

Peter Müller

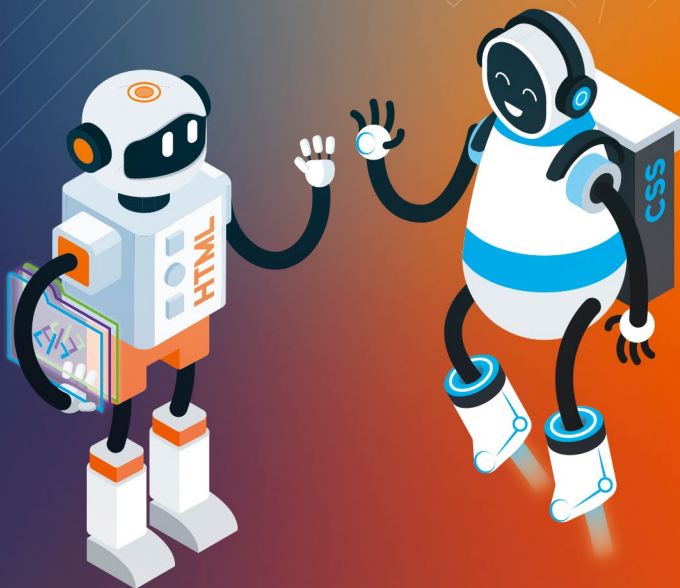
Inkl. digitaler
Barrierefreiheit

Einstieg in HTML und CSS

Webseiten programmieren und gestalten

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

3. Auflage



Alle Übungsdateien zum Download



Rheinwerk
Computing

Vorwort

Das Web ist allgegenwärtig, und alle Webseiten bestehen aus HTML und CSS. Diese beiden Sprachen sind das Fundament des Web, und wenn Sie mehr darüber wissen möchten, sind Sie hier genau richtig. Dieses Buch vermittelt modernes Grundwissen zu HTML und CSS, und in dieser dritten Auflage wurden die Inhalte aktualisiert, zum Teil neu strukturiert und um zahlreiche Abschnitte erweitert.

Für wen ist dieses Buch?

Sie sollten mit einem Computer und einem Editor umgehen können und sich vor einer hexadezimalen Farbangabe wie `#07b` nicht erschrecken. Abgesehen davon sind die einzigen Voraussetzungen Interesse an HTML und CSS und Lust am Lernen. Unter anderem ist dieses Buch gedacht für

- ▶ alle, die mehr über HTML und CSS wissen möchten,
- ▶ Einsteiger, die wissen möchten, wie man Webseiten erstellt und gestaltet und wie HTML und CSS funktionieren,
- ▶ Webdesigner, die eine kompakte, strukturierte Einführung in modernes HTML und CSS suchen,
- ▶ Nutzer von Content-Management-Systemen wie WordPress oder anderen, die das HTML und CSS in ihren Themes, Templates und Layouts verstehen und vielleicht anpassen möchten,
- ▶ Webworker, die vorhandenes, älteres Wissen zu HTML und CSS aktualisieren und auf den neuesten Stand bringen möchten,
- ▶ Programmierer, die sich fragen, wozu man HTML braucht und wie dieses komische CSS-Zeug funktioniert. *CSS is awesome.*

Wie ist dieses Buch aufgebaut?

Das Buch besteht aus vier aufeinander aufbauenden Teilen.

▶ Teil I: Webseiten, HTML und CSS

In Kapitel 1 erfahren Sie, dass Webseiten aus Quelltext bestehen, und bekommen Tipps zu Browsern, Editoren und Referenzen. In Kapitel 2 und Kapitel 3 folgt dann ein Schnelleinstieg zu HTML und CSS.

► **Teil II: HTML (mit einer Prise CSS)**

Von Kapitel 4 bis Kapitel 10 lernen Sie die wichtigsten HTML-Elemente kennen: Überschriften, Absätze, Listen, Bilder, Audio, Video, semantische Strukturelemente, Formulare und Tabellen. Diese Elemente bekommen gleich eine grundlegende Gestaltung per CSS. In Kapitel 11 erweitern Sie die bis dahin erstellte Übungswebseite zu einer Übungswebsite.

► **Teil III: CSS-Grundlagen**

In Kapitel 12 bis Kapitel 17 geht es um die Grundlagen von CSS: Box-Modell, Farbwerte, Einheiten, Selektoren, Textgestaltung, Kaskade, Vererbung und Standardwert sowie das Gestalten der Boxen selbst. In Kapitel 18 finden Sie dann benutzerdefinierte Eigenschaften (CSS-Variablen) und Tipps zum Aufräumen und Organisieren von Stylesheets.

► **Teil IV: CSS-Layout**

In Kapitel 19 bis Kapitel 22 lernen Sie verschiedene Techniken zum Erstellen von CSS-Layouts kennen: den Flow, position und float, Media Queries, Flexbox und CSS-Grid. In Kapitel 23 machen Sie einen Ausflug in die Welt flexibler Icons (mit SVG) und responsiver Bilder, und zum Abschluss erstellen Sie in Kapitel 24 für die Übungswebsite eine responsive Navigation.

In vielen Kapiteln gibt es Kästchen mit dem Titel »Übungswebsite«. Wenn Sie diese Kästchen der Reihe nach durcharbeiten, erhalten Sie eine funktionierende Website, wenn Sie den Text dazwischen lesen, wissen Sie auch, warum.

Wie sollten Sie dieses Buch lesen?

- Einsteiger arbeiten das Buch am besten von vorne bis hinten durch, denn die Themen bauen aufeinander auf.
- Wenn Sie schon etwas Vorwissen haben, überfliegen Sie die Kapitel kurz und picken sich dann die Themen raus, die Sie am interessantesten finden.
- Um vorhandenes Wissen zu HTML und CSS zu aktualisieren, schauen Sie einfach ins Inhalts- oder Stichwortverzeichnis und springen direkt zu den Themen, die Sie interessieren.

Die Übungswebsite zieht sich wie ein roter Faden durch das gesamte Buch, aber mit den Übungsdateien können Sie in jedem Kapitel einsteigen und mitmachen.

Was ist dieses Buch nicht?

Sie erstellen zwar Schritt für Schritt eine kleine Übungswebsite, aber das Buch ist *keine* Anleitung zum Veröffentlichen und Betreiben von Websites. Themen wie Domain-Namen, Webpace, SSL, SFTP, rechtliche Aspekte oder die Optimierung für Suchmaschinen (SEO) kommen in diesem Buch nicht oder nur am Rande vor.

Es geht um das Verstehen von HTML und CSS, und der Schwerpunkt liegt auf klassischen Webseiten zur Präsentation von Informationen, nicht auf der Interaktion mit Benutzern.

Die Website zum Buch: Übungsdateien und Errata

Auf der folgenden Website erhalten Sie aktuelle Informationen zu HTML und CSS sowie Errata zum Buch:

► html-und-css.de

Auf dieser Website können Sie auch die Übungsdateien herunterladen.

Nach dem Entpacken des ZIP-Archivs finden Sie für die meisten Kapitel jeweils einen Ordner mit ein oder zwei Unterordnern:

- Der Unterordner *uebungen* enthält einzelne Übungsdateien zu Themen, die im jeweiligen Kapitel behandelt werden.
- Der Unterordner *uebungswebsite* enthält den jeweils aktuellen Stand der Schritt für Schritt weiterentwickelten Übungswebsite.

Den Unterordner *uebungswebsite* gibt es nicht zu jedem Kapitel, aber wenn es ihn gibt, dann hat er zwei Unterordner namens *anfang* und *ende*:

- Der Ordner *anfang* enthält den Stand der Übungswebsite am Anfang des Kapitels. Wenn Sie ein bestimmtes Kapitel durcharbeiten möchten, kopieren Sie diese Ordner und Dateien in einen Übungsordner und legen los.
- Im Ordner *ende* liegen die fertigen Übungsdateien so, wie sie am Ende des Kapitels sein sollten, wenn Sie alle mit »Übungswebsite« beschrifteten Kästchen durchgearbeitet haben. So können Sie, falls etwas partout nicht klappen will, nachschauen, wie es sein sollte.

Mit den Übungsdateien können Sie wie gesagt in jedem Kapitel einsteigen: Einfach den Inhalt vom Ordner *anfang* kopieren, in einem leeren Übungsordner wieder einfügen, und dann kann es losgehen.

Vielen Dank ...

- ▶ ... an Sie als Leser. Ohne Sie würde das Schreiben nur halb so viel Spaß machen.
- ▶ ... an die vielen Autoren von Artikeln zu HTML und CSS im Web.
- ▶ ... an alle Mitarbeiter beim Rheinwerk Verlag, die zur Entstehung des Buches beigetragen haben. Besonders an Stephan Mattescheck vom *Lektorat Computing* und Isolde Kommer für das Korrektorat.
- ▶ ... an Annette Schwindt für die wochenlange detaillierte Auseinandersetzung mit dem Manuskript und das sich daraus ergebende Feedback. Danke nicht nur für die vielen guten Anregungen, sondern auch für den Spaß bei der Arbeit.
- ▶ ... an Karin, Hobbe, Irma, Maud und Axel.

Peter Müller

Groningen

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.

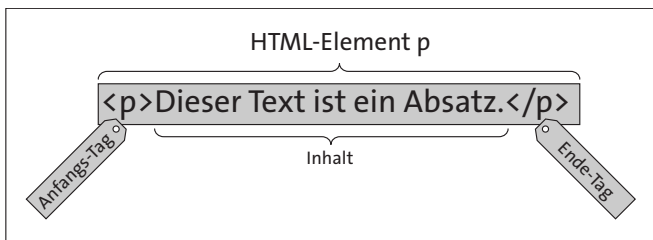


Abbildung 2.2 Ein Element besteht aus Anfangs-Tag, Inhalt und Ende-Tag.

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 Ordernamen 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.

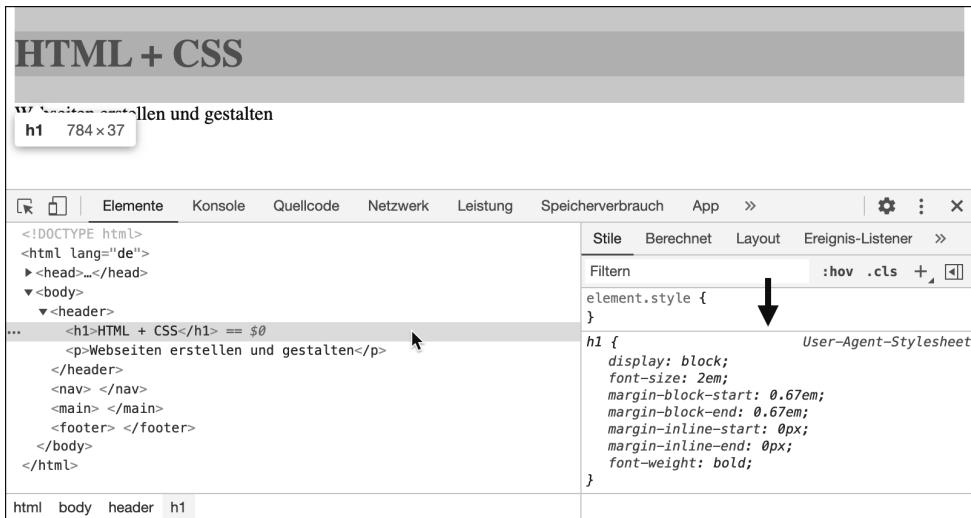


Abbildung 3.1 »index.html« in Chrome mit geöffnetem Entwicklerwerkzeug

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.

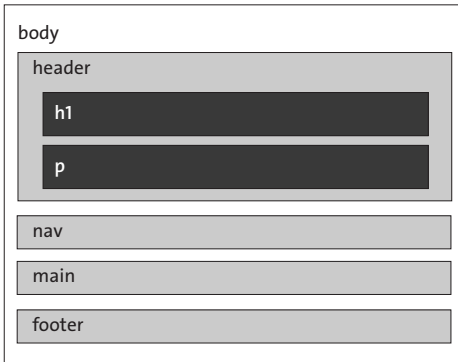


Abbildung 3.2 Die Elemente im Quelltext als rechteckige Kästchen

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  +  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.

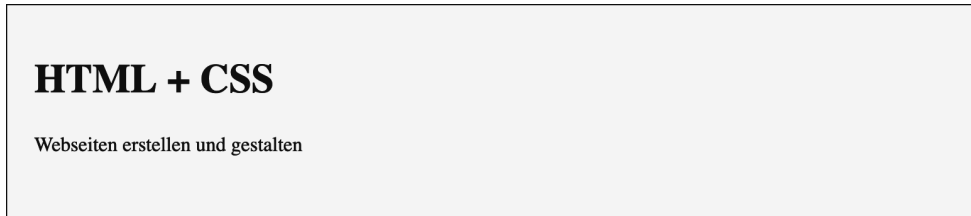


Abbildung 3.3 Die Seite mit einer hellen Hintergrundfarbe für `<body>`

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 ``
- ▶ 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

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. 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 *ersetzttes Element* (*replaced element*), denn das ``-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 ``

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>
  
</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.

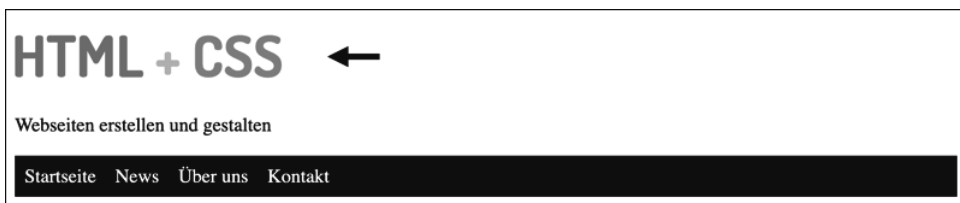


Abbildung 6.1 Die Übungswebsite mit einem Bild als Logo

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.

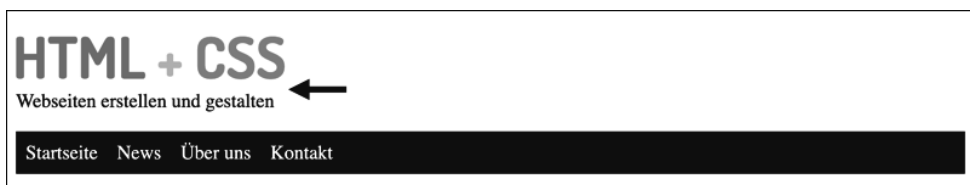


Abbildung 6.2 Logo und Slogan mit angepassten Außenabständen

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):

```

```

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
<i>html-und-css-logo-222.png</i>	222 px	36 px	2,16 kb
<i>html-und-css-logo-444.png</i>	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:

```

```

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 AUS-BLENDEN (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.

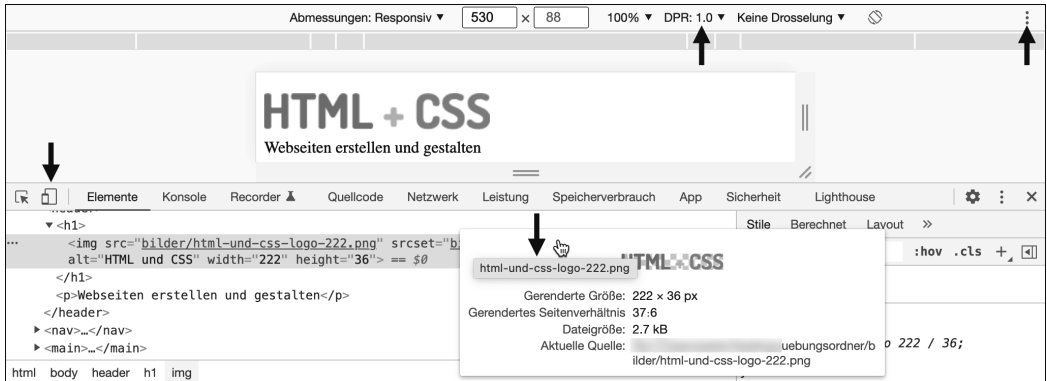


Abbildung 6.4 Bei einer DPR von 1.0 verwendet Chrome die kleine Grafik.

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).

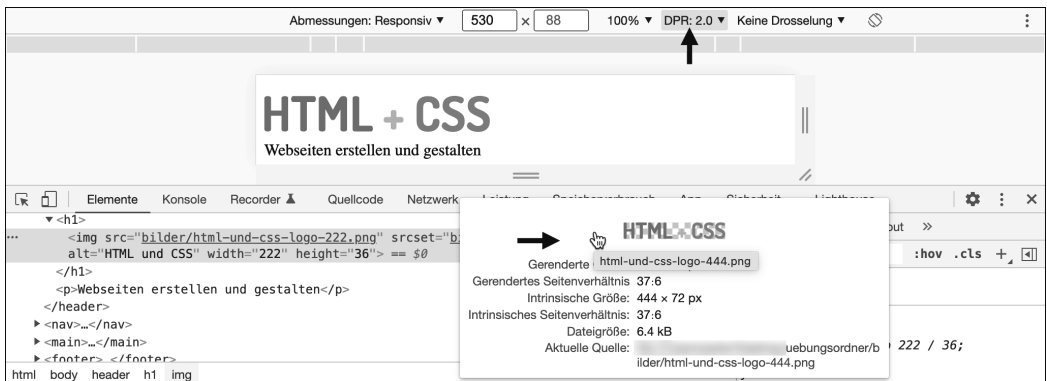


Abbildung 6.5 Bei einer DPR von 2.0 verwendet Chrome die große Grafik.

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 1px 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 Elementselector 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 Elementselector 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 `<div>` wird relativ positioniert.

Abbildung 19.2 zeigt das Ergebnis im Browserfenster.

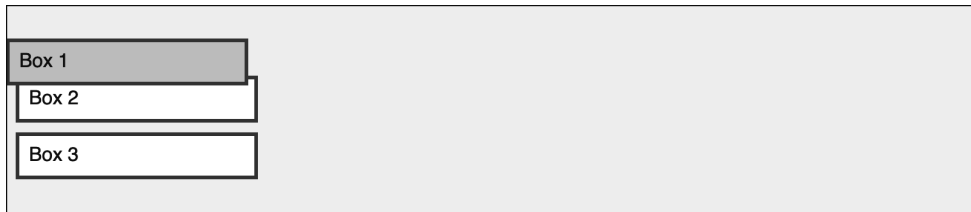


Abbildung 19.2 Relative Positionierung in Aktion

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).

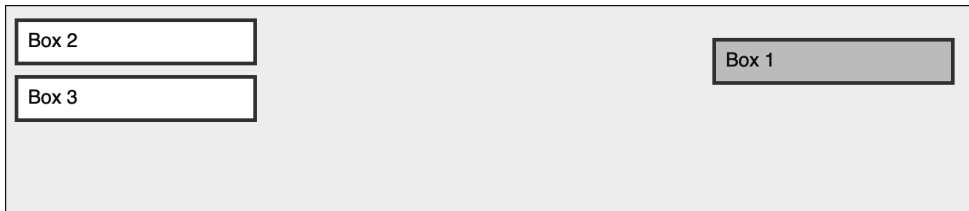


Abbildung 19.3 Nur ein Wort geändert – und »absolute« anders

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>
  
  <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.

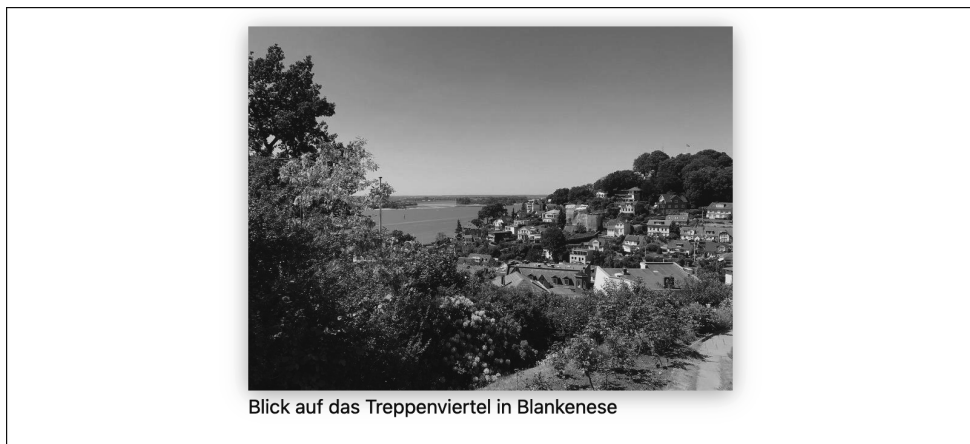


Abbildung 19.4 Bildbeschriftung mit `<figure>`, `` und `<figcaption>`

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 halbttransparent 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 `↑`; 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>
```

Inhalt

Materialien zum Buch	26
Geleitwort	27
Vorwort	29

TEIL I Webseiten, HTML und CSS

1 Wissenswertes über Webseiten 35

1.1 Webseiten sehen nicht überall gleich aus	35
1.2 Webseiten bestehen aus Quelltext	37
1.3 Quelltext besteht aus HTML, CSS und JavaScript	38
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	41
1.4.1 Die bekanntesten Browser: Chrome, Firefox, Safari, Edge und Co	41
1.4.2 Viele Browser sind miteinander verwandt	42
1.4.3 Besonderheiten: Browser unter iOS und Internet Explorer	42
1.5 Webseiten und Barrierefreiheit	43
1.6 Editoren zum Schreiben und Bearbeiten von Quelltext	44
1.7 Websites zum Nachschlagen von HTML und CSS	45
1.7.1 SelfHTML – das deutschsprachige Urgestein	46
1.7.2 Die »MDN Web Docs« – das Nachschlagwerk	46
1.7.3 Anlaufstelle für Fragen zur Browserunterstützung: »caniuse.com«	47
1.8 Auf einen Blick	48

2 HTML kennenlernen: die erste Webseite erstellen 49

2.1 Webseiten bestehen aus rechteckigen Kästchen	50
2.2 HT-M-L: die »HyperText Markup Language«	51

2.2.1	HT wie »Hypertext«: Hyperlinks erstellen	51
2.2.2	M wie »Markup«: Etiketten kleben	51
2.2.3	L wie »Language«: Vokabeln und Grammatikregeln	51
2.2.4	Der Unterschied zwischen »HTML-Elementen« und »HTML-Tags«	52
2.3	Die erste Webseite erstellen: »index.html«	52
2.3.1	Die Datei »index.html« im Editor erstellen und speichern	53
2.3.2	Eine gute Angewohnheit: <!-- Kommentare -->	53
2.4	Jede Webseite hat ein HTML-Grundgerüst	54
2.5	Der <!doctype> und das Stammelement <html>	56
2.5.1	Die Dokumenttyp-Definition <!doctype html>	56
2.5.2	Das Stammelement: <html> und </html> umschließen den Quelltext	56
2.6	HTML-Elemente können im Anfangs-Tag Attribute enthalten	57
2.7	<head> enthält wichtige Infos über die Webseite	58
2.7.1	<meta charset="utf-8"> ist die Angabe des Zeichensatzes	58
2.7.2	<meta name="viewport" ...> ist wichtig für responsive Webseiten	58
2.7.3	Zwischen <title> und </title> steht der Seitentitel	60
2.7.4	<meta name="description"> enthält eine kurze Beschreibung der Seite	61
2.8	<body> enthält den im Browser sichtbaren Bereich der Webseite	61
2.9	Der Kopfbereich <header> mit Überschrift und Slogan	63
2.10	Entwicklerwerkzeuge: HTML im Browser untersuchen	64
2.11	Auf einen Blick	66
3	CSS kennenlernen: die erste Webseite gestalten	67
3.1	Jeder Browser hat ein eingebautes Stylesheet	67
3.2	HTML-Elemente als rechteckige Kästchen visualisieren	69
3.3	Das erste eigene Stylesheet: »style.css«	70
3.3.1	Schritt 1: Einen Unterordner und ein Stylesheet erstellen	70
3.3.2	Schritt 2: HTML-Datei und CSS-Datei verbinden mit <link>	71

3.4	Die erste eigene CSS-Regel: Hintergrund- und Schriftfarbe für <body>	72
3.4.1	Auch in CSS eine gute Angewohnheit: /* Kommentare */	72
3.4.2	Hintergrund- und Schriftfarbe für <body> ändern	72
3.5	Den Kopfbereich <header> im CSS selektieren und gestalten	74
3.5.1	Hintergrund- und Schriftfarbe für <header> ändern	74
3.5.2	Etwas Abstand zwischen Text und Rand einfügen mit »padding«	75
3.6	Wichtige Vokabeln: der Aufbau einer CSS-Regel	76
3.7	Entwicklerwerkzeuge: CSS im Browser untersuchen	77
3.8	Auf einen Blick	79

TEIL II HTML (mit einer Prise CSS)

4 HTML-Elemente für Text: Überschriften, Absätze, Hervorhebungen und Listen 83

4.1	Überschriften strukturieren den Inhalt: <h1> bis <h6>	84
4.1.1	HTML kennt sechs Ebenen für Überschriften	84
4.1.2	Eine <h2>-Überschrift im Inhaltsbereich einfügen	85
4.2	Absätze und Hervorhebungen: <p>, , 	86
4.2.1	Absätze mit <p> und Hervorhebungen mit und 	86
4.2.2	Absätze und Hervorhebungen auf der Übungswebsite einfügen	86
4.2.3	HTML-Elemente verschachteln – zuerst geöffnet, zuletzt geschlossen	87
4.3	Webseiten in unterschiedlichen Umgebungen testen	88
4.4	Listen erstellen mit , und 	89
4.4.1	Ungeordnete Listen mit einem Aufzählungszeichen: und 	90
4.4.2	Geordnete Listen mit einer Nummerierung: und 	91
4.5	Listen verschachteln: eine Liste in einer Liste	93
4.6	Über Blockelemente, Inline-Elemente und »display«	95
4.6.1	Blockelemente werden so breit, wie es geht, und fließen von oben nach unten	95
4.6.2	Inline-Elemente werden nur so breit wie ihr Inhalt und fließen von links nach rechts	95
4.7	Auf einen Blick	96

5 HTML und Hyperlinks – das Besondere am Web 97

5.1	Das Standardverhalten von Hyperlinks	97
5.2	Anatomie eines Hyperlinks: <code>Linktext</code>	98
5.3	Hyperlinks in neuem Tab oder Fenster öffnen	100
5.4	Eine Navigation ist eine Liste mit Links	101
5.5	Eine grundlegende Gestaltung für die Navigation	103
5.5.1	Schritt 1: Die Navigationsliste gestalten	104
5.5.2	Schritt 2: Den Navigationsbereich gestalten	105
5.5.3	Schritt 3: Den Text der Navigationslinks gestalten	106
5.5.4	Schritt 4: Die Links im Navigationsbereich gestalten	107
5.6	Im Fußbereich einen Link »Nach oben« einfügen	109
5.6.1	Schritt 1: Das HTML für einen Link nach oben auf derselben Seite	109
5.6.2	Schritt 2: Sanftes Scrollen aktivieren	110
5.6.3	Schritt 3: Eine grundlegende Gestaltung für den Footer	111
5.7	Links zu Dateien und E-Mail-Adressen	112
5.7.1	Hyperlinks auf Dateien, die keine Webseiten sind: PDF & Co	112
5.7.2	Links auf E-Mail-Adressen	113
5.8	Auf einen Blick	114

6 HTML-Elemente für Bilder, Audio und Video 115

6.1	Über Grafikformate im Web: JPEG, GIF, PNG, SVG & Co	115
6.2	Ein Bild als Logo einbinden mit <code></code>	117
6.2.1	Das Element <code></code> und seine wichtigsten Attribute	117
6.2.2	Ein Logo auf der Übungswebsite einfügen mit <code></code>	118
6.2.3	Die Abstände zwischen Logo und dem Absatz darunter anpassen	119
6.3	Pixelbilder in Zeiten hochauflösender Bildschirme	121
6.3.1	Das Problem: Das Logo ist auf hochauflösenden Bildschirmen unscharf	121
6.3.2	Einfache Lösung: Eine doppelt so große Grafik einbinden	122
6.3.3	Besser: Je nach Pixeldichte unterschiedliche Dateien einbinden	123
6.3.4	Testen: Die korrekte Einbindung der Grafiken im Browser überprüfen	124

6.4	Bilder mit flexibler Breite: »max-width: 100%«	125
6.4.1	Das Problem: Pixelbilder haben eine feste Breite	126
6.4.2	Die Lösung: Flexible Bilder mit »max-width: 100%«	127
6.5	Abbildungen beschriften: <figure> und <figcaption>	128
6.5.1	Ein Foto mit einer Beschriftung: <figure> und <figcaption> im Einsatz	128
6.5.2	Die Einrückung von <figure> entfernen und die Beschriftung zentrieren	129
6.6	»Lazy Loading«: Seiten mit vielen Bildern optimieren	130
6.7	Audiodateien einbinden mit <audio>	131
6.7.1	Audioformate, Browserunterstützung und Audioplayer	131
6.7.2	Die Einbindung von Sound-Dateien mit <audio>	132
6.7.3	Audiodateien beschriften mit <figure> und <figcaption>	133
6.8	Videodateien einbinden mit <video>	133
6.8.1	Videoformate und Browserunterstützung im Überblick	134
6.8.2	Die Einbindung von Videodateien mit <video>	134
6.8.3	Flexible Videos per CSS mit »max-width: 100%«	135
6.9	Auf einen Blick	136

7 Semantische HTML-Elemente zur Strukturierung von Webseiten und Inhalten 139

7.1	Warum semantische Strukturelemente eine sinnvolle Sache sind	140
7.2	Kopfbereiche auszeichnen mit <header>	140
7.2.1	Das Element <header> kann auf einer Seite mehrfach vorhanden sein.	140
7.2.2	Übungswebsite: Den Kopfbereich um eine Klasse erweitern	141
7.3	Navigationsbereiche erstellen mit <nav>	143
7.3.1	<nav> für die Site-Navigation auf der Übungswebsite	143
7.3.2	Übungswebsite: Den Navigationsbereich um eine Klasse erweitern	143
7.3.3	<nav> kann in der HTML-Struktur auch an anderen Positionen stehen	144

7.4	Der Hauptinhalt einer Webseite steht in <main>	145
7.4.1	Übungswebsite: Den Inhaltsbereich um eine Klasse und eine ID erweitern	145
7.5	Fußbereiche auszeichnen mit <footer>	146
7.5.1	Der Fußbereich <footer> auf der Übungswebsite	146
7.5.2	Übungswebsite: Den Fußbereich um eine Klasse erweitern	147
7.6	Inhaltliche Abschnitte erstellen mit <section>	147
7.7	In sich geschlossene, eigenständige Blöcke mit <article>	149
7.7.1	Infoboxen erstellen mit dem Element »article«	150
7.7.2	Grundlegende Gestaltung für den Abschnitt und die Infoboxen	151
7.8	Bereiche mit zusätzlichen Informationen: <aside>	153
7.9	Elemente mit <div> semantisch neutral gruppieren	155
7.10	Orientierungspunkte für Screenreader: »ARIA Landmark Roles«	156
7.10.1	»ARIA Landmark Roles« bieten Orientierungspunkte für Screenreader	157
7.10.2	Semantische Strukturelemente haben integrierte Orientierungspunkte	158
7.10.3	ARIA Landmarks mit einer Erweiterung im Browser testen	159
7.11	Auf einen Blick	160

8 Weitere HTML-Elemente zur Auszeichnung von Text 161

8.1	Zitate auszeichnen mit <blockquote> und <cite>	161
8.1.1	Das HTML für Blockzitate: <blockquote> und <cite>	162
8.1.2	Ein Blockzitat mit Quellenangabe	162
8.1.3	Eine grundlegende Gestaltung für ein Zitat mit Quellenangabe	164
8.2	Einen Zeilenumbruch erzwingen mit
	165
8.3	Kontaktinformationen auszeichnen mit <address>	165
8.3.1	Eine Kontaktadresse auszeichnen mit <address>	166
8.3.2	Eine grundlegende Gestaltung für eine Kontaktadresse	166

8.4	Zeitangaben für Menschen und Maschinen: <time>	167
8.4.1	Datumsangaben mit <time>	167
8.4.2	Die Uhrzeit mit <time>	168
8.5	Ausklappbare Inhalte: <details> und <summary>	169
8.5.1	Das HTML für ausklappbare Inhalte: <details> und <summary>	169
8.5.2	Eine grundlegende Gestaltung für <details> und <summary>	171
8.6	Änderungen am Text dokumentieren: und <ins>	172
8.6.1	Das HTML für Änderungen am Text	172
8.6.2	Eine grundlegende Gestaltung für Änderungen am Text	173
8.7	Kurz vorgestellt: , <hr> und <small>	174
8.7.1	 ist ein semantisch neutrales Inline-Element	174
8.7.2	<hr> markiert einen inhaltlichen Bruch innerhalb eines Abschnitts	174
8.7.3	Das sprichwörtliche Kleingedruckte mit <small>	175
8.8	Weitere Inline-Elemente in der Übersicht	175
8.9	Know-how: Zeichensätze und Sonderzeichen	176
8.9.1	UTF-8: Wissenswertes über Zeichensätze	177
8.9.2	Die Kodierung von Sonderzeichen in HTML	177
8.10	Auf einen Blick	179

9 HTML-Elemente zum Erstellen von Formularen 181

9.1	Im Web basieren alle Interaktionen mit Besuchern auf HTML-Formularen	181
9.2	Das Element <form> definiert ein Formular	182
9.3	Einzeilige Eingabefelder mit <input> und <label>	183
9.3.1	Ein einzeiliges Eingabefeld für Text: <input type="text">	184
9.3.2	Die Beschriftung eines Eingabefeldes mit <label>	184
9.3.3	Ein Eingabefeld für E-Mail-Adressen: <input type="email">	186
9.3.4	Pflichtfelder definieren: das Attribut »required«	187
9.4	Mehrzeilige Eingabefelder mit <textarea> und <label>	188
9.5	Zum Anklicken: Kontrollkästchen, Optionsfelder und Auswahllisten	189
9.5.1	Eckige Kontrollkästchen mit <input type="checkbox">	189
9.5.2	Runde Optionsfelder mit <input type="radio">	190
9.5.3	Auswahllisten mit <select> und <option>	191

9.6	Formularfelder gruppieren mit <fieldset> und <legend>	193
9.7	Ein Button zum Abschicken der Formulardaten	194
9.8	Ein Kontaktformular mit DSGVO-Checkbox erstellen	195
9.8.1	Schritt 1: Das HTML für die Eingabefelder	196
9.8.2	Schritt 2: DSGVO-Einverständnis per Kontrollkästchen	197
9.8.3	Schritt 3: Eine grundlegende Gestaltung für das Formular	198
9.8.4	Schritt 4: Beschriftung und Formularfelder ausrichten	199
9.9	Auf einen Blick	201
10	HTML-Elemente zum Erstellen von Tabellen	203
10.1	Eine einfache HTML-Tabelle: <table>, <tr> und <td>	203
10.2	Tabellenüberschriften stehen in <th>	205
10.3	Tabellen strukturieren: <thead>, <tbody> und <tfoot>	206
10.4	Zellen verbinden mit »colspan« und »rowspan«	207
10.5	Übung: Tabelle für »Die 10 besten Alben aller Zeiten«	208
10.5.1	Schritt 1: Das HTML für die Beispieltabelle	208
10.5.2	Schritt 2: Eine grundlegende Gestaltung für die Beispieltabelle	210
10.5.3	Schritt 3: Zwischenraum und Zebrastreifen für die Beispieltabelle	211
10.6	Auf einen Blick	213
11	Von der Webseite zur Website	215
11.1	»Sie sind hier«: aktuellen Menüpunkt hervorheben	215
11.1.1	Schritt 1: Den aktuellen Menüpunkt hervorheben	216
11.1.2	Schritt 2: Die Listenelemente in der Navigation gestalten	218
11.2	Rechtliche Pflichtlinks im Footer einfügen	219
11.3	Das HTML überprüfen mit dem HTML-Validator	221
11.4	Die Seiten »News«, »Über uns« und »Kontakt« erstellen	223
11.4.1	Die Seite »News« erstellen und den Quelltext anpassen	223
11.4.2	Die Seiten »Über uns« und »Kontakt« erstellen und anpassen	225

11.5	Inhalte für die Seite »News« hinzufügen	226
11.5.1	Einen neuen Abschnitt hinzufügen: <code><section class="beitragsliste"></code>	226
11.5.2	Einen Bereich mit Linklisten erstellen: <code><aside class="linklisten"></code>	228
11.5.3	Eine grundlegende Gestaltung für die Inhalte auf der Seite »News«	229
11.6	Ein Bild auf der Seite »Über uns« einfügen	230
11.7	Die Seite »Kontakt« mit Kontaktdaten und Formular	232
11.7.1	Einen Abschnitt mit Kontaktdaten hinzufügen	233
11.7.2	Einen Abschnitt mit einem Kontaktformular hinzufügen	233
11.8	Die Seiten »Datenschutz« und »Impressum«	234
11.9	Auf einen Blick	235

TEIL III CSS – Grundlagen

12 Gestalten per CSS: Box-Modell, Farben und Einheiten 239

12.1	CSS kann man an drei verschiedenen Stellen schreiben	239
12.1.1	Externes Stylesheet: CSS-Regeln in einer eigenen Datei	240
12.1.2	»Style-Block«: CSS-Regeln mit <code><style></code> im <code><head></code> einer Webseite	240
12.1.3	»Inline-Styles«: Deklarationen mit dem Attribut »style« im Element	241
12.1.4	Empfohlenes Vorgehen: CSS möglichst in externen Stylesheets schreiben	241
12.2	Das Box-Modell kennenlernen: »padding«, »border« und »margin«	242
12.2.1	Der Inhaltsbereich einer Box sollte möglichst flexibel bleiben	243
12.2.2	Der Innenabstand »padding« schafft Platz zwischen Inhalt und Rand ...	243
12.2.3	Eine Box kann Rahmenlinien drumherum haben: »border«	244
12.2.4	Der Außenabstand »margin« regelt den Abstand zu den umgebenden Boxen	244
12.2.5	Das intuitive Border-Box-Modell: »box-sizing: border-box«	245
12.2.6	Das Box-Modell für Inline-Boxen funktioniert etwas anders	247
12.3	Farben in CSS: Farbnamen, hexadezimale Schreibweise und Transparenz	248
12.3.1	Der Aufbau eines hexadezimalen Farbwertes	249
12.3.2	Die hexadezimale Kurzschreibweise <code>#rgb</code> ist sehr praktisch	250
12.3.3	Hexadezimale Farbangaben für die Übungswebsite	251
12.3.4	Farben definieren mit <code>rgb()</code> – auch mit Transparenz	251

12.4	Die wichtigsten Längeneinheiten: px, em, rem, % & Co	253
12.4.1	Die Einheit »px«: Ein Pixel ist nicht immer ein Pixel	254
12.4.2	Die Einheit »em« hat verschiedene Berechnungsgrundlagen	255
12.4.3	Die Einheit »rem« entspricht der Standardschriftgröße des Browsers	256
12.4.4	Die Einheit »%« (Prozent) ist meist relativ zum Elternelement	256
12.5	Auf einen Blick	257

13 Das Box-Modell in Aktion: »padding«, »border« und »margin« 259

13.1	Das Box-Modell im Browser visualisieren	259
13.2	Die Breite begrenzen: »min-width« und »max-width«	261
13.3	Der Abstand zum Rand: »padding«	262
13.3.1	Das »padding« für den Kopfbereich der Seite	262
13.3.2	Das »padding« für die Navigation und den Fußbereich	264
13.3.3	Das »padding« für den Inhaltsbereich	265
13.4	Rahmenlinien gestalten mit »border« und »border-radius«	266
13.4.1	Die Eigenschaften zum Gestalten von Rahmenlinien	266
13.4.2	Abgerundete Ecken mit »border-radius«	269
13.5	Blockboxen horizontal zentrieren mit »margin«	270
13.6	Abstände zwischen den Elementen mit »margin«	272
13.7	Tipps zum Gestalten mit »padding« und »margin«	273
13.7.1	Entscheidungshilfe: Abstände mit »padding« oder »margin«?	273
13.7.2	Übersicht: Die Kurzschreibweise für »padding« und »margin«	273
13.7.3	Logische Eigenschaften für das Box-Modell: »inline« und »block«	274
13.8	Know-how: Collapsing Margins – vertikale Außenabstände kollabieren	276
13.8.1	Praktisch: Vertikale Außenabstände aufeinanderfolgender Elemente kollabieren	276
13.8.2	Problematisch: Außenabstände kollabieren nicht nur bei aufeinanderfolgenden Elementen	277
13.8.3	Ein Beispiel: Kopfbereich mit Überschrift und »Collapsing Margins«	277

13.8.4	»padding-top« für den Kopfbereich verhindert das Kollabieren der Außenabstände	279
13.8.5	Nützlich: Eine CSS-Regel zur Vermeidung von »Collapsing Margins«	280
13.9	Auf einen Blick	280
14	Die wichtigsten Selektoren und ihre Spezifität	283
14.1	Einfache Selektoren: Elemente, Gruppierung und *	284
14.1.1	»Der Name der Kiste« – einfache Elementselektoren	284
14.1.2	Mehrere Kästchen zugleich: Selektoren mit einem Komma gruppieren	284
14.1.3	Alle Kästchen auswählen: der Universalselektor »*«	285
14.2	Klassen sind klasse: der Selektor mit dem Punkt	285
14.2.1	Beispiele für den Einsatz von Klassen auf der Übungswebsite	285
14.2.2	Gebundene Klassen: Klassen auf einen Elementtyp beschränken	286
14.2.3	Ein HTML-Element kann mehrere Klassennamen haben	287
14.3	IDs sind einmalig: der Selektor mit der Raute	287
14.4	Attributselektoren haben eckige Klammern: [attribut]	288
14.4.1	Der Selektor [attribut] prüft nur, ob es das Attribut gibt	289
14.4.2	Attributselektoren mit einem Gleichheitszeichen: [attribut="wert"]	289
14.4.3	Attributselektoren mit Tilde und Gleichheitszeichen: [attribut~="wert"]	290
14.4.4	Attributselektoren mit senkrechtem Strich und Gleichheitszeichen: [attribut = "wert"]	290
14.4.5	Attributselektoren mit Hütchen und Gleichheitszeichen: [attribut^="wert"]	291
14.4.6	Attributselektoren mit Dollar und Gleichheitszeichen: element[attribut\$="zeichen"]	291
14.4.7	Attributselektoren mit Sternchen und Gleichheitszeichen: element[attribut*="zeichen"]	292
14.4.8	Zum Nachschlagen: die Attributselektoren auf einen Blick	292
14.5	DOM: die Hierarchie der HTML-Elemente	293
14.6	Nachfahren auswählen: der Selektor mit Leerzeichen	294

14.7	Selektoren zum Auswählen von Kindelementen	294
14.7.1	Der Kindselektor: der Selektor mit dem »>« (Größer-als-Zeichen)	295
14.7.2	Praktisch: Die Pseudoklassen »:first-child« und »:last-child«	295
14.7.3	Der Zauberstab zum Auswählen von Kindern: »:nth-child()«	297
14.8	Nachbarn und Geschwister selektieren mit + und ~	298
14.9	Kurz vorgestellt: der praktische Elternselektor :has()	300
14.10	Nützliche Quellen zum Nachschlagen von Selektoren	301
14.11	Spezifität: Einige Selektoren sind wichtiger als andere	302
14.11.1	Einer wird gewinnen: »Spezifität« als Punktesystem für Selektoren	302
14.11.2	Einige Beispiele für die Spezifität von Selektoren	302
14.11.3	»Spezifität« ist genau genommen eine Matrix und kein Punktesystem	303
14.12	Auf einen Blick	304
15	Text gestalten: Schriften, Listen, Links und mehr	305
15.1	Klassische Schriftarten mit und ohne Serifen im Web	306
15.1.1	Es gibt Schriftarten mit und ohne »Serifen«	306
15.1.2	Sehr praktisch: Die Schriftgestaltung in Firefox analysieren	306
15.2	Die Schriftart definieren mit »font-family«	307
15.2.1	Bitte eine Schriftart ohne Serifen mit »font-family«	307
15.2.2	Generische Schriftfamilien im Überblick	309
15.3	Die Systemschrift des Geräts: gut lesbar und echt schnell	310
15.4	Webfonts – die Schriftart gleich mitliefern	311
15.5	Schnell und einfach: Google Fonts selbst ausliefern	312
15.5.1	Schritt 1: Schriftart und Zeichensatz auswählen	313
15.5.2	Schritt 2: Die gewünschten Styles (Schriftschnitte) festlegen	314
15.5.3	Schritt 3: Den Code für die Schriftarten kopieren und einfügen	314
15.5.4	Schritt 4: Schriftdateien herunterladen und im Ordner »font« speichern	315
15.5.5	Schritt 5: Die neue Schriftart im CSS nutzen	317
15.6	Gut lesbarer Text mit »font-size« und »line-height«	317
15.6.1	Schriftgröße definieren mit »font-size« und einer Längeneinheit	318
15.6.2	Die Schriftgröße für die Überschriften ändern	318

15.6.3	Schriftgröße definieren mit »font-size« und einem Wort	319
15.6.4	Wichtig für die Lesbarkeit: Der Zeilenabstand mit »line-height«	320
15.7	Listen: Aufzählungszeichen gestalten per CSS	322
15.7.1	Die Gestaltung von Listen in den Browser-Stylesheets	322
15.7.2	Aufzählungszeichen gestalten mit »list-style« & Co	323
15.7.3	Aufzählungszeichen gestalten mit dem Pseudoelement »::marker«	324
15.8	Hyperlinks: Unterstreichen gestalten mit »text-decoration«	325
15.8.1	Zusätzliche Eigenschaften zur Unterstreich von Links	325
15.8.2	Die Unterstreich der Links gestalten	326
15.9	Hyperlinks: Linkzustände gestalten mit Pseudoklassen	327
15.9.1	Besuchte und nicht besuchte Hyperlinks mit »:link« und »:visited«	327
15.9.2	Benutzeraktionen gestalten mit »:hover«, »:focus« und »:active«	328
15.10	Links in neuem Tab kennzeichnen mit dem Pseudoelement »::after«	330
15.10.1	Schritt 1: Links auswählen mit einem Attributselektor	331
15.10.2	Schritt 2: Das Pseudoelement »::after« und die Eigenschaft »content«	331
15.10.3	Schritt 3: Links kennzeichnen mit einem Unicode-Symbol	332
15.11	Weitere Eigenschaften zur Gestaltung von Schrift und Text	333
15.11.1	Die wichtigsten Eigenschaften zur Schrift- und Textgestaltung	333
15.11.2	Schrift gestalten: fett, kursiv, Kapitälchen und Zeichenabstand	333
15.11.3	Die Kurzschreibweise »font«	334
15.11.4	Text ausrichten und die erste Zeile einrücken	334
15.11.5	Schatten im Text: »text-shadow«	335
15.12	Auf einen Blick	336

16 Der Browser und das CSS: Vererbung, Standardwert und Kaskade 339

16.1	Überblick: Vererbung, Standardwert und Kaskade	339
16.2	Nichts gefunden? Vererbung oder Standardwert	340
16.2.1	»Vererbung« macht ein Stylesheet übersichtlicher	341
16.2.2	Bestimmte Eigenschaften werden nicht vererbt	341
16.2.3	Jede CSS-Eigenschaft hat einen fest definierten »Standardwert«	342

16.3	Die Kaskade: Wichtigkeit, Spezifität und Reihenfolge	343
16.3.1	Die Kaskade ist eine Entscheidungshilfe für den Browser	343
16.3.2	Die Ausgangssituation: Das Beispiel im Überblick	343
16.3.3	Intuitiv: Die Reihenfolge im CSS entscheidet	344
16.3.4	Ungewohnt: Spezifität ist wichtiger als Reihenfolge	345
16.3.5	Ausnahme: »!important« gewinnt immer	347
16.3.6	Die Kaskade im Browser analysieren	348
16.4	Auf einen Blick	350

17 Boxen gestalten: Hintergrund, Schatten und am Bildschirm ausblenden 351

17.1	Hintergrundgrafiken per CSS einbinden und gestalten	352
17.1.1	Hintergrundgrafiken einbinden: »background-image«	352
17.1.2	Hintergrundgrafiken wiederholen: »background-repeat«	353
17.1.3	Hintergrundgrafiken positionieren: »background-position«	354
17.1.4	Hintergrundgrafiken fixieren: »background-attachment«	355
17.1.5	Die Größe der Hintergrundgrafik definieren: »background-size«	355
17.1.6	Die Kurzschreibweise: »background«	357
17.1.7	Das Box-Modell, der Hintergrund und die dritte Dimension	357
17.2	Lineare Farbverläufe: »background-image« und »linear-gradient()«	358
17.3	Schattenboxen mit »box-shadow«	360
17.4	Zitate als Kundenstimmen gestalten	362
17.4.1	Das HTML: »section« und »blockquote«	362
17.4.2	Zitate gestalten mit den Box-Modell-Eigenschaften	363
17.5	»Call to Action«: Links in Buttons verwandeln	365
17.5.1	Die Ausgangssituation: Zwei ganz normale Hyperlinks	366
17.5.2	Schritt 1: Die grundlegende Gestaltung der beiden Links	367
17.5.3	Schritt 2: Die Unterschiede – primäre und sekundäre Buttons	368
17.5.4	Schritt 3: Die Linkzustände der Buttons gestalten	369
17.5.5	Schritt 4: Einen sanften Übergang mit »transition« hinzufügen	370
17.5.6	Schritt 5: Boxen skalieren mit »transform«	371
17.6	Boxen am Bildschirm ausblenden: »visually-hidden«	372
17.6.1	Schritt 1: Die Klasse »visually-hidden« erstellen	373
17.6.2	Schritt 2: Den Elementen die Klasse »visually-hidden« zuweisen	374

17.7	»Skip-Link«: per Tastatur direkt zum Inhalt springen	375
17.7.1	Skip-Links helfen Tastaturbenutzern bei der Navigation	375
17.7.2	Schritt 1: Den Skip-Link erstellen und verstecken	375
17.7.3	Schritt 2: Den Skip-Link bei Tastaturbedienung sichtbar machen	376
17.8	Auf einen Blick	378

18 Ordnung halten: Stylesheets aufräumen und organisieren 379

18.1	Benutzerdefinierte Eigenschaften: Variablen in CSS	380
18.1.1	Schritt 1: Eine benutzerdefinierte Eigenschaft erstellen	380
18.1.2	Schritt 2: Eine benutzerdefinierte Eigenschaft als Variable anwenden	381
18.1.3	Schritt 3: Ein Farbschema mit benutzerdefinierten Eigenschaften	382
18.2	Stylesheets mit Kommentaren strukturieren	383
18.2.1	Der Kommentar am Anfang des Stylesheets	383
18.2.2	Ein Stylesheet mit Kommentaren in Abschnitte unterteilen	384
18.3	Verschiedene Schreibweisen für CSS-Regeln	384
18.3.1	Übersichtlich und weit verbreitet: Auf jeder Zeile eine Deklaration	385
18.3.2	Kurze Regeln: Alles in einer Zeile	385
18.3.3	Übersichtlich: Mehrere Selektoren auf Zeilen verteilen	386
18.3.4	Die Reihenfolge der Deklarationen in einer CSS-Regel	386
18.4	CSS überprüfen mit dem CSS-Validator	387
18.5	Modulbauweise: Ein zentrales Stylesheet erleichtert die Entwicklung	388
18.5.1	Während der Entwicklung: Modulbauweise mit mehreren Stylesheets	389
18.5.2	Für die Live-Site: Alles wieder in einem Stylesheet vereinen	389
18.6	Die Modul-Stylesheets erstellen und importieren	390
18.6.1	Schritt 1: Die einzelnen Stylesheets erstellen	390
18.6.2	Schritt 2: Stylesheets mit @import in »style.css« einbinden	391
18.7	Das CSS auf die Modul-Stylesheets verteilen	393
18.7.1	Das Modul »basis.css« ist das Fundament	393
18.7.2	Das Modul »layout.css« für Seitenlayout und Layoutbereiche	394

18.7.3	Das Modul »navi-flex.css« für die Hauptnavigation	396
18.7.4	Das Modul »content.css« zur Gestaltung der Inhalte	397
18.7.5	Das Modul »forms.css« für Formulare	398
18.8	Ein neues Modul für ein modernes Layout	398
18.8.1	Schritt 1: Das HTML anpassen – die Dopplung mit »div« im HTML	399
18.8.2	Schritt 2: Das Stylesheet »layout-modern.css« erstellen	402
18.8.3	Schritt 3: Fine-Tuning für die Infoboxen auf der Startseite	403
18.9	Auf einen Blick	405

TEIL IV CSS – Layout

19	Der normale Flow, »position« und »float«	409
19.1	Flow: Die Seite ist ein langer ruhiger Fluss	410
19.1.1	Der normale Flow im Block Formatting Context	410
19.1.2	Auch kürzere Blockboxen stehen im Flow untereinander	410
19.2	Versetzt weiterfließen mit »position: relative«	411
19.3	Raus aus dem Flow mit »position: absolute«	412
19.3.1	Absolute Positionierung hebt eine Box aus dem Flow	413
19.3.2	Ein beliebter Trick: absolute und relative Positionierung kombinieren	414
19.4	Wie ein Fels in der Brandung mit »position: fixed«	416
19.5	Scrollen und stehen bleiben mit »position: sticky«	418
19.6	Positionierte Boxen und der »z-index«	419
19.7	Text um eine Abbildung fließen lassen mit »float«	421
19.7.1	Die Ausgangssituation: ein <figure> mit Bild und Beschriftung	421
19.7.2	Das <figure>-Element nach rechts floaten mit »float: right«	422
19.7.3	Die umgebenden Elemente reichen bis unter die gefloatete Box	423
19.8	Floats beenden mit »clear: both«	424
19.9	Praktisch: Klassen zum Floaten und Clearen	425

19.10	Floats umschließen mit »display: flow-root«	426
19.10.1	Das Problem: Floats ragen nach unten aus dem Elternelement heraus	426
19.10.2	Die Lösung: Floats umschließen mit »display: flow-root«	426
19.10.3	Das Umschließen von Floats für ältere Browser mit »@supports«	427
19.11	Floats nicht rechteckig umfließen mit »shape-outside«	429
19.12	Auf einen Blick	430
20	Media Queries und responsives Webdesign	433
20.1	Responsives Webdesign: Das Web wird flexibel	433
20.2	Medientypen definieren das Ausgabemedium	434
20.2.1	Die Medientypen in der Übersicht	434
20.2.2	Eine Druckversion für eine Webseite mit »@media print« erstellen	435
20.3	Media Queries = Medientypen + Medieneigenschaften	437
20.3.1	Die Syntax: »@media Medientyp and (Eigenschaft: Wert)«	438
20.3.2	Eine Media Query zur Änderung der Hintergrund- und Schriftfarbe	438
20.3.3	Weitere Beispiele für mögliche Media Queries	440
20.3.4	Die wichtigsten Medieneigenschaften im Überblick	440
20.4	Media Queries brauchen den »Meta-Viewport«	441
20.4.1	Media Queries und die virtuelle Viewportbreite mobiler Browser	442
20.4.2	Der Meta-Viewport definiert die Viewportbreite für mobile Browser neu	442
20.5	Media Queries und die Suche nach dem Breakpoint	443
20.5.1	Weit verbreitet: Breakpoints für »Mobile«, »Tablet« und »Desktop«	443
20.5.2	Breakpoints sollte man vom Layout ableiten, nicht von Geräten	444
20.6	Responsive Schriftgröße mit und ohne Media Queries	444
20.6.1	Responsive Schriftgröße in Stufen mit Media Queries und Breakpoints	445
20.6.2	Ohne Media Queries: stufenlos größer werdende Schrift mit »clamp()«	446
20.7	Auf einen Blick	448

21	Flexbox: Mehrspaltige Layouts mit »display: flex«	449
21.1	Flexbox und Grid – jenseits vom »Block Formatting Context«	449
21.2	Flex-Container erstellen: »display: flex«	450
21.2.1	Die Ausgangsposition – eine einfache Navigation	450
21.2.2	Der erste Schritt: Flex-Container erstellen mit »display: flex«	451
21.2.3	Layouten per Flexbox: flexibel und in einer Richtung	453
21.3	Fließrichtung von Flex-Items kontrollieren: »flex-flow«	454
21.3.1	Jede Flexbox hat eine »Hauptachse« und eine »Querachse«	454
21.3.2	»flex-direction« ändert die Fließrichtung: von »row« zu »column«	455
21.3.3	»flex-wrap« ermöglicht einen Zeilenumbruch in der Flexbox	456
21.3.4	Shorthand: »flex-flow« ist kurz für »flex-direction« und »flex-wrap«	457
21.4	Flex-Items an der Hauptachse ausrichten: »justify-content«	458
21.5	Flex-Items an der Querachse ausrichten: »align-items« und »align-self«	459
21.5.1	Flex-Items an der Querachse ausrichten mit »align-items«	459
21.5.2	Einzelne Flex-Items an der Querachse ausrichten mit »align-self«	461
21.6	Automatische Abstände für Flex-Items mit »margin«	462
21.6.1	Flex-Items am Ende des Flex-Containers ausrichten mit »margin«	462
21.6.2	Elemente horizontal und vertikal zentrieren mit »margin: auto«	463
21.7	Flexibilität für Flex-Items: die Zauberformel »flex: 1«	464
21.7.1	»Lieber Browser, bitte mach alle Flex-Items gleich groß.«	464
21.7.2	»flex« ist kurz für »flex-grow«, »flex-shrink« und »flex-basis«	464
21.7.3	Überraschung: »flex-grow« in einer Flexbox mit Zeilenumbruch	465
21.8	Flexbox in Aktion: »Sticky Footer« am unteren Rand des Browserfensters	466
21.8.1	Schritt 1: <body> wird Flex-Container, die Layoutbereiche Flex-Items	466
21.8.2	Schritt 2: Die Zauberformel »flex: 1« für den Inhaltsbereich	468
21.8.3	Schritt 3: Die Navigation im Fußbereich gestalten	469
21.9	Die Reihenfolge von Flex-Items ändern	470
21.10	Auf einen Blick	472

22	CSS-Grid: Mehrspaltige Layouts erstellen mit »display: grid«	475
22.1	Ein »Grid« ist ein Raster und schafft Ordnung	475
22.2	Mehr Platz für moderne Browser: »@supports«	476
22.3	Ein Grid-Layout: Boxen neben- und untereinander	477
22.3.1	Ein Blick auf das HTML für den Abschnitt mit den Infoboxen	477
22.3.2	Schritt 1: Einen Grid-Container definieren mit »display: grid«	478
22.3.3	Schritt 2: Das Raster definieren mit »grid-template-columns« und der Einheit »fr«	480
22.3.4	Schritt 3: Den Zwischenraum kontrollieren mit »gap«	481
22.3.5	Ein Raster-Layout besteht aus Zeilen und Spalten	482
22.4	Grid-Items manuell platzieren: 1. nummerierte Linien	483
22.4.1	Ein Blick auf das HTML für den Abschnitt mit den Kundenstimmen	484
22.4.2	Grid-Container definieren und Grid-Layout erstellen	485
22.4.3	Grid-Items manuell platzieren mit Grid-Linien und »grid-column«	486
22.5	Grid-Items manuell platzieren: 2. benannte Bereiche	488
22.5.1	Die HTML-Struktur für den Inhaltsbereich der Seite »News«	489
22.5.2	Grid-Container definieren und die benannten Grid-Bereiche erstellen	489
22.6	Die Grid-Zauberformel: responsiv ohne Media Query	492
22.6.1	Das Fundament: HTML und grundlegende Gestaltung per CSS	492
22.6.2	Schritt 1: »repeat()« erzeugt mit »auto-fit« beliebig viele Spalten	494
22.6.3	Schritt 2: Die Funktion »minmax()« macht das responsive Grid perfekt	496
22.7	Verschachtelte Grids mit »subgrid«	498
22.8	Flexbox und Grid sind ein gutes Team	499
22.8.1	Beispiel: Grid für das Nebeneinander, Flexbox für die Linkausrichtung	500
22.8.2	Grid oder Flexbox? Das ist manchmal nicht so einfach	501
22.9	Auf einen Blick	502

23 Flexible Icons und responsive Bilder 505

23.1	Flexible Icons: skalierbare Symbole mit SVG	505
23.2	SVG-Icons mit als Datei einfügen	507
23.2.1	Ein SVG-Icon mit als Datei einbinden	507
23.2.2	SVG-Icons kann man in einem Code-Editor bearbeiten	508
23.3	SVG-Icons inline direkt im HTML einfügen	510
23.4	HTML und responsive Bilder	511
23.5	Unterschiedliche Bilddateien je nach Viewportbreite	512
23.5.1	Tausche X gegen W: , »srcset w« und »sizes«	513
23.5.2	»sizes« kann per Media Query die Breite des Viewports abfragen	515
23.6	Unterschiedliche Bildmotive und Dateiformate	517
23.6.1	Art direction: Mit <picture> unterschiedliche Motive ausliefern	517
23.6.2	Unterschiedliche Dateiformate servieren	519
23.7	Auf einen Blick	519

24 Eine responsive Navigation erstellen 521

24.1	Die responsive Navigation im Überblick	521
24.1.1	Die Navigation in schmalen und in breiten Viewports	521
24.1.2	Überblick: In fünf Schritten zur responsiven Navigation	523
24.2	Schritt 1: Eine vertikale Navigation für schmale Viewports	524
24.2.1	Teil 1: Das Stylesheet »navi-responsiv.css« erstellen und einbinden	524
24.2.2	Teil 2: Mobile First – die vertikale Navigation für schmale Viewports	525
24.3	Schritt 2: Eine horizontale Navigation für breitere Viewports	526
24.4	Schritt 3: Menübutton einfügen mit <template> und JavaScript	528
24.4.1	Teil 1: Mit <template> das HTML für den Button bereitstellen	528
24.4.2	Teil 2: Per JavaScript das HTML für den Button kopieren und einfügen	530
24.5	Schritt 4: Menübutton per CSS gestalten	532
24.5.1	Teil 1: Das CSS zur Gestaltung des Menü-Buttons	532
24.5.2	Teil 2: Mit dem Pseudoelement »::before« ein Icon einfügen	534

24.6	Schritt 5: Navigation per CSS ein- und ausblenden	536
24.6.1	Teil 1: Die Navigationsliste ein- und ausblenden	536
24.6.2	Teil 2: Sanfter Übergang mit »transition« und Burger-Icon austauschen	538
24.6.3	Teil 3: Die horizontale Navigation wieder sichtbar machen	539
24.7	Auf einen Blick	540
Index		541

Materialien zum Buch

Auf der Webseite zu diesem Buch stehen folgende Materialien für Sie zum Download bereit:

► **alle Übungsdateien**

Gehen Sie auf www.rheinwerk-verlag.de/5917. Klicken Sie auf den Reiter MATERIALIEN. Sie sehen die herunterladbaren Dateien samt einer Kurzbeschreibung des Dateiinhalts. Klicken Sie auf den Button HERUNTERLADEN, um den Download zu starten. Je nach Größe der Datei (und Ihrer Internetverbindung) kann es einige Zeit dauern, bis der Download abgeschlossen ist.

Einstieg in HTML und CSS

Peter Müller führt Sie Schritt für Schritt in alle Techniken ein, die Sie für eine moderne Webgestaltung brauchen. Dabei lernen Sie grundlegende Konzepte und neue Trends im Webdesign kennen. Seine anschaulichen Beispiele können Sie leicht umsetzen und auf eigene Projekte anwenden.



Erstellen Sie Webseiten mit HTML

Sie gewinnen schnell solide HTML-Kenntnisse und lernen wichtige semantische Elemente wie `<header>`, `<nav>`, `<main>` und `<footer>` kennen. Mit diesem Wissen erstellen Sie Schritt für Schritt eine komplette Übungswebsite.

Gestalten Sie Webseiten mit CSS

So geht modernes Webdesign mit CSS: flexible Einheiten, Farben, Webfonts, Schatten, Rundungen und Farbverläufe. Dabei lernen Sie quasi nebenbei wichtige CSS-Konzepte wie Spezifität, Box-Modell, Vererbung und Kaskade kennen.

Machen Sie Ihre Webseiten responsiv und barrierefrei

Mobile First! Gestalten Sie ein Layout, das sich bei genügend Platz automatisch zu einer mehrspaltigen Variante erweitert. Inkl. Media Queries, Flexbox, CSS-Grid und mobiler Navigation.



Alle Übungsdateien zum Download



Peter Müller arbeitet seit vielen Jahren als IT-Dozent und ist als Autor von HTML-, CSS- und WordPress-Büchern bekannt. Er versteht es, komplizierte Sachverhalte auf einfache und unterhaltsame Weise darzustellen.

Grundlagen

- + Schnellstart mit HTML und CSS
- + Bilder, Audio und Video
- + Formulare und Tabellen
- + Selektoren und Einheiten

Konzepte

- + Spezifität
- + Box-Modelle
- + Kaskade und Vererbung
- + Stylesheets organisieren

Gestaltung

- + Schriften, Farben und Hyperlinks
- + Responsives Webdesign
- + Flexbox und CSS Grid
- + SVG und responsive Bilder
- + Barrierefreie, mobile Navigation

