Einstieg in HTML und CSS

Webseiten programmieren und gestalten

DIE LESEPROBE

Kapitel 2
HTML kennenlernen: die erste Webseite erstellen

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

Die Themen im Überblick:

- Webseiten bestehen aus rechteckigen Kästchen
- HT-M-L: die »HyperText Markup Language«
- Die erste Webseite erstellen: »index.html«
- Jede Webseite hat ein HTML-Grundgerüst
- Der <!doctype> und das Stammelement <html>
- HTML-Elemente können im Anfangs-Tag Attribute enthalten
- <head> enthält wichtige Infos über die Webseite
- <body> enthält den im Browser sichtbaren Bereich der Webseite
- Der Kopfbereich <header> mit Überschrift und Slogan
- Entwicklerwerkzeuge: HTML im Browser untersuchen
- Auf einen Blick

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 Übungswebsite, die im Laufe des Buches Schritt für Schritt weiterentwickelt wird.

2.1 Webseiten bestehen aus rechteckigen Kästchen

Webseiten bestehen am Bildschirm aus rechteckigen Kästchen, die im Browserfenster übereinander, nebeneinander und ineinander gestapelt werden und auf Englisch *box* genannt werden. Webseiten bestehen also aus lauter *little boxes*.

Abbildung 2.1 zeigt die Startseite der Übungswebsite am Ende des Buches, wobei die rechteckigen Kästchen mit einer Rahmenlinie sichtbar gemacht wurden.

Abbildung 2.1 Die Startseite der Übungswebsite mit sichtbaren Kästchen

Everything is a box. Je eher Sie sich an den Gedanken gewöhnen, dass Webseiten aus Rechtecken bestehen, desto leichter wird Ihnen das Gestalten von Webseiten fallen.

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

- HTML-Elemente strukturieren die Webseite und erstellen die Kästchen.
- CSS-Regeln gestalten die Kästchen und deren Inhalte.

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

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.

2.2.1 HT wie »Hypertext«: Hyperlinks erstellen

Hypertext ist Text mit *Hyperlinks*. Das World Wide Web besteht aus Milliarden von Webseiten, die durch Hyperlinks 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* besagt, dass man mit HTML Hyperlinks erstellen kann.

2.2.2 M wie »Markup«: Etiketten kleben

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. Diese Etiketten kleben Sie quasi in den Text, damit die Browser wissen, worum es sich dabei handelt:

```
<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 Referenzen zum Nachschlagen. Einige nützliche Links finden Sie in Abschnitt 1.7, »Websites zum Nachschlagen von HTML und CSS«.

2.2.4 Der Unterschied zwischen »HTML-Elementen« und »HTML-Tags«

Da die Begriffe *Elemente* und *Tags* im HTML-Alltag häufig Verwirrung stiften, möchte ich den Unterschied kurz erläutern.

HTML besteht aus Elementen, und die Namen dieser 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.

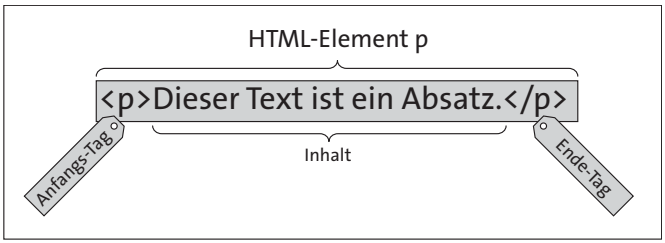

Abbildung 2.2 Ein Element besteht aus Anfangs-Tag, Inhalt und Ende-Tag.

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 ein *Element*. Die *Tags* stehen in spitzen Klammern, das *Element* heißt aber 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 Die erste Webseite erstellen: »index.html«

In diesem Abschnitt erstellen Sie 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 immer 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, und sie haben zwei Funktionen:

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

In HTML sieht ein Kommentar etwas seltsam aus. Er beginnt mit `<!--` (kleiner als, Ausrufezeichen und zwei Bindestriche) und endet mit `-->` (zwei Bindestriche und größer als):

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

Listing 2.1 Beispiel für einen HTML-Kommentar

Wenn der Browser die Zeichenfolge `<!--` sieht, weiß er, dass ein Kommentar anfängt und er den Text bis zum Kommentarende `-->` 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.

2.4 Jede Webseite hat ein HTML-Grundgerüst

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` enthält wichtige Infos über die Webseite.
4. `body` enthält den im Browser sichtbaren Bereich der Webseite.

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">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Startseite - Einstieg in HTML + CSS</title>
    <meta name="description" content="Beschreibung für diese Webseite">
  </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 in Abschnitt 2.3 erstellte Datei *index.html* im Editor.
2. Geben Sie das in Listing 2.2 gezeigte HTML-Grundgerüst ein.
3. Speichern Sie die Datei im Editor.
4. Öffnen Sie die Datei 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 Anzeigebereich des Browserfensters, dem sogenannten *Viewport*.

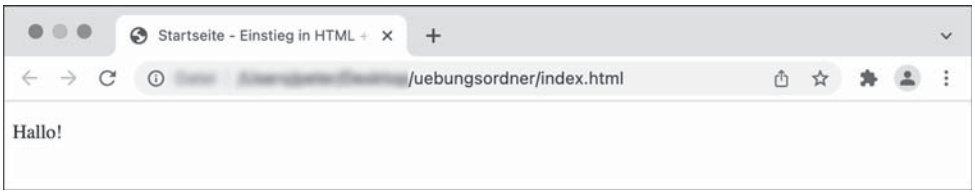

Abbildung 2.3 Die Startseite mit <title> im Tab und <p> im Browserfenster

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.

Sie sollten den Quelltext möglichst übersichtlich schreiben

Es ist empfehlenswert, den Quelltext so wie in Listing 2.2 gezeigt möglichst übersichtlich zu schreiben. Menschen hilft das beim Verstehen des Codes, und unübersichtlich wird er ganz von allein.

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

2.5 Der <!doctype> und das Stammelement <html>

Das in Listing 2.2 gezeigte Grundgerüst beginnt mit dem `doctype` und dem Stammelement `html`.

2.5.1 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.

2.5.2 Das Stammelement: <html> und </html> umschließen den Quelltext

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

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

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

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

2.6 HTML-Elemente können im Anfangs-Tag Attribute enthalten

Das Stammelement `html` enthält in Listing 2.4 im Anfangs-Tag noch eine zusätzliche Angabe:

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

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

Der Zusatz `lang="de"` definiert 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 HTML-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.
- Zwischen dem Attributnamen, dem Gleichheitszeichen, den Anführungsstrichen und dem 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 dabei 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 könnte Browsern bei der korrekten Silbentrennung und Screenreadern bei der korrekten Aussprache helfen.

Kapitel 3
CSS kennenlernen: die erste Webseite gestalten

Worin Sie CSS kennenlernen, die bisher erstellten HTML-Elemente gestalten und sehen, wie man CSS im Browser-Entwicklertool untersuchen kann.

Die Themen im Überblick:

- Jeder Browser hat ein eingebautes Stylesheet
- HTML-Elemente als rechteckige Kästchen visualisieren
- Das erste eigene Stylesheet: »style.css«
- Die erste eigene CSS-Regel: Hintergrund- und Schriftfarbe für <body>
- Den Kopfbereich <header> im CSS selektieren und gestalten
- Wichtige Vokabeln: der Aufbau einer CSS-Regel
- Entwicklerwerkzeuge: CSS im Browser untersuchen
- Auf einen Blick

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 eingebautes Stylesheet

Abbildung 3.1 zeigt *index.html* mit geöffneten Entwicklertools in Chrome:

1. Öffnen Sie die Startseite *index.html* in Chrome.
2. Klicken Sie im Browserfenster mit der rechten Maustaste auf die Überschrift »HTML + CSS«.
3. Wählen Sie im Kontextmenü den Befehl Untersuchen.

Wenn Sie mit dem Mauszeiger links im HTML-Bereich auf das Element `h1` zeigen, wird es auf der Webseite oben im Browserfenster hervorgehoben.

HTML-Elemente dienen zur Strukturierung von Webseiten und haben kein Aussehen, aber die Startseite im Browser ist durchaus ein bisschen gestaltet:

- Die Seite hat einen weißen Hintergrund und schwarze Schrift.
- Die Überschrift ist fett und hat eine Schriftgröße von 32 px.
- Der Absatz darunter hat eine Schriftgröße von 16 px.
- Überschrift und Absatz haben oben und unten etwas Abstand.

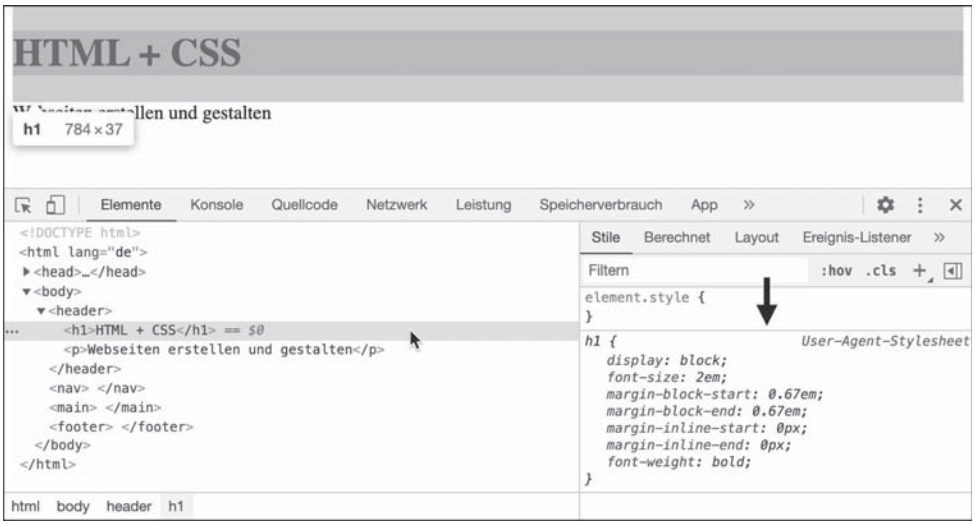

Abbildung 3.1 »index.html« in Chrome mit geöffnetem Entwicklerwerkzeug

Verantwortlich für diese grundlegende Gestaltung ist eine Mischung aus Browsereinstellungen und einem fest eingebauten Browser-Stylesheet.

In Abbildung 3.1 zeigt Chrome im Register STILE das zur Gestaltung der Überschrift verwendete CSS und gibt als Quelle das *User Agent Stylesheet* an, und damit ist das Browser-Stylesheet gemeint.

Machen Sie sich momentan nicht allzu viele Gedanken über die genaue Syntax, aber man erkennt die Gestaltung der Überschrift wieder:

- `display: block` bewirkt, dass die Überschrift die gesamte Zeile *blockt* und sich von ganz links bis ganz rechts erstreckt. Mehr dazu erfahren Sie in Abschnitt 4.6, »Über Blockelemente, Inline-Elemente und ›display‹«.

- `font-size: 2em` definiert die Schriftgröße, und *2em* sind in diesem Fall 32 px. Einheiten wie *em* lernen Sie in Abschnitt 12.4, »Die wichtigsten Längeneinheiten: px, em, rem, % & Co«, kennen, die Gestaltung von Text kommt dann in Kapitel 15.
- Die `margin`-Zeilen gestalten die Außenabstände der Kästchen. Mehr dazu erfahren Sie in Abschnitt 12.2 beim Kennenlernen des Box-Modells.
- `font-weight: bold` schließlich macht die Überschrift fett.

Vereinfacht gesagt: Wenn der Browser eine `h1`-Überschrift sieht, denkt er: »Mmmh, das ist eine wichtige Überschrift, und hier steht nirgendwo, wie genau die aussehen soll. Also schreibe ich den Text mal auf eine eigene Zeile und mach ihn groß und fett.«

Das Browser-Stylesheet sorgt dafür, dass HTML-Elemente ohne weitere Gestaltung im Browserfenster lesbar sind. Von Ihnen definierte CSS-Regeln überschreiben das Browser-Stylesheet, aber für alle Elemente und Eigenschaften, die Sie *nicht* selbst gestalten, gelten weiterhin die Vorgaben vom Browser.

3.2 HTML-Elemente als rechteckige Kästchen visualisieren

»Kenne dein HTML« ist das oberste Motto beim Gestalten von Webseiten, denn wenn man die HTML-Struktur nicht kennt, 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 class="startseite">
  <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 (engl. *box*) dargestellt, und besonders am Anfang ist es manchmal hilfreich, sich die Struktur des Quelltextes mit einer einfachen Zeichnung zu verdeutlichen. Visualisiert sieht der Quelltext aus Listing 3.1 so aus wie in Abbildung 3.2.

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

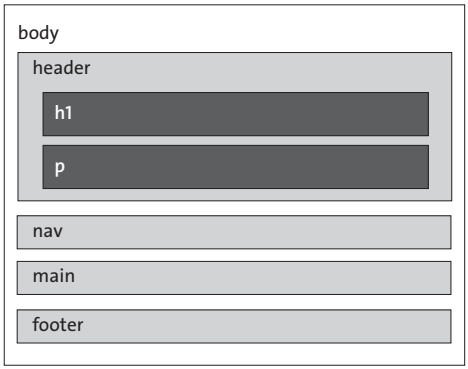

Abbildung 3.2 Die Elemente im Quelltext als rechteckige Kästchen

Die Namen der CSS-Eigenschaften sind englische Begriffe in amerikanischer Schreibweise. So heißt die Eigenschaft zur Gestaltung der Schriftfarbe `color` und nicht wie im britischen Englisch *colour*.

In diesem Kapitel gestalten Sie die beiden Kästchen für `body` und `header` inklusive Überschrift und Absatz. Die Bereiche `nav`, `main` und `footer` enthalten keinen Inhalt und sind daher im Browser nicht sichtbar, 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 Beispielseite *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 erstellen

1. Wechseln Sie im Finder oder Explorer in den Übungsordner, in dem Sie die Startseite *index.html* gespeichert haben.
2. Erstellen Sie im selben Ordner 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 zwingend 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 das Webseite und Stylesheet miteinander verbindet:

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Startseite - Einstieg in HTML + CSS</title>
  <meta name="description" content="Beschreibung für diese Webseite">

  <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. Im Beispiel liegt *style.css* 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>` eine leere Zeile ein.
3. Fügen Sie dort wie in Listing 3.2 gezeigt das `link`-Element zur Verknüpfung von *index.html* mit dem Stylesheet *style.css* ein.
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 eigene CSS-Regel: Hintergrund- und Schriftfarbe 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 Auskommentieren (Ausblenden via Kommentar) 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 ⇧ + 7 bzw. ⇧ + +. Auf dem Ziffernblock geht es noch einfacher:

- Den Schrägstrich / finden Sie auf der Taste für »geteilt durch« (Division).
- Das Sternchen * ist die Taste mit dem 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 Ihrer Wahl.
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.

Am Bildschirm werden alle Farben aus Licht in den Farben Rot, Grün und Blau gemischt. Zur Definition der jeweiligen Farbanteile gibt es in CSS diverse Möglichkeiten, aber für den Einstieg verwenden Sie Farbnamen wie `black`, `white` oder `whitesmoke`. Das hat den Vorteil, dass man vorab nichts erklären muss, denn zumindest die Grundfarben sind auch auf Englisch verständlich. In Abschnitt 12.3, »Farben in CSS: Farbnamen, hexadezimale Schreibweise und Transparenz«, lernen Sie weitere Möglichkeiten zur Definition von Farbwerten kennen.

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

Geschweifte Klammern finden Sie auf deutschen Tastaturlayouts so:

- unter Windows mit Alt + 7 bzw. Alt + 9
- unter macOS mit ⌥ + 8 bzw. ⌥ + 9

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`, als Farbwert wird `floralwhite` verwendet. Blütenweiß.
- Die Eigenschaft für die Schriftfarbe heißt `color`, und *schwarz* wird die Schrift mit `black`.

Die Reihenfolge der Zeilen zwischen den geschweiften Klammern spielt in diesem Beispiel 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: Hintergrund- und Schriftfarbe für <body>

1. Öffnen Sie das Stylesheet *style.css* im Editor.
2. Fügen Sie nun unterhalb des Kommentars am Anfang der Datei die CSS-Regel aus Listing 3.4 ein.
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.

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

Spielen Sie ein bisschen mit den Farben

Probieren Sie auch einmal andere Farbkombinationen aus. Im Web gibt es einige Referenzen für Farbnamen:

- *w3schools.com/colors/colors_names.asp*
- *wiki.selfhtml.org/wiki/Grafik/Farbe/Farbpaletten#Farbnamen*

Im Augenblick sollten Sie einfach nur darauf achten, dass der Kontrast zwischen Hintergrund- und Schriftfarbe groß genug ist. Viel Spaß dabei.

3.5 Den Kopfbereich <header> im CSS selektieren und gestalten

In diesem Abschnitt soll der Kopfbereich zu Übungszwecken einen dunkleren Hintergrund und eine weiße Schrift bekommen. Anschließend fügen Sie zwischen Text und dem Rand der Box noch ein bisschen Abstand hinzu.

3.5.1 Hintergrund- und Schriftfarbe für <header> ändern

Das HTML-Element für den Selektor heißt `header`, und Listing 3.5 zeigt die komplette CSS-Regel zu dessen Gestaltung auf einen Blick:

```
header {
  background-color: steelblue;
  color: white;
}
```

Listing 3.5 Hintergrund- und Schriftfarbe für den Kopfbereich gestalten

Im folgenden Kasten speichern Sie diese Regel im Stylesheet.

Kapitel 6
HTML-Elemente für Bilder, Audio und Video

Worin Sie HTML-Elemente zur Einbindung von Bildern, Audiodateien und Videos kennenlernen und diese mit wenigen CSS-Regeln flexibilisieren.

Die Themen im Überblick:

- Über Grafikformate im Web: JPEG, GIF, PNG, SVG & Co
- Ein Bild als Logo einbinden mit <img>
- Pixelbilder in Zeiten hochauflösender Bildschirme
- Bilder mit flexibler Breite: »max-width: 100%«
- Abbildungen beschriften: <figure> und <figcaption>
- »Lazy Loading«: Seiten mit vielen Bildern optimieren
- Audiodateien einbinden mit <audio>
- Videodateien einbinden mit <video>
- Auf einen Blick

In diesem Kapitel sehen Sie, wie man Pixelbilder auf Webseiten einfügt, für hochauflösende Bildschirme optimiert und sie flexibel darstellt und beschriftet. Danach lernen Sie die HTML-Elemente zum Einfügen von Audio- und Videodateien auf Webseiten kennen.

6.1 Über Grafikformate im Web: JPEG, GIF, PNG, SVG & Co

Ein Bild sagt nicht nur mehr als tausend Worte, im Web lädt es oft auch länger, denn jedes Kilobyte muss vom Webserver 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 Farbflächen, Logos oder Fotos mit Freistellungen vor einem Hintergrund. Für normale Fotos ist JPEG meist 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* mit der Endung *.svg*, das *Vektorgrafiken* bereitstellt. SVG-Dateien enthalten also keine Pixel, sondern 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 23.1, »Flexible Icons: skalierbare Symbole mit SVG«, lernen Sie das Format näher kennen.

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

Falls Sie ein gutes Tool zum Komprimieren von Grafikdateien suchen, ist *compressor-die.com* von Christoph Erdmann eine Empfehlung, die besonders bei JPGs tolle Ergebnisse erzielt. Die Website *squoosh.app* stammt von den Google Chrome Labs und kann Pixelbilder in so ziemlich alle bestehenden Formate konvertieren, inklusive AVIF.

Es gibt neue Grafikformate wie WebP oder AVIF

Es gibt relativ neue Grafikformate wie *WebP* (*webpi* gesprochen) und besonders *AVIF*, das die Vorteile von JPG, PNG und GIF in einem Format vereint:

- *de.wikipedia.org/wiki/WebP*
- *de.wikipedia.org/wiki/AV1_Image_File_Format*

Der Haken ist, dass ältere Browser die neuen Formate nicht unterstützen. Eine Lösung bietet das in Abschnitt 23.6 vorgestellte `picture`-Element, mit dem man unter anderem eine Grafik in mehreren Formaten ausliefern kann.

6.2 Ein Bild als Logo einbinden mit <img>

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 <img> und seine wichtigsten Attribute

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

Das folgende Listing zeigt ein Beispiel:

```
<img src="bilddatei.jpg" alt="Alternativer Text" width="400" height="300">
```

Listing 6.1 Das Element <img> 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. Für die Barrierefreiheit ist ein alternativer Text wichtig, denn Screenreader lesen ihn zur Beschreibung des Bildes vor. Eine Entscheidungshilfe zum Schreiben von alternativen Texten für Bilder finden Sie im Hinweiskasten etwas weiter unten. Auch die Suchmaschinen werten den Alt-Text zur Beschreibung eines Bildes aus.
- `width=""` und `height=""`
  Die Attribute `width` und `height` teilen dem Browser mit, wie groß die Grafik dargestellt werden soll. So kann der Browser beim Erstellen der Webseite den Platz für das Bild schon einplanen, *bevor* er die Datei selbst überhaupt erhalten hat.

Das Element `img` erzeugt im Browserfenster keinen Zeilenumbruch, sondern fließt wie ein Inline-Element einfach in der Zeile. `img` ist ein sogenanntes *ersetztes Element* (*replaced element*), denn das `<img>`-Tag wird durch die Grafikdatei ersetzt.

Entscheidungshilfe beim Schreiben von alternativem Text für Bilder

Eine Textalternative für Bilder ist wichtig für die Barrierefreiheit, und zur Erleichterung der Arbeit stellt die *Web Accessibility Initiative* (WAI) eine kleine Entscheidungshilfe bereit:

- *w3.org/WAI/tutorials/images/decision-tree/*

Die Informationen auf dieser Seite sind zum Schreiben von alternativen Texten für verschiedene Arten von Bildern eine echte Hilfe.

6.2.2 Ein Logo auf der Übungswebsite einfügen mit <img>

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, die sie in den Übungsdateien im Ordner *medienlager* finden.
- Diese Datei sollten Sie im Übungsordner im Unterordner *bilder* einfügen.
- Der alternative Text enthält einfach den in der Grafik dargestellten Text *HTML und CSS*. So wird es in der Entscheidungshilfe des WAI zum Schreiben von alternativen Bildern empfohlen (siehe Kasten weiter oben).
- Die Logo-Datei hat eine Breite von 222 Pixel und eine Höhe von 36 Pixel. Mithilfe der Attribute `width` und `height` teilen Sie dem Browser diese Abmessungen mit.

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>
  <img src="bilder/html-und-css-logo-222.png"
       alt="HTML und CSS"
       width="222"
       height="36">
</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-und-css-logo-222.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 Beispielseite sieht jetzt im Browser ungefähr so aus wie in Abbildung 6.1.

Abbildung 6.1 Die Übungswebsite mit einem Bild als Logo

6.2.3 Die Abstände zwischen Logo und dem Absatz darunter anpassen

Der Abstand zwischen dem Logo in der `h1`-Überschrift und dem Absatz ist mit den Vorgaben vom Browser-Stylesheet sehr groß, und deshalb überschreiben Sie diese mit eigenem CSS. Listing 6.3 enthält dazu zwei einfache Regeln:

```
/* Abstand zwischen Logo und dem folgenden Absatz anpassen */
header h1 {
  margin-bottom: 0;
}
header h1 + p {
  margin-top: 0;
}
```

Listing 6.3 Außenabstände von Logo und Slogan anpassen

Die erste Regel selektiert die `h1`-Überschrift im Kopfbereich und entfernt mit der Eigenschaft `margin-bottom` den Außenabstand nach unten, die zweite wählt den auf die Überschrift folgenden Absatz 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 die CSS-Regeln aus Listing 6.3, am besten nach dem einleitenden Kommentar und vor der grundlegenden Gestaltung des Navigationsbereichs.
3. Speichern Sie das Stylesheet, und betrachten Sie die Webseite in einem Browser.

Abbildung 6.2 zeigt, dass der Abstand zwischen Logo und dem folgenden Absatz sich geändert hat, sodass der Header jetzt etwas kompakter aussieht.

Abbildung 6.2 Logo und Slogan mit angepassten Außenabständen

Mehr zur Gestaltung des Kopfbereiches und zu »margin«

In Abschnitt 7.2 präzisieren Sie den Selektor zur Gestaltung des Kopfbereiches mit einer Klasse und in Kapitel 14, »Die wichtigsten Selektoren und ihre Spezifität«, werden Selektoren wie `h1 + p` ausführlich vorgestellt. Die Eigenschaft `margin` wird in Abschnitt 12.2, »Das Box-Modell kennenlernen: ›padding‹, ›border‹ und ›margin‹«, näher erläutert.

6.3 Pixelbilder in Zeiten hochauflösender Bildschirme

Vor einigen Jahren wäre das zum Einbinden von Bildern auf Webseiten bereits alles gewesen, aber inzwischen ist die Sache nicht mehr ganz so einfach, denn viele moderne Smartphones, Tablets und auch immer mehr Computer haben sogenannte *hochauflösende Bildschirme*. Apple nennt sie *Retina*, bei anderen Herstellern haben sie andere Bezeichnungen.

Diese Bildschirme nutzen zur Darstellung eines Bildpixels gleich mehrere Gerätepixel und erreichen dadurch eine so hohe Pixeldichte, dass die Netzhaut des Auges bei einem normalen Betrachtungsabstand keine einzelnen Pixel mehr erkennen kann.

Während Schrift und Vektorgrafiken auf diesen Bildschirmen gestochen scharf wirken, muss man bei Pixelbildern aufpassen, dass sie nicht unscharf werden.

6.3.1 Das Problem: Das Logo ist auf hochauflösenden Bildschirmen unscharf

Das im vorherigen Abschnitt eingebundene Logo hat eine Abmessung von 222 × 36 Pixel und wird im Browser mit genau dieser Breite und Höhe dargestellt (siehe Listing 6.2). Auf traditionellen Displays passt das genau, aber auf hochauflösenden Bildschirmen hätten die Browser zur Darstellung der Grafik eigentlich viel mehr Pixel nötig. Da diese nicht vorhanden sind, vergrößert der Browser einfach die vorhandenen Pixel, wodurch die Bilder unscharf werden (Abbildung 6.3):

- Oben sieht das Logo gut aus (traditioneller Bildschirm).
- Unten ist es leicht verschwommen (hochauflösender Bildschirm).

Abbildung 6.3 Das Logo – oben scharf, unten leicht verschwommen

DPR: Das Verhältnis von Gerätepixeln zu logischen Pixeln

Das Verhältnis zwischen logischen Bildpixeln und physischen Gerätepixeln wird als *DPR* (*device-pixel-ratio*) bezeichnet. Hochauflösende Bildschirme haben eine DPR von 2 oder

mehr. Falls Sie die DPR eines Bildschirms nicht kennen, betrachten Sie mit dem Gerät einfach die folgende Webseite in einem Browser:

- *mydevice.io*

Im Bereich SCREEN METRICS finden Sie Infos zum Bildschirm des Geräts. Die DPR wird hier als CSS PIXEL-RATIO bezeichnet.

6.3.2 Einfache Lösung: Eine doppelt so große Grafik einbinden

Im Alltag nutzen viele Webseitenbauer einen simplen, aber effektiven Trick und verwenden einfach eine doppelt so große Grafik: Damit das im Browserfenster 222 × 36 Pixel große Logo auf hochauflösenden Bildschirmen scharf bleibt, wird im HTML eine doppelt so große Grafikdatei eingebunden (444 × 72):

```
<img src="bilder/html+css-logo-444.png"
     alt="HTML + CSS"
     width="222"
     height="36">
```

Listing 6.4 Ein Bild als Logo in einer Überschrift

Dieser Trick funktioniert frei nach dem Motto »One size fits all« im Alltag, ist aber natürlich genau genommen gemogelt, denn traditionelle Bildschirme bekommen ein zu großes Bild. Tabelle 6.1 zeigt, dass bei einer größeren Grafik auch mehr Kilobyte übertragen werden.

Dateiname	Breite	Höhe	Dateigröße
html-und-css-logo-222.png	222 px	36 px	2,16 kb
html-und-css-logo-444.png	444 px	72 px	3,76 kb

Tabelle 6.1 Die Daten für die beiden Logo-Dateien auf einen Blick

Bei einem kleinen Logo ist der Unterschied in der Dateigröße mit gut 1,5 kb nicht dramatisch, aber bei größeren Fotos sind es oft viele Hundert unnötigerweise übertragene Kilobyte. Im folgenden Abschnitt zeige ich Ihnen daher, wie man dem Browser je nach Pixeldichte des Bildschirms eine passende Bilddatei anbieten kann.

6.3.3 Besser: Je nach Pixeldichte unterschiedliche Dateien einbinden

Idealerweise laden die Browser je nach Pixeldichte des Bildschirms eine passende Datei. Dazu benötigen Sie zwei Grafikdateien und erweitern das `img`-Element um das Attribut `srcset`, mit dem Sie dem Browser die zweite Grafik anbieten. Für das Logo auf der Übungswebsite sieht die Lösung im Quelltext so aus wie im folgenden Listing:

```
<img src="bilder/html-und-css-logo-222.png"
     srcset="bilder/html-und-css-logo-444.png 2x"
     alt="HTML und CSS"
     width="222"
     height="36">
```

Listing 6.5 Ein zweites Bild für hochauflösende Bildschirme

Auf hochauflösenden Bildschirmen wird die 444 px breite Grafik im Browser durch die Angabe von `width` mit einer Breite von 222 px dargestellt, und das Logo ist dadurch scharf. Dieses Listing funktioniert so:

- Auf traditionellen Bildschirmen nutzt der Browser die als Wert für das Attribut `src` angegebene Grafik *html-und-css-logo-222.png*.
- Das Attribut `srcset` bietet dem Browser mit *html-und-css-logo-444.png* eine zweite Grafikdatei an.
- Der `x`-Wert hinter dem Dateinamen steht für die DPR des Bildschirms.

Im Folgenden setzen Sie diese Lösung für die Übungswebsite um.

Übungswebsite: Je nach Pixeldichte ein anderes Logo ausliefern

1. Kopieren Sie die Datei *html-und-css-logo-444.png* in den Unterordner *bilder*, sodass dort beide Logo-Dateien liegen.
2. Suchen Sie im Quelltext das Element `img` zur Einbindung des Logos.
3. Erweitern Sie wie in Listing 6.5 gezeigt das HTML für `img`.
4. Speichern Sie die Seite, und betrachten Sie sie in einem Browser.

Nach diesen Schritten bekommen nur die hochauflösenden Bildschirme die große Datei:

1. Der Browser weiß, welche Pixeldichte der Bildschirm hat.
2. Er schaut im Quelltext, welche Dateien zur Verfügung stehen, und holt nur die passende Datei vom Server:
   - Bei DPR 1 wird *html-und-css-logo-222.png* verwendet.
   - Bei DPR 2 oder mehr ist *html-und-css-logo-444.png* dran.

Fazit: Mit `img` und `srcset` mit `x`-Wert können Sie bestimmte Dateien nur an hochauflösende Bildschirme schicken. Die Syntax ist relativ leicht zu verstehen und die Mehrarbeit überschaubar: Die Grafiken müssen in zwei Versionen bereitgestellt und die `img`-Elemente angepasst werden. Für Logos und andere Grafikdateien mit einer festen Breite ist diese Lösung optimal und besser als das Bereitstellen einer eigentlich zu großen Grafik.

Eine Optimierung für mehr als DPR 2 ist meist nicht nötig

Viele Geräte haben Bildschirme mit einer DPR 3 oder sogar 4. Sollte man also jetzt von jedem Bild gleich drei oder vier verschiedene Versionen ausliefern?

Kurze Antwort: Nein. Jenseits von `2x` kann das menschliche Auge oft keinen qualitativen Unterschied mehr feststellen. Ausführlichere Antwort:

pmueller.de/bilder-optimieren-dpr-2-ist-meist-genug/

6.3.4 Testen: Die korrekte Einbindung der Grafiken im Browser überprüfen

Mit den Entwicklertools in Chrome können Sie prüfen, ob bei der Einbindung des Logos alles funktioniert hat:

1. Öffnen Sie die Übungsdatei *index.html* in Chrome.
2. Aktivieren Sie das Entwicklertool, zum Beispiel mit F12, und markieren Sie das `img`-Element mit dem Logo.
3. Blenden Sie mit einem Klick auf das Symbol Geräte-Symbolleiste ein- und ausblenden (zweites Symbol von links in der Menüleiste der Entwicklertools) die Geräte-Symbolleiste ein.
4. Klicken Sie oben in der Geräte-Symbolleiste rechts außen auf das Drei-Punkte-Menü, und aktivieren Sie die Option Pixel-Verhältnis des Gerätes hinzufügen.

Jetzt erscheint in der Geräte-Symbolleiste die Option DPR. Falls ein bestimmtes Gerät simuliert wird, ist dessen DPR fest eingestellt. Um den DPR-Wert ändern zu können, ändern Sie einfach die Größe des Viewports mit der Maus oder geben im Eingabefeld für dessen Abmessungen einen anderen Wert ein. Dann steht links daneben Abmessungen: Responsiv und Sie können die gewünschte DPR auswählen. Nach dem Ändern der DPR müssen Sie die Seite einmal neu laden.

Abbildung 6.4 zeigt, dass Chrome auf einem Bildschirm mit einer DPR von 1.0 zur Darstellung des Logos die Datei *html-und-css-logo-222.png* nutzt.

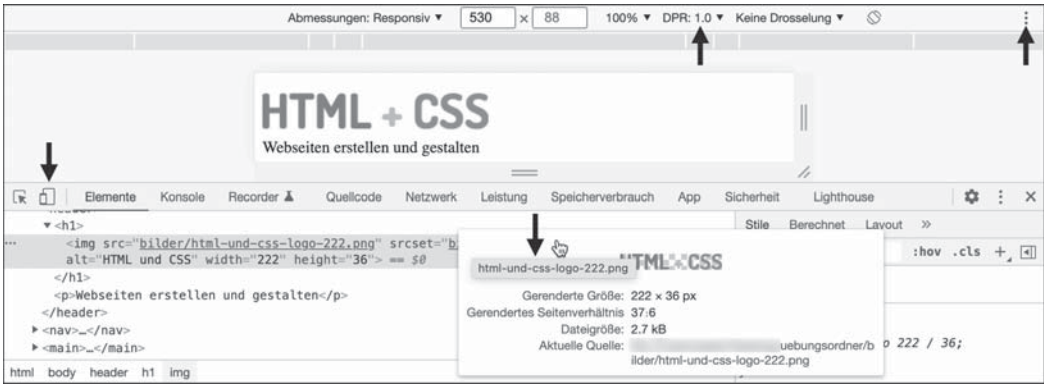

Abbildung 6.4 Bei einer DPR von 1.0 verwendet Chrome die kleine Grafik.

Wenn Sie im Entwicklertool die DPR auf 2.0 ändern und die Seite mit einem Rechtsklick NEU LADEN, verwendet Chrome für das Logo die Datei *html-und-css-logo-444.png* (Abbildung 6.5).

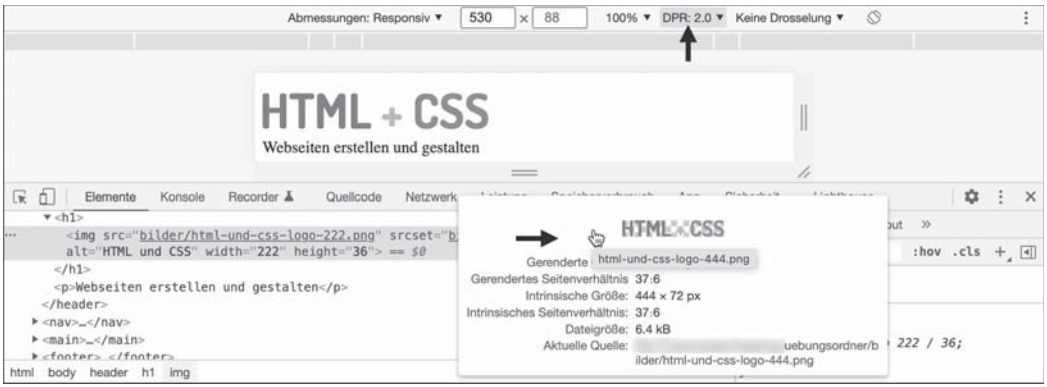

Abbildung 6.5 Bei einer DPR von 2.0 verwendet Chrome die große Grafik.

6.4 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 für dieses Kapitel im Unterordner *uebungen* die Datei *bilder-mit-flexibler-breite-einbinden.html*, in der das in Abbildung 6.6 gezeigte Bild eingebunden wird.

Kapitel 14
Die wichtigsten Selektoren und ihre Spezifität

Worin Sie die wichtigsten Selektoren kennenlernen, Beispiele für deren Einsatz sehen und erfahren, warum einige Selektoren wichtiger sind als andere.

Die Themen im Überblick:

- Einfache Selektoren: Elemente, Gruppierung und *
- Klassen sind klasse: der Selektor mit dem Punkt
- IDs sind einmalig: der Selektor mit der Raute
- Attributselektoren haben eckige Klammern: [attribut]
- DOM: die Hierarchie der HTML-Elemente
- Nachfahren auswählen: der Selektor mit Leerzeichen
- Selektoren zum Auswählen von Kindelementen
- Nachbarn und Geschwister selektieren mit + und ~
- Kurz vorgestellt: der praktische Elternselektor :has()
- Nützliche Quellen zum Nachschlagen von Selektoren
- Spezifität: Einige Selektoren sind wichtiger als andere
- Auf einen Blick

Der Umgang mit Selektoren ist eine der wichtigsten Fertigkeiten beim Gestalten per CSS, und deshalb spielen sie in diesem Kapitel die Hauptrolle. Zum Kennenlernen und zum späteren Nachschlagen. In den Übungsdateien finden Sie im Ordner zu diesem Kapitel im Unterordner *uebungen* dazu diverse Beispiele. Zum Abschluss des Kapitels erfahren Sie im Abschnitt über *Spezifität*, warum einige Selektoren aus Sicht des Browsers wichtiger sind als andere.

14.1 Einfache Selektoren: Elemente, Gruppierung und *

Selektoren wählen aus, welche Kästchen gestaltet werden sollen, und jede CSS-Regel beginnt mit einem *Selektor*. In diesem Abschnitt lernen Sie die ersten genauer kennen (siehe *element-und-universalselektor.html* in den Übungsdateien).

14.1.1 »Der Name der Kiste« – einfache Elementselektoren

Der einfachste Selektor ist der Name der Kästchen, die gestaltet werden sollen. Ein solcher *Elementselektor* wird manchmal auch *Typselektor* genannt, weil er alle Elemente eines bestimmten Typs auswählt:

```
p { color: maroon; }
```

Listing 14.1 Die Textfarbe für alle Absätze gestalten

In diesem Listing selektiert der Selektor `p` *alle* Absätze und färbt deren Text rotbraun.

14.1.2 Mehrere Kästchen zugleich: Selektoren mit einem Komma gruppieren

Manchmal ist es praktisch, Selektoren mit einem Komma zu gruppieren. Schauen Sie sich die folgenden Regeln an:

```
h1 { color: darkred; }
h2 { color: darkred; }
h3 { color: darkred; }
```

Listing 14.2 Einige sehr ähnliche CSS-Regeln

Für alle drei Selektoren wird dieselbe Textfarbe definiert. Durch eine Gruppierung der Selektoren mit einem Komma können Sie sich ein bisschen Tipparbeit sparen und die Übersichtlichkeit erhöhen:

```
/* Selektoren mit einem Komma gruppiert */
h1, h2, h3 { color: darkred; }
```

Listing 14.3 Selektoren gruppieren mit einem Komma

Wenn Sie jetzt die Schriftfarbe für alle Überschriftebenen ändern möchten, müssen Sie nur noch eine Zeile bearbeiten. Wichtig: Wenn in einer Gruppierung aufgrund z. B. eines Tippfehlers ein Selektor ungültig ist, wird die gesamte CSS-Regel ignoriert.

Gruppieren von Selektoren geht auch mit :is() und :where()

Statt Selektoren mit einem Komma zu gruppieren, kann man auch die relativ neuen Pseudoklassen `:is()` und `:where()` nutzen:

- *wiki.selfhtml.org/wiki/CSS/Selektoren/is*
- *wiki.selfhtml.org/wiki/CSS/Selektoren/where*

14.1.3 Alle Kästchen auswählen: der Universalselektor »*«

Sie werden ihn nur selten benutzen, aber es gibt ihn: das Sternchen * als universellen Selektor, der *alle* Kästchen selektiert:

```
/* Universalselektor - alle Elemente bekommen eine rote Umrisslinie */
* { outline: 1px solid red; }
```

Listing 14.4 Das Sternchen als universeller Selektor

Dieses Beispiel selektiert *alle* HTML-Elemente und macht sie mit einer 1 px breiten, durchgehenden roten Linie sichtbar. Mehr zur Eigenschaft `outline` erfahren Sie in Abschnitt 15.9 beim Gestalten von Hyperlinks.

14.2 Klassen sind klasse: der Selektor mit dem Punkt

Sehr praktisch ist die Möglichkeit, Elementen im HTML mit dem Attribut *class* eigene Namenszusätze geben zu können. Auch im Stylesheet der Übungswebsite haben Sie diese Klassen bereits ausgiebig genutzt, und im Folgenden werden sie ausführlich vorgestellt.

14.2.1 Beispiele für den Einsatz von Klassen auf der Übungswebsite

In Abschnitt 7.2, »Kopfbereiche auszeichnen mit <header>«, haben Sie gesehen, dass das Element `header` auf einer Seite mehrfach auftreten kann. Um gezielt nur den Kopfbereich der Seite gestalten zu können, haben Sie ihm auf der Übungswebsite die Klasse `site-header` gegeben:

```
<header class="site-header"> ...
```

Listing 14.5 Das Element <header> bekommt eine Klasse.

Im CSS selektieren Sie Klassen, indem Sie den Namen der Klasse hinschreiben und einen Punkt ohne Leerzeichen voranstellen:

```
.site-header { ... }
```

Listing 14.6 Im CSS nutzt man die Klasse als Selektor.

Ein weiteres Beispiel für den Einsatz von Klassen finden Sie auf der Startseite der Übungswebsite. Dort gibt es drei mit einem `article`-Element umgesetzte Infoboxen, die gemeinsam gestaltet werden sollen. Ein Elementselektor wie `article` ist zu ungenau, denn er würde *alle* Artikel auf *allen* mit dem Stylesheet gestalteten Webseiten auswählen. Um im CSS nur die drei Infoboxen auf der Startseite zu gestalten, bekommen sie im HTML eine gemeinsame Klasse:

```
<article class="infobox"><h3>News</h3> ... </article>
<article class="infobox"><h3>Über uns</h3> ... </article>
<article class="infobox"><h3>Kontakt</h3> ... </article>
```

Listing 14.7 Drei Elemente werden mit derselben Klasse gruppiert.

Um diese Elemente zu gestalten, schreiben Sie im CSS wie gehabt einen Punkt, gefolgt vom Namen der Klasse:

```
.infobox {
  text-align: center;
  background-color: white;
  padding: 1rem;
  margin: 1rem;
}
```

Listing 14.8 Der Selektor mit dem Punkt

Dieser Selektor wählt alle Elemente mit der Klasse `infobox` aus.

14.2.2 Gebundene Klassen: Klassen auf einen Elementtyp beschränken

Der Selektor `.infobox` wählt wie gesehen *alle* Elemente aus, die die Klasse `infobox` haben, egal ob es `article`, `section`, `div` oder ganz etwas anderes ist.

Um eine Klasse auf einen bestimmten Elementtyp zu beschränken, können Sie im CSS einen Elementselektor und eine Klasse kombinieren:

```
article.infobox {
  text-align: center;
  background-color: white;
  padding: 1rem;
  margin: 1rem;
}
```

Listing 14.9 Der Selektor mit dem Punkt und dem Namen des Elements

Mit diesem Selektor gestaltet der Browser alle `article`-Elemente mit der Klasse `infobox`. Elemente wie `<div class="infobox">` werden damit *nicht* selektiert. In Abschnitt 14.11, »Spezifität: Einige Selektoren sind wichtiger als andere«, werden Sie sehen, dass gebundene Klassen etwas wichtiger sind als ungebundene.

14.2.3 Ein HTML-Element kann mehrere Klassennamen haben

Sie können einem HTML-Element mehrere Klassen mit auf den Weg geben, wobei sie durch eine Leerstelle getrennt werden. Im folgenden Listing hat das `article`-Element die Klassen `infobox` und `wichtig`:

```
<article class="infobox wichtig"> ... </article>
```

Listing 14.10 Mehrere CSS-Klassen für ein HTML-Element

Dieses `article`-Element kann sowohl mit `.infobox` als auch mit `.wichtig` selektiert werden. Wenn Sie im CSS die Namen der Klassen mit Punkt und *ohne* Leerzeichen hintereinanderschreiben, müssen *alle* Klassen im HTML vorhanden sein:

```
.infobox.wichtig { ... }
```

Listing 14.11 Ein Element mit mehreren Klassen selektieren

Dieser Selektor gestaltet ein `article`-Element nur, wenn es sowohl die Klasse `infobox` als auch die Klasse `wichtig` hat.

14.3 IDs sind einmalig: der Selektor mit der Raute

Genau wie mit Klassen können Sie HTML-Elemente mit einer ID um eigene Namenszusätze ergänzen, aber anders als Klassen dürfen IDs frei nach dem Highlander-Motto »Es kann nur einen geben« auf jeder Seite nur ein einziges Mal auftauchen. ID ist die Kurzform für *Identität* und der Satz *Can I see your ID please?* ist im Englischen die Bitte, sich mit einem gültigen Dokument auszuweisen.

Kapitel 19

Der normale Flow, »position« und »float«

Worin Sie sehen, dass HTML-Elemente einem natürlichen Flow folgen. Danach lernen Sie, wie man Boxen mit »position« im Browserfenster verschieben und mit »float« nach links oder rechts schweben und von Text umfließen lassen kann.

Die Themen im Überblick:

- Flow: Die Seite ist ein langer ruhiger Fluss
- Versetzt weiterfließen mit »position: relative«
- Raus aus dem Flow mit »position: absolute«
- Wie ein Fels in der Brandung mit »position: fixed«
- Scrollen und stehen bleiben mit »position: sticky«
- Positionierte Boxen und der »z-index«
- Text um eine Abbildung fließen lassen mit »float«
- Floats beenden mit »clear: both«
- Praktisch: Klassen zum Floaten und Clearen
- Floats umschließen mit »display: flow-root«
- Floats nicht rechteckig umfließen mit »shape-outside«
- Auf einen Blick

In diesem Kapitel geht es los mit dem *Block Formatting Context* und seinem normalen *Flow* von Block- und Inline-Boxen. Anschließend sehen Sie, wie man mit den Eigenschaften `position` und `float` Boxen ganz oder teilweise aus diesem Flow heben und auf der Webseite positionieren kann.

Falls Sie die Übungen ausprobieren möchten, finden Sie die Übungsdateien wie immer im Ordner zu diesem Kapitel im Unterordner *uebungen*.

19.1 Flow: Die Seite ist ein langer ruhiger Fluss

Das Layouten mit CSS basiert auf der Idee, dass die von HTML-Elementen erzeugten Boxen innerhalb eines *Formatting Context* existieren, einer *Gestaltungsumgebung*, in der bestimmte Regeln gelten.

Nach dem Erstellen einer Webseite arbeiten Sie in einem sogenannten *Block Formatting Context*, der durch das Stammelement `html` erstellt wird. Darin gibt es drei Möglichkeiten zum Layouten von Boxen: der normale Flow von Block- und Inline-Boxen, die Eigenschaft `position` und die Eigenschaft `float`. Diese beiden Eigenschaften, die Sie in diesem Kapitel kennenlernen, waren lange Zeit alles, was Webdesigner zum Layouten hatten.

19.1.1 Der normale Flow im Block Formatting Context

In einem *Block Formatting Context* gibt es Block- und Inline-Boxen, die einfachen Layoutregeln folgen (siehe Abschnitt 4.6):

- *Inline-Boxen* fließen in Schreibrichtung *horizontal, von links nach rechts*. Sie fangen so weit wie möglich links oben an, werden nur so breit wie ihr Inhalt und stehen automatisch *nebeneinander*. Wenn kein Platz mehr ist, rutschen sie eine Zeile tiefer und fangen wieder links an.
- *Blockboxen* fließen *vertikal, von oben nach unten*. Sie beginnen ebenfalls so weit wie möglich links oben, *blocken* aber den in Schreibrichtung nach rechts zur Verfügung stehenden Platz und stehen dadurch automatisch *untereinander*.

Inline-Boxen fließen also von links nach rechts, Blockboxen von oben nach unten. Das ist der normale Flow eines Dokuments. Boxen, die sich in diesem Flow befinden, respektieren einander und überlappen sich nicht. *Panta rhei* – alles fließt.

19.1.2 Auch kürzere Blockboxen stehen im Flow untereinander

Um ein bisschen Gefühl für den Flow zu bekommen, zeige ich Ihnen vor der Zähmung der Widerspenstigen durch `position` und `float` zunächst kurz das natürliche Verhalten der Boxen in freier Wildbahn. Bemerkenswert ist dabei besonders, dass auch kurze Blockboxen immer untereinander stehen.

Abbildung 19.1 zeigt drei `div`-Elemente, die mit `max-width: 25%` in der Breite verkürzt wurden. Horizontal hätte der Browser also genügend Platz, um alle drei Boxen nebeneinanderzustellen. Blockboxen fließen aber von oben nach unten, und die drei Blockboxen bleiben daher untereinander stehen, auch wenn genügend Platz für ein Nebeneinander wäre.

Abbildung 19.1 Blockboxen stehen im Flow immer untereinander.

Der Flow ist für einfache Dokumente mit Text und ein paar Bildern gedacht und zum Erstellen von komplexen Layouts eher ungeeignet. Für Elemente im Flow kann man in einem Block Formatting Context eigentlich nur mit `display` die Art der Box ändern. Diesen Trick haben Sie beim Gestalten des Kontaktformulars in Abschnitt 9.8 bereits genutzt, als Sie dem Element `label` zur Beschriftung eines Formularfeldes je nach Bedarf die Deklarationen `display: block` bzw. `display: inline` zugewiesen haben.

Für das Nebeneinander der Dinge gibt es Flexbox und CSS-Grid

Für das Nebeneinander in mehrspaltigen Layouts lernen Sie weiter hinten im Buch neue *Formatting Contexts* kennen:

- Den *Flex Formatting Context* mit `display: flex` (Flexbox, Kapitel 21)
- Den *Grid Formatting Context* mit `display: grid` (CSS-Grid, Kapitel 22)

19.2 Versetzt weiterfließen mit »position: relative«

Zur Positionierung von Boxen im Viewport gibt es die Eigenschaft `position`, deren Standardwert `static` lautet. Das bedeutet für Boxen nichts anderes als »Nimm deine normale Position im Flow ein«.

Die relative Positionierung mit `position: relative` belässt die Elemente im Flow, macht aber zusätzlich zwei Dinge:

- Sie verschiebt eine Box von ihrer normalen Position im Flow.
- Sie markiert den ursprünglichen Platz der Box als geschützt.

Mit den Eigenschaften `top`, `right`, `bottom` und `left` können Sie positionierte Boxen im Viewport verschieben. Als *positioniert* gilt eine Box, wenn `position` einen anderen Wert als `static` hat. Bei Boxen mit `position: static` werden `top`, `right`, `bottom` und `left` ignoriert.

Im folgenden Beispiel wird das erste `div` relativ positioniert und bekommt mit `top` und `right` eine Positionierung zugewiesen:

```
div:first-child {
  background: #ecc;
  position: relative;
  top: 1rem;
  right: 0.5rem;
}
```

Listing 19.1 Das erste wird relativ positioniert.

Abbildung 19.2 zeigt das Ergebnis im Browserfenster.

Abbildung 19.2 Relative Positionierung in Aktion

Die beiden anderen Boxen haben sich nicht bewegt und verhalten sich so, als sei die positionierte Box noch an ihrem *ursprünglichen* Platz im Flow. Box 1 hingegen wurde verschoben:

- `top:1rem` drückt die Box um 1 rem nach *unten*.
- `right:0.5rem` schiebt die Box um 0,5 rem nach *links*.

Bei der relativen Positionierung ist der Bezugspunkt für die Werte von `top`, `right`, `bottom` und `left` also die ursprüngliche Position der Box im Flow. Das ist in sich alles ganz logisch, wirkt aber anfangs etwas umständlich. In der Praxis werden Sie `position: relative` in erster Linie nutzen, um einen Bezugspunkt für die absolute Positionierung zu definieren (siehe Abschnitt 19.3.2).

19.3 Raus aus dem Flow mit »position: absolute«

Im Gegensatz zur relativen nimmt die *absolute Positionierung* das Element komplett aus dem Flow heraus, und es wird – bildlich gesprochen – aus dem Fluss gezogen. Alle anderen Elemente auf der Seite verhalten sich so, als ob es gar nicht da wäre.

19.3.1 Absolute Positionierung hebt eine Box aus dem Flow

Das HTML für dieses Beispiel ist bis auf ein Wort identisch mit dem für die relative Positionierung in Listing 19.1:

```
div:first-child {
  background: #ecc;
  position: absolute;
  top: 1rem;
  right: 0.5rem;
}
```

Listing 19.2 Das erste »div« wird absolut positioniert.

Im CSS wurde `relative` durch `absolute` ersetzt, aber die Wirkung ist enorm. Box 1 steht plötzlich rechts außen, und die beiden anderen Boxen rutschen nach oben (siehe Abbildung 19.3).

Abbildung 19.3 Nur ein Wort geändert – und »absolute« anders

Absolut positionierte Boxen werden aus dem Flow gehoben und liegen *über* den anderen. Die genauen Koordinaten werden wieder mit den Eigenschaften `top`, `right`, `bottom` oder `left` angegeben, aber die Werte für diese vier Eigenschaften orientieren sich nicht mehr an der ursprünglichen Flow-Position der Box:

- Die absolute Positionierung einer Box bezieht sich auf die nächste umgebende, *positionierte* Box. Als positioniert gilt eine Box, wenn `position` nicht den Wert `static` hat.
- Falls keine umgebende positionierte Box vorhanden ist, erfolgt die Positionierung relativ zum Stammelement `html`.

Diese beiden Aussagen können Sie wie folgt zusammenfassen:

- *Absolute* Positionierung ist *immer relativ* zu einem Bezugspunkt.
- Es gibt zwei mögliche Bezugspunkte:
  - das nächste umgebende *positionierte* Element
  - das Stammelement `html`

Die Kombination von `relative` und `absolute` sorgt im Alltag beim Umgang mit `position` für viel Verwirrung und wird daher im folgenden Abschnitt an einem konkreten Beispiel erläutert.

19.3.2 Ein beliebter Trick: absolute und relative Positionierung kombinieren

In der Praxis kombiniert man die absolute und relative Positionierung, und diesen Trick nutzen Sie in diesem Abschnitt, um eine Beschriftung über ein Bild zu legen. Als Grundlage dient das Beispiel aus Abschnitt 6.5, »Abbildungen beschriften: <figure> und <figcaption>«. Hier zur Erinnerung das HTML:

```
<figure>
  <img src="bilder/hamburg-blankenese-treppenviertel-400.jpg"
       srcset="bilder/hamburg-blankenese-treppenviertel-800.jpg"
       alt=""
       width="400" height="300">
  <figcaption>Blick auf das Treppenviertel in Blankenese</figcaption>
</figure>
```

Listing 19.3 Das HTML für die Bildbeschriftung

Abbildung 19.4 zeigt das Bild und darunter die Beschriftung.

Abbildung 19.4 Bildbeschriftung mit <figure>, <img> und <figcaption>

Um `figcaption` über das Bild zu legen, wird im CSS zunächst einmal das umgebende Element `figure` auf die Breite des Bildes beschränkt und relativ positioniert:

```
figure {
  position: relative;
  max-width: 400px;
  margin: auto;
}
```

Listing 19.4 <figure> wird relativ positioniert.

Die relative Positionierung verändert optisch nichts. `figure` bleibt im Flow und verändert seine Position nicht, wird durch das `position: relative` aber zum Bezugspunkt für die im folgenden Listing gezeigte absolute Positionierung von `figcaption`:

```
figcaption {
  position: absolute;
  left: 0;
  right: 0;
  bottom: 0;
  text-align: center;
  color: white;
  background: rgb(50,50,50,0.8);
  padding: 0.5rem;
}
figure img { display: block; }
```

Listing 19.5 Die Beschriftung absolut positionieren und gestalten

In diesem Listing wird zunächst die Position von `figcaption` definiert:

- `position: absolute` hebt die Box aus dem Flow.
- `left: 0` und `bottom: 0` positionieren die Box ganz links und ganz unten. Bezugspunkt ist `figure`, weil es relativ positioniert wurde und somit das erste umgebende, positionierte Element ist.
- `right: 0` bewirkt, dass die Beschriftung sich über die volle Breite von `figure` erstreckt, von ganz links bis ganz rechts.
- `img` bekommt `display: block`. Ohne diese Deklaration wäre es eine Inline-Box, und es gäbe eine kleine Lücke unter dem Bild.

In den weiteren Deklarationen bekommt `figcaption` unter anderem einen leicht transparenten Hintergrund. Abbildung 19.5 zeigt, dass die Beschriftung nach diesem Listing über dem Bild liegt.

Abbildung 19.5 Die Beschriftung liegt halbtransparent über dem Bild.

19.4 Wie ein Fels in der Brandung mit »position: fixed«

Die feste Positionierung mit `position:fixed` verhält sich fast genau wie `position: absolute`, mit einem kleinen, aber feinen Unterschied: Ein fixiertes Element scrollt nicht mit.

Absolut positionierte Elemente sind relativ zu einem Bezugspunkt im Dokument und scrollen daher mit. Bei fixierten Elementen ist das anders:

- Fixierte Elemente orientieren sich am Browserfenster selbst und nicht am Stammelement `html` innerhalb dieses Fensters.
- Da das Browserfenster selbst nicht mitscrollt, bleiben fixierte Elemente an derselben Stelle stehen.

Wie bei der absoluten Positionierung ist nicht mehr der Browser, sondern der Webdesigner dafür verantwortlich, dass sich fixierte Elemente nicht danebenbenehmen.

In Abbildung 19.6 sehen Sie als Beispiel einen Link *Zurück nach oben*, der immer rechts unten im Browserfenster steht.

Gegeben sei folgendes HTML. Das Kürzel `&#8593;` fügt das Unicode-Zeichen für einen nach oben zeigenden Pfeil ein:

```
<body>
<article>
    <h1>Wie ein Fels in der Brandung</h1>
    <p>Absatz 1. Lorem ipsum ... </p>
    <p>Absatz 2. Nullam vulputate ... </p>
```