.1.2	Mehrere Kästchen zugleich: Selektoren mit einem Komma gruppieren	254
13.1.3	Alle Kästchen auswählen: der Universalselektor »*«	255
13.2	Klassen sind klasse: Der Selektor mit dem Punkt	255
13.2.1	Klassen zur getrennten Gestaltung von gleichnamigen Elementen	255
13.2.2	Klassen zum Gruppieren von Elementen zur gemeinsamen Gestaltung	256
13.2.3	Gebundene Klassen: Klassen auf einen Elementtyp beschränken	256
13.2.4	Ein HTML-Element kann mehrere Klassennamen haben	257
13.3	IDs sind einmalig: Der Selektor mit der Raute	257
13.4	Selektoren für Nachfahren und Kinder	258
13.4.1	Familienaufstellung: HTML-Elemente im DOM-Baum	259
13.4.2	Der Nachfahrenselektor: Der Selektor mit dem Leerzeichen	260
13.4.3	Der Kindselektor: Der Selektor mit dem »» (Größer-als-Zeichen)	260
13.4.4	Spezielle Kinder selektieren mit »:first-child« und »:last-child«	261
13.4.5	Die Geschwisterselektoren: Pluszeichen »+« und Tilde »~«	262
13.5	Attributselektoren haben eckige Klammern	264
13.5.1	Nur das Attribut: element[attribut]	264
13.5.2	Mit einem Gleichheitszeichen: element[attribut="zeichen"]	265
13.5.3	Mit Tilde und Gleichheitszeichen: element[attribut~="zeichen"]	265
13.5.4	Mit Hütchen und Gleichheitszeichen: element[attribut^="zeichen"] ...	265
13.5.5	Mit Dollar und Gleichheitszeichen: element[attribut\$="zeichen"]	266
13.5.6	Mit Sternchen und Gleichheitszeichen: element[attribut*="zeichen"]	266
13.6	Spezifität: Einige Selektoren sind wichtiger als andere	267

13.6.1	Einer wird gewinnen: So funktioniert Spezifität	267
13.6.2	Ganz sparsam benutzen: »!important« macht Anweisungen WICHTIG!	268
13.7	Auf einen Blick	269

14 Der Browser und das CSS: Kaskade, Vererbung oder Standardwert

14.1	Die Kaskade: Wichtigkeit, Spezifität und Reihenfolge	271
14.1.1	Die Kaskade hilft dem Browser, genau eine Anweisung zu finden	272
14.1.2	Die Ausgangssituation: Das Beispiel im Überblick	272
14.1.3	Beispiel Nr. 1: Die Reihenfolge entscheidet.	273
14.1.4	Beispiel Nr. 2: Die Spezifität ist wichtiger als die Reihenfolge.	274
14.1.5	Beispiel Nr. 3: »!important« gewinnt immer.	276
14.2	Die Kaskade im Browser analysieren	277
14.2.1	Die Kaskade im Entwicklerwerkzeug des Firefox	277
14.2.2	Der Trick mit dem Trichter: Anweisungen nach Eigenschaft filtern	279
14.3	Nichts gefunden? Vererbung oder Standardwert	279
14.3.1	»Vererbung« macht ein Stylesheet übersichtlicher	279
14.3.2	Bestimmte Eigenschaften werden nicht vererbt	280
14.3.3	Falls er gar nichts findet, nimmt der Browser den »Standardwert«	281
14.4	Überblick: Kaskade, Vererbung oder Standardwert	281
14.5	Auf einen Blick	283

15 Schrift und Text gestalten per CSS

15.1	Klassische Schriftarten im Web	285
15.1.1	Schriftarten mit und ohne Serifen	285
15.1.2	Sehr praktisch: Die Schriftgestaltung im Firefox analysieren	286
15.2	Die Schriftart definieren mit »font-family«	287
15.2.1	Bitte eine Schrift ohne Serifen: »font-family: sans-serif«	287
15.2.2	Generische Schriftfamilien im Überblick	289
15.2.3	Die Systemschrift: Gut zu lesen und echt schnell	289

15.3	Webfonts – die Schriftart gleich mitliefern	291
15.3.1	Webfonts im Überblick	291
15.3.2	Einfach, schnell und kostenlos: Google Fonts	292
15.3.3	Schritt 1: Schriftart und Schriftschnitte auswählen	293
15.3.4	Schritt 2: Den Code für die Schriftart kopieren und einfügen	293
15.4	Gut lesbarer Fließtext: »font-size« und »line-height«	295
15.4.1	Schriftgröße definieren mit »font-size« und einer Längeneinheit	295
15.4.2	Schriftgröße definieren mit »font-size« und einem Wort	297
15.4.3	Wichtig für die Lesbarkeit: Der Zeilenabstand mit »line-height«	298
15.5	Hyperlinks gestalten mit Pseudoklassen	299
15.5.1	Besuchte und nicht besuchte Hyperlinks mit »:link« und »:visited«	299
15.5.2	Benutzeraktionen gestalten mit »:hover«, »:focus« und »:active«	301
15.6	Weitere Eigenschaften zur Schrift- und Textgestaltung	303
15.6.1	Die wichtigsten Eigenschaften zur Schrift- und Textgestaltung im Überblick	303
15.6.2	Schrift gestalten: fett, kursiv, Kapitälchen und Zeichenabstand	303
15.6.3	Die Kurzschriftweise »font«	304
15.6.4	Text ausrichten und die erste Zeile einrücken	305
15.6.5	Schatten im Text: »text-shadow«	305
15.7	Auf einen Blick	306

16	Das Box-Modell für Block- und Inline-Boxen	307
16.1	Das klassische Box-Modell für Blockboxen	307
16.1.1	Breite und Höhe für den Inhalt definieren: »width«, »height« & Co	308
16.1.2	Der Innenabstand »padding« schafft Platz zwischen Inhalt und Rand	309
16.1.3	Die Rahmenlinien drumherum: »border«	310
16.1.4	Der Außenabstand »margin« regelt den Abstand zu anderen Boxen	310
16.1.5	Der Unterschied zwischen Abständen mit »padding« und »margin«	311
16.2	Das Box-Modell im Browser visualisieren	312
16.3	Die Breite von Blockboxen begrenzen: »max-width«	313
16.4	Blockboxen zentrieren mit »margin: auto«	314

16.5	Der Abstand zum Rand: »padding«	315
16.5.1	Das »padding« für den Kopfbereich der Seite	316
16.5.2	Das »padding« für die Navigation und den Fußbereich	317
16.5.3	Das »padding« für den Inhaltsbereich	318
16.6	Vertikale Außenabstände und Collapsing Margins	319
16.6.1	Vertikale Außenabstände aufeinanderfolgender Elemente kollabieren	319
16.6.2	Ein Kopfbereich mit Überschrift und überraschende Abstände	320
16.6.3	Nützlich: Eine CSS-Regel zur Vermeidung von Collapsing Margins	322
16.7	Das intuitivere Box-Modell: »box-sizing: border-box«	324
16.7.1	Das Border-Box-Modell in der Übersicht	324
16.7.2	Das Border-Box-Modell aktivieren mit »box-sizing: border-box«	325
16.8	Das Box-Modell für Inline-Boxen	326
16.9	Inline-Block: Blockboxen, aber nebeneinander	327
16.10	Auf einen Blick	328

17 Boxen gestalten per CSS

17.1	Hintergrundgrafiken per CSS einbinden und gestalten	329
17.1.1	Hintergrundgrafiken einbinden: »background-image«	330
17.1.2	Hintergrundgrafiken wiederholen: »background-repeat«	331
17.1.3	Hintergrundgrafiken positionieren: »background-position«	332
17.1.4	Hintergrundgrafiken fixieren: »background-attachment«	333
17.1.5	Die Größe der Hintergrundgrafik definieren: »background-size«	333
17.1.6	Die Kurzschriftweise: »background«	335
17.1.7	Das Box-Modell und die dritte Dimension	335
17.2	Schattenboxen mit »box-shadow«	336
17.3	Abgerundete Ecken mit »border-radius«	338
17.4	Lineare Farbverläufe mit »linear-gradient()«	339
17.5	Gestalten mit dem Box-Modell: Zitate	341
17.5.1	Das HTML: »section« und »blockquote«	341
17.5.2	Zitate gestalten mit »padding«, »border« und »margin«	342
17.6	Links de luxe: Hyperlinks als Button gestalten	344
17.6.1	Die Ausgangssituation: Zwei ganz normale Hyperlinks	345

17.6.2	Schritt 1: Die grundlegende Gestaltung der beiden Links	345
17.6.3	Schritt 2: Die Unterschiede – primäre und sekundäre Buttons	346
17.6.4	Schritt 3: Einen Hover-Effekt mit »transition« animieren	347
17.7	Externe Hyperlinks kennzeichnen mit »::after«	349
17.7.1	Schritt 1: Externe Hyperlinks auswählen mit einem Attributselektor ...	349
17.7.2	Schritt 2: Das Pseudoelement »::after« und die Eigenschaft »content«	350
17.7.3	Schritt 3: Links kennzeichnen mit einem Unicode-Symbol	351
17.8	Boxen am Bildschirm ausblenden: »visually-hidden«	352
17.8.1	Schritt 1: Die Klasse »visually-hidden« erstellen	352
17.8.2	Schritt 2: Den Elementen die Klasse »visually-hidden« zuweisen	354
17.9	Auf einen Blick	355

18 **Ordnung halten: Stylesheets organisieren**

357

18.1	Kommentare zum Strukturieren von Stylesheets	358
18.1.1	Der Kommentar am Anfang des Stylesheets	358
18.1.2	Ein Stylesheet mit Kommentaren in Abschnitte unterteilen	359
18.2	Verschiedene Schreibweisen für CSS-Regeln	359
18.2.1	Weit verbreitet: Eine Anweisung pro Zeile	359
18.2.2	Kurze Regeln: Alles in einer Zeile	360
18.2.3	Übersichtlich: Nur ein gruppierter Selektor pro Zeile	360
18.2.4	Reihenfolge der Anweisungen: 1. Am Box-Modell orientieren	361
18.2.5	Reihenfolge der Anweisungen: 2. Am Alphabet orientieren	362
18.3	Ein zentrales Stylesheet erleichtert die Entwicklung	362
18.3.1	Während der Entwicklung: Modulbauweise mit mehreren Stylesheets	362
18.3.2	Für die Live-Site: Alles in einem Stylesheet vereinen	363
18.4	Die einzelnen Stylesheets erstellen und einbinden	364
18.4.1	Schritt 1: Die einzelnen Stylesheets erstellen	364
18.4.2	Schritt 2: Stylesheets mit @import in »style.css« einbinden	365
18.5	Aufräumen: Das CSS auf die Stylesheets verteilen	366
18.5.1	Allgemeine Einstellungen und grundlegende Gestaltung in »basis.css«	366
18.5.2	Das Seitenlayout und die Inhaltsbereiche: »layout.css«	368

18.5.3	Die Gestaltung der Navigation in »navi-inline.css«	370
18.5.4	Die Gestaltung des Inhalts: »content.css«	371
18.5.5	Die Gestaltung der Formulare: »formulare.css«	372
18.6	CSS überprüfen mit dem CSS-Validator	373
18.7	Ein neues Modul für ein modernes Layout	374
18.7.1	Schritt 1: Das HTML anpassen – die Dopplung mit »div«	375
18.7.2	Schritt 2: Das Stylesheet »layout-modern.css« hinzufügen	377
18.7.3	Schritt 3: Fine-Tuning für die Infoboxen auf der Startseite	379
18.8	Auf einen Blick	380

TEIL IV **CSS – Layout**

19	Media Queries und responsives Webdesign	383
19.1	»Getting responsive«: Das Web wird flexibel	383
19.2	Medientypen definieren das Ausgabemedium	384
19.2.1	Die Medientypen in der Übersicht	384
19.2.2	Eine Druckversion für die Übungswebsite mit »@media print«	385
19.3	Media Queries = Medientypen + Medieneigenschaften	387
19.3.1	Die Syntax: »@media Medientyp and (Eigenschaft: Wert)«	387
19.3.2	Ein Beispiel: Eine Media Query zur Änderung der Hintergrundfarbe	388
19.3.3	Weitere Beispiele für Media Queries	389
19.3.4	Die wichtigsten Medieneigenschaften im Überblick	390
19.4	Media Queries und der »Meta-Viewport«	391
19.4.1	Media Queries funktionieren in mobilen Browsern manchmal nicht	391
19.4.2	Der Meta-Viewport definiert die Viewportbreite für mobile Browser neu	392
19.5	Media Queries und der richtige Breakpoint	393
19.5.1	Weit verbreitet: Breakpoints für Mobil, Tablet und Desktop	393
19.5.2	Breakpoints sollte man vom Layout ableiten, nicht von Geräten	394
19.6	Mehrspaltiger Text mit dem »Multi-column Layout«	394
19.7	Auf einen Blick	396

20	Der Flow und die Eigenschaft »position«	399
20.1	Der normale Flow mit »position: static«	399
20.2	Versetzt weiterfließen mit »position: relative«	401
20.3	Raus aus dem Flow mit »position: absolute«	402
20.4	Der Trick: »absolute« und »relative« kombinieren	403
20.5	Wie ein Fels in der Brandung: »position: fixed«	405
20.5.1	In einem schmalen Viewport: Icons nebeneinander, unter dem Text ...	406
20.5.2	In einem breiteren Viewport: Icons untereinander, neben dem Text	407
20.6	Scrollen und dann stehen bleiben: »position: sticky«	408
20.7	Positionierte Boxen und der »z-index«	409
20.8	Auf einen Blick	412
21	Schweben und schweben lassen: »float«	413
21.1	Text um eine Abbildung fließen lassen mit »float«	413
21.1.1	Die Ausgangssituation: ein <figure> mit Bild und Beschriftung	413
21.1.2	Das <figure>-Element nach rechts floaten mit »float: right«	414
21.1.3	Gefloatete Boxen in einem schmalen Viewport überprüfen	416
21.1.4	Die umgebenden Elemente reichen bis unter die gefloatete Box	416
21.2	FLOATS beenden mit »clear: both«	417
21.3	FLOATS umschließen mit »display: flow-root«	417
21.3.1	Das Problem: Floats ragen nach unten aus dem Elternelement heraus	418
21.3.2	Die Lösung: Floats umschließen mit »display: flow-root«	418
21.4	Praktisch: Klassen zum Floaten und Clearen	419
21.5	Das Umschließen von Floats mit »@supports«	420
21.6	Auf einen Blick	422

22	Gestalten per Flexbox: Das »CSS Flexible Box Layout«	423
22.1	Flexbox und Grid – das neue CSS-Layout	423
22.1.1	Der »Block Formatting Context« ist für Layouts nur bedingt geeignet	424
22.1.2	Jenseits vom »Block Formatting Context«: Flexbox und Grid	424
22.1.3	Layouthilfen per Flexbox: Neue Möglichkeiten ohne alte Probleme	424
22.2	Los geht's: Flex-Container erstellen mit »display: flex«	425
22.2.1	Die Ausgangsposition – eine einfache Navigation	425
22.2.2	Eine Flexbox definieren mit »display: flex«	426
22.3	Fließrichtung: Die Richtung ändern mit »flex-flow«	428
22.3.1	Jede Flexbox hat eine Hauptachse und eine Querachse	428
22.3.2	»flex-direction« ändert die Fließrichtung: von »row« zu »column«	428
22.3.3	»flex-wrap« ermöglicht einen Zeilenumbruch in der Flexbox	429
22.3.4	»flex-flow« ist kurz für »flex-direction« und »flex-wrap«	430
22.4	Ausrichtung: Leerraum verteilen mit »justify-content«	431
22.5	Ausrichtung: Automatische Abstände mit »margin«	432
22.5.1	Flex-Items am Ende des Flex-Containers ausrichten mit »margin«	433
22.5.2	Elemente horizontal und vertikal zentrieren mit »margin: auto«	433
22.6	Flexibilität: Die Zauberformel »flex: 1«	434
22.6.1	»Lieber Browser, bitte mach alle Flex-Items gleich groß.«	434
22.6.2	»flex« ist kurz für »flex-grow«, »flex-shrink« und »flex-basis«	435
22.6.3	Überraschung: »flex-grow« in einer mehrzeiligen Flexbox	436
22.7	Flexbox in Aktion: Den Footer platzieren	437
22.7.1	Schritt 1: <body> wird Flex-Container, die Layoutbereiche Flex-Items	437
22.7.2	Schritt 2: Die Zauberformel »flex: 1« für den Inhaltsbereich	438
22.8	Die Reihenfolge der Flex-Items ändern	440
22.9	Auf einen Blick	441

23 Eine responsive Navigation erstellen	443
23.1 Die responsive Navigation im Überblick	443
23.2 Schritt 1: Grundlegende Formatierung der Navigation	445
23.2.1 Neues Stylesheet erstellen und einbinden	445
23.2.2 Die grundlegende Formatierung der Navigation für schmale Viewports	445
23.3 Schritt 2: Den Menübutton im Quelltext erstellen	447
23.4 Schritt 3: Den Menübutton per CSS gestalten	449
23.4.1 Den Menübutton per CSS gestalten	449
23.4.2 Eine Grafik für den Menübutton mit »::before« hinzufügen	450
23.5 Schritt 4: Die Navigationsliste per CSS ausblenden	452
23.6 Schritt 5: Die Navigationsliste per CSS einblenden	453
23.7 Schritt 6: Media Query und horizontale Navigation	455
23.8 Die Meta-Navigation im Fußbereich gestalten	456
23.8.1 Ein neues Stylesheet »navi-meta.css« erstellen und einbinden	457
23.8.2 Die Navigation im Fußbereich gestalten	457
23.9 Auf einen Blick	458
24 Gestalten mit Raster: Das »CSS Grid Layout«	459
24.1 Ein Grid ist ein Raster und schafft Ordnung	459
24.2 Mehrspaltiges Layout nur für moderne Browser: »@supports«	460
24.3 »Let's grid«: Drei Infoboxen nebeneinander	461
24.3.1 Ein Blick auf das HTML für den Abschnitt mit den Infoboxen	461
24.3.2 Schritt 1: Einen Grid-Container definieren mit »display: grid«	462
24.3.3 Schritt 2: Ein Raster definieren mit »grid-template-columns« und »fr«	463
24.3.4 Schritt 3: Den Zwischenraum kontrollieren mit »grid-gap«	465
24.4 Grid-Items mit nummerierten Linien platzieren	467
24.4.1 Das HTML für den Abschnitt mit den Kundenstimmen	467
24.4.2 Einen Grid-Container definieren und das Raster erstellen	468
24.4.3 Grid-Items manuell auf dem Raster platzieren mit »grid-column«	469

24.5 Praktisch: Ein Grid mit benannten Bereichen	471
24.5.1 Die HTML-Struktur für den Inhaltsbereich der Seite »News«	472
24.5.2 Einen Grid-Container definieren und das Raster erstellen	472
24.6 Die Grid-Zauberformel: Responsiv ohne Media Query	474
24.6.1 Die Ausgangsposition: HTML und CSS für die Teamvorstellung	475
24.6.2 Schritt 1: »repeat()« erzeugt mit »auto-fit« beliebig viele Spalten	477
24.6.3 Schritt 2: Die Funktion »minmax()« macht das responsive Grid perfekt	478
24.7 Auf einen Blick	480
25 Flexible Icons und responsive Bilder	483
25.1 Flexible Icons: Skalierbare Symbole mit SVG	483
25.1.1 Fertige SVG-Icons herunterladen und einbinden	484
25.1.2 Möglichkeit 1: SVG-Icons als Datei einbinden mit 	485
25.1.3 Eine SVG-Datei im Editor öffnen: SVG ist Markup. Wie HTML	487
25.1.4 Möglichkeit 2: SVG-Icon inline direkt im HTML-Quelltext einbinden ...	488
25.2 Pixelbilder und hochauflösende Bildschirme	490
25.2.1 DPR: Das Verhältnis von Gerätelpixeln zu logischen Pixeln	490
25.2.2 Die einfache Lösung: Eine doppelt so große Grafik einbinden	491
25.3 Unterschiedliche Bilder je nach Pixeldichte	491
25.4 Unterschiedliche Bilder je nach Viewportbreite	493
25.4.1 Tausche X gegen W: , »srcset w« und »sizes«	493
25.4.2 Das Attribut »sizes« kann die Breite des Viewports abfragen	495
25.5 Auf einen Blick	497
Index	499