CSS Stylesheets

Stand: 04.07.2022

CSS Grundlagen

HTML ist die Basis für Webseiten; allerdings war das ursprünglich schon Anfang der 90er Jahre entstandene HTML nicht mehr allen Anforderungen gewachsen. **HTML** bietet zwar Formatierungsmöglichkeiten für Text und Ähnliches, allerdings lassen sich diese nur spezifisch für jedes Element vornehmen. Es gab kaum Möglichkeiten, den Inhalt vom Layout zu trennen und Formatierungen zentral vorzunehmen. Der zweite Nachteil von HTML wirkt genauso schwer: Die Formatierungsmöglichkeiten sind ausgesprochen begrenzt. Beispielsweise gibt es in HTML Schriftgrößen nur in sieben festgelegten Schritten. Einem Designer, der im Printbereich punktgenaue Angaben macht, ist das zu wenig.

Die Lösung für diese Probleme heißt **Cascading Style Sheets** oder kurz CSS. Diese Technologie ist wie HTML vom W3C standardisiert und besteht aus vielen Befehlen. Die Syntax ist immer gleich: Sie schreiben zuerst den Befehl und dann nach einem Doppelpunkt den Wert. Das Ganze schließen Sie mit einem Strichpunkt ab:

```
Befehl: Wert;
```

Info

Bei nur einem Befehl oder beim letzten Befehl ist der Strichpunkt nicht erforderlich, kann aber gesetzt werden.

Ein Beispiel für einen CSS-Befehl ist z.B. `font-size` für die Schriftgröße. Dort können Sie nicht nur relative Angaben vergeben, sondern auch absolute Schriftgrößen mit verschiedenen Maßeinheiten wie Punkt (`pt`) oder Zentimeter (`cm`).

```
font-size: 12pt;
```

Info

Bei der Schriftgröße ist sich die Webdesigner-Riege uneins, welche Maßeinheit zu bevorzugen wäre. Pixel und Punkt haben Nachteile, da der Internet Explorer dann nicht mehr auf Nutzereingabe die Schriftgröße verändert, Punkt ist bildschirmrespektive auflösungsabhängig. Und relative Schriftgrößen wie em sind aus Designersicht wenig genau. In diesem Artikel verwenden wir `pt`.

Nun ist nur noch die Frage, wie CSS in HTML eingebunden wird? Dafür gibt es drei Möglichkeiten:

- Als Inline-Stil in einem HTML-Tag.
- Im Kopf der HTML-Seite. Die CSS-Befehle gelten dann für ein oder mehrere Tags.
- In einer externen Datei. So ist zentrales Layout für mehrere HTML-Seiten möglich.

Alle drei Optionen finden Sie in den nächsten Abschnitten erläutert. Wenn Sie ein CSS im Kopf der **HTML-Seite** oder in einer externen Datei platzieren, müssen Sie außerdem noch die Befehle bestimmten Tags zuweisen. Dafür sind die so genannten Selektoren verantwortlich, denen wir ebenfalls einen eigenen Abschnitt widmen.

Inline-Stile

Ein Inline-Stil ist die einfachste Form, einen oder mehrere CSS-Befehle zuzuweisen. Sie fügen diese einfach im `style`-Attribut für das Tag ein, das Sie formatieren möchten. Hier ein Beispiel – Ausgangspunkt ist ein HTML-Grundgerüst:

- 1. Erstellen Sie einen Absatz in Ihrer Seite.

```
<p>Absatz mit Formatierung</p>
```

- 2. Fügen Sie dann im öffnenden <p>-Tag das Attribut `style` hinzu.
- 3. Vergeben Sie als Wert für das **Attribut** die gewünschten CSS-Befehle durch Strichpunkte getrennt:

```
<p style="font-size: 10pt; font-weight: bold;">Absatz mit Formatierung</p>
```

Den Befehl `font-weight` kennen Sie noch nicht. Er bestimmt die Dicke der Schrift. Der Wert `bold` besagt, dass die Schrift im Fettdruck dargestellt wird.

Info

Inline-Stile kommen immer dann zum Einsatz, wenn Sie die Formatierung wirklich nur lokal auf das eine HTML-Element begrenzen möchten. Soll eine Formatierung für mehrere Elemente gelten, sind CSS-Befehle im Kopf der HTML-Seite oder in einer externen Datei vorzuziehen.

Im Kopf der Seite

CSS-Befehle im Kopf der Seite sind sehr praktisch: Sie legen die Befehle einmal im <head>-Abschnitt der Seite an und weisen sie dann mit Selektoren den Tags zu.

Auch hier ein Beispiel – Ausgangspunkt ist wieder ein **HTML-Grundgerüst**.

- 1. Fügen Sie zuerst einen Block mit `<style>`– und `</style>`-Tags innerhalb der <head>-Tags ein. Also so:

```
<head>
<title>CSS im Kopf der Seite</title>
<style></style>
```

- 2. Schreiben Sie in das öffnende `<style>`-Tag, dass es sich um CSS handelt:

```
<style type="text/css">
```

- 3. Fügen Sie **HTML-Kommentare** innerhalb des `<style>`-Blocks ein.

```
<style type="text/css"><!--
--></style>
```

Diese Maßnahme verhindert bei Browsern, die mit CSS nicht klarkommen, dass die CSS-Befehle einfach ausgegeben oder interpretiert werden.

- 4. Schreiben Sie anschließend den Namen des Tags, für das Sie einen Stil anlegen möchten, z.B.:

```
<style type="text/css"><!--
p {
}
--></style>
```

Dies ist bereits ein Selektor, und zwar ein Selektor für alle <p>-Tags, d.h. alle Absätze.

- 5. Danach fügen Sie in geschweiften Klammern alle CSS-Befehle für diesen Selektor ein.

```
<style type="text/css"><!--
p {
font-size: 16pt;
font-weight: bold;
}
--></style>
```

- 6. Zum Schluss fügen Sie im Körper der **HTML-Seite** noch einen oder mehrere Absätze ohne eigene Formatierungen ein.

Hier der komplette Code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- css_kopf.html -->
<html>
<head>
<title>CSS im Kopf der Seite</title>
<style type="text/css"><!--
p {
font-size: 16pt;
font-weight: bold;
}
--></style>
</head>
<body>
<p>Absatz mit Formatierung</p>
<p>Noch ein Absatz</p>
</body>
</html>
```

Externe Datei

CSS-Befehle in einer externen Datei bedeuten das Maximum an Zentralisierung. Die externe CSS-Datei kann von beliebig vielen HTML-Dateien verwendet werden. Es steht Ihnen außerdem völlig frei, die CSS-Datei in ein eigenes Verzeichnis zu packen oder sie im gleichen Verzeichnis wie Ihre HTML-Dateien zu belassen. Für die vorliegenden Beispiele legen Sie die CSS-Datei in den gleichen Ordner wie die HTML-Datei.

Eine CSS-Datei hat im Allgemeinen die Dateiendung .css. Absolut notwendig ist das zwar nicht, empfehlenswert aber auf jeden Fall. Eine .css-Datei ist per se eine einfache Textdatei. Sie enthält ausschließlich die **CSS-Selektoren** mit den zugehörigen Befehlen, sonst nichts, also keine `<style>`-Tags oder Ähnliches.

Und so erstellen Sie eine externe CSS-Datei:

- 1. Erstellen Sie in Ihrem Editor eine neue Datei und speichern Sie diese mit der Dateiendung .css.

Als Name kommt hier style.css zum Einsatz – nicht sehr einfallsreich, aber häufig gebraucht. Sie sollten hier nur dieselben Regeln beachten wie bei HTML-Dateinamen, also idealerweise acht Zeichen und Unterscheidung zwischen Groß- und Kleinschreibung.

- 2. Schreiben Sie nun einen Selektor in die externe Datei:

```
p {
}
```

p enthält, wie oben schon gesehen, CSS-Befehle für alle <p>-Tags.

- 3. Fügen Sie dann noch **CSS-Befehle** in den geschweiften Klammern hinzu.

```
p {
font-size: 16pt;
font-weight: bold;
}
```

Nun müssen Sie die externe CSS-Datei in jeder HTML-Seite einbinden, in der Sie sie verwenden möchten.

- 1. Fügen Sie dazu das `<link>`-Tag in den `<head>`-Bereich der Seite ein:

```
<head>
<title>Externes CSS</title>
<link />
</head>
```

Dieses Tag gehört zu den Tags ohne Inhalt und wird deswegen mit einem Schrägstrich am Ende XHTML-konform geschlossen.

- 2. Ergänzen Sie im `<link>`-Tag das `rel`-Attribut mit der Angabe `stylesheet`:

```
<link rel="stylesheet" />
```

- 3. Anschließend geben Sie das `href`-Attribut mit dem Link auf die externe CSS-Datei an:

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

Der oben gezeigte Link verweist auf die Datei style.css im gleichen Ordner. Wollten Sie z.B. auf eine CSS-Datei im Unterordner css verweisen, ginge das mit `css/style.css`. Die Verlinkung folgt hier den Regeln für Links.

- 4. Zum Schluss geben Sie noch den Typ des verlinkten CSS an. Dieser ist immer `text/css`.

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

Info

Die Reihenfolge der hier erwähnten Parameter ist nicht von Bedeutung. HTML erlaubt alle Attribute eines Tags in beliebiger Reihenfolge. Der Entwickler versucht natürlich wie auch wir hier eine logische Reihenfolge einzuhalten.

Es gibt noch eine zweite Möglichkeit, externe CSS-Dateien einzubinden, nämlich den Befehl `@import`. Er hat den Nachteil, dass er nicht im Netscape Navigator 4.x und älteren Browsern funktioniert. Wenn Sie diesen allerdings ausschließen möchten, ist auch dieser Befehl einsetzbar. Hier ein Beispiel:

```
<style type="text/css">
<!--
@import url("style.css")
--></style>
```

Selektoren

Wenn Sie CSS-Befehle nicht direkt als Inline-Stil in das `style`-Attribut für ein Tag schreiben, müssen Sie deutlich machen, für welche Tags die Befehle gelten sollen. Genau diese Aufgabe übernehmen die Selektoren. CSS kennt eine Menge an Selektoren, wobei wir uns hier auf die wichtigsten beschränken, die auch in allen relevanten Browsern funktionieren.

Die erste Gruppe, die **Tag-Selektoren**, haben Sie bereits kennen gelernt. Sie schreiben den Namen des Tags als Selektor. Dann werden alle Tags dieser Art formatiert:

```
p {
font-size: 16pt;
font-weight: bold;
}
```

Mehrere Tags lassen sich mit einer Komma-separierten Liste von Tag-Selektoren gemeinsam formatieren. Das heißt, die CSS-Befehle gelten für alle Tags in der Liste. Im folgenden Beispiel gelten die Befehle für Absätze (<p>-Tags) und Elemente von Listen (`<li>`).

```
p, li {
font-size: 16pt;
font-weight: bold;
```

```
}
```

Selektoren für verschachtelte Tags erlauben Ihnen, ineinander verschachtelte Tags zielgenau zu formatieren. Die verschachtelten Tags werden dabei einfach durch Leerzeichen getrennt eingegeben. Der folgende Selektor formatiert alle **Listenelemente**, die in einer nummerierten Liste (<ol>-Tag) liegen, die wiederum innerhalb einer zweiten nummerierten Liste liegt:

```
ol ol li {
font-size: 12pt;
font-style: italic
}
```

Info

Wir haben es hier allerdings dennoch gewählt, da es in der Praxis bei weitem das häufigste ist. Hier ein vollständiges Beispiel, bei dem zuerst die erste nummerierte Liste formatiert wird, dann die darunter verschachtelte:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- css_verschachtelt.html -->
<html>
<head>
<title>Verschachtelte Selektoren</title>
<style type="text/css"><!--
ol li {
font-size: 16pt;
font-weight: bold;
}
ol ol li {
font-size: 12pt;
font-weight: normal;
font-style: italic;
}
--></style>
</head>
<body>
<ol>
<li>Element 1</li>
<li>Element 2</li>
<li>
<ol>
<li>Element 2.1</li>
<li>Element 2.2</li>
```

```
</ol>
</li>
<li>Element 3</li>
</ol>
</body>
</html>
```

Info

Wichtig ist in diesem Beispiel, dass in der verschachtelten Liste die Schriftstärke mit `font-weight` wieder auf normal gesetzt wird. Ansonsten würden die Listenelemente der verschachtelten Liste die Eigenschaft `bold` der übergeordneten Liste erben. Siehe hierzu auch den Abschnitt „Präferenzreihenfolge“.

Die Selektoren, die Sie bisher kennen gelernt haben, sind speziell auf ein oder mehrere Tags beschränkt. Wenn Sie aber ganz bestimmte Elemente Ihrer HTML-Seite mit CSS-Befehlen versehen wollen, sollten Sie zu Klassen greifen. Eine Klasse definieren Sie einmal und können Sie dann beliebig vielen Elementen zuweisen.

Und so geht es:

- 1. Zuerst definieren Sie die Klasse mit einem beliebig von Ihnen wählbaren Namen ohne Sonderzeichen.

```
.hervorgehoben {
}
```

Eine Klasse erkennen Sie immer am vorangestellten Punkt vor dem Namen.

- 2. Anschließend legen Sie die CSS-Befehle fest.

```
.hervorgehoben {
font-size: 16pt;
font-weight: bold;
color: red;
}
```

`color` färbt den entsprechenden Text. Die anderen Befehle kennen Sie schon.

- 3. Nun fügen Sie jedem Tag, das die Klasse verwenden soll, das Attribut `class` hinzu.
- 4. Als Wert des Attributs geben Sie den Klassennamen ohne vorangehenden Punkt an:

```
<p class="hervorgehoben">Absatz</p>
```

Hier ein vollständiges Beispiel, bei dem zwei **HTML-Elemente** mit der Klasse formatiert werden und zwei nicht. Zu den formatierten Elementen gehören auch Überschriften (<h1> und <h2>).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>Klassen</title>
<style type="text/css"><!--
.hervorgehoben {
font-size: 16pt;
font-weight: bold;
color: red;
}
--></style>
</head>
<body>
<h1>Überschrift</h1>
<p class="hervorgehoben">Absatz mit Inhalt</p>
<h2 class="hervorgehoben">Überschrift</h2>
<p>Zweiter Absatz mit Inhalt</p>
</body>
</html>
```

Klassen können Sie auch mit Tag-Selektoren kombinieren. Dazu schreiben Sie einfach vor den Punkt das entsprechende Tag, für das die Klasse gelten soll. Wird die Klasse für ein anderes Tag verwendet, ist sie unwirksam. Im folgenden Beispiel ließen sich mit der Klasse hervorgehoben also nur Absätze, nicht aber z.B. Überschriften formatieren:

```
p.hervorgehoben {
font-size: 16pt;
font-weight: bold;
color: red;
}
```

Info

Es gibt neben den normalen Klassen noch so genannte Pseudoklassen. Sie stehen z.B. bei Links für einen

bestimmten Zustand.

Die letzte Selektorenart, die hier zur Sprache kommen soll, sind IDs. ID steht dabei für Identifier. IDs funktionieren im Prinzip wie Klassen. Man definiert im Kopf der Seite oder in einem externen CSS eine ID. Statt bei Klassen der Punkt kommt hier das Doppelkreuz (#, auch Hash) zum Einsatz.

```
#navigation {
color: blue;
}
```

Beim Tag wird statt dem `class`-Attribut das `id`-Attribut verwendet.

```
<p id="navigation">Navigationselemente</p>
```

Info

Einen gewaltigen Unterschied gibt es zwischen Klassen und IDs. Eine ID kennzeichnet genau ein Element innerhalb eines Dokuments, ist also eindeutig. Klassen können dagegen auf beliebig viele Elemente angewendet werden. Zwar kommen Browser auch mit einer mehrfach vergebenen ID zurecht, da dies aber unter Umständen Probleme bei der JavaScript- oder serverseitigen Programmierung hervorrufen kann, sollten Sie eine ID immer pro Dokument nur einmal vergeben.

In der Praxis werden IDs hauptsächlich für eindeutig auf einer Seite nur einmal vorhandene Elemente wie Navigationsleiste, Kopfzeile, Copyright-Vermerk etc. verwendet. Wie auch Klassen, so lassen sich IDs an bestimmte Tags koppeln. Da die ID allerdings sowieso Tag-übergreifend eindeutig sein sollte, ist das eher eine Spielerei.

Präferenzreihenfolge

Sie haben mehrere Arten kennen gelernt, CSS-Befehle auf HTML-Tags anzuwenden. Diese lassen sich beliebig mischen. Daraus ergibt sich die Frage, was passiert, wenn zwei sich widersprechende CSS-Befehle einem Tag zugewiesen werden. CSS kennt hierfür eine eindeutige Reihenfolge. Diese ließe sich auch berechnen, allerdings reichen in der Praxis vier Regeln:

Einige CSS-Befehle beispielsweise zu Schriftart und Schriftfarbe werden vererbt. Ein Beispiel: Wenn Sie einem <p>-Tag eine Formatierung zuweisen, gilt die Formatierung auch für darin eingeschlossene <b>-Tags.

Vererbte CSS-Befehle werden immer von speziell dem einzelnen Tag zugewiesenen überschrieben. Bei einem Tag selbst gilt die folgende Präferenzreihenfolge:

- Klasse vor Tag-Selektor
- ID vor Klasse

- Inline-Stil vor ID

Bei gleichwertigen Stilanweisungen, die z.B. im Kopf der Seite und in einer externen CSS-Datei jeweils als Klasse festgelegt werden, entscheidet die Reihenfolge des Auftauchens im Quellcode. Immer das zuunterst im <head>-Bereich eingebundene CSS – egal ob intern oder extern – setzt sich durch.

Mit dem Nachsatz `!important` gekennzeichnete CSS-Befehle sind allen nicht damit gekennzeichneten vorzuziehen. Untereinander gelten bei `!important`-Befehlen die oben genannten drei Regeln.

```
color: blue !important;
```

und

Im CSS-Kontext gibt es zwei HTML-Tags, die auf jeden Fall noch vorgestellt werden sollten: `<div>` und `<span>`. `<div>` ist ein Blockelement, ähnlich wie ein Absatz (<p>). Allerdings besitzt `<div>` keinerlei Eigenleben, also keine eigenen Formate, wohingegen ein Absatz vor- und nachgelagert einen Abstand besitzt. `<div>`-Elemente kommen z.B. zum Positionieren mit CSS zum Einsatz.

```
<div>Text im div-Block mit <span>nicht erkennbarem span-Bereich</span>.</div>
```

Das zweite besondere Tag ist `<span>`. Hierbei handelt es sich um ein Inline-Tag, es definiert also keinen Block, sondern nur einen Bereich. Dieser Bereich ist ebenfalls nicht vordefiniert. In der Praxis wird `<span>` hauptsächlich dazu verwendet, **CSS-Formatierungen** für einen bestimmten Textbereich vorzunehmen.

Medienspezifische CSS

Der CSS-Standard bietet einige Möglichkeiten, die mit Fug und Recht als fortschrittlich zu bezeichnen sind. Dazu gehört, Stile nur bei bestimmten Medien anzuwenden. Das heißt, Sie definieren CSS-Befehle, die dann z.B. ausschließlich zum Ausdrucken benutzt werden. Das ist z.B. praktisch, wenn Sie dem Nutzer eine Druckversion Ihrer Seite angeben möchten.

Es gibt drei Möglichkeiten, einen Medientyp für Stylesheets oder einzelne CSS-Befehle anzugeben:

- Für externe CSS-Dateien im `<link>`-Tag mit dem Attribut `media`.

```
<link href="style_print.css" rel="stylesheet" type="text/css" media="print" />
```

- Für externe Stylesheets mit `@import`. Hier schreiben Sie die Medientypen einfach hinter dem URL des CSS. Mehrere Medientypen werden per Kommata getrennt:

```
@import url("style_print.css") print, braille;
```

- Mittels des `@media`-Befehls direkt im Stylesheet. Dies klappt allerdings erst mit Internet Explorer 5.5 und nicht mit 5.0. Mehrere Medientypen werden auch hier per Kommata aneinander gehängt.

```
@media print {
CSS-Befehle
}
```

Tabelle zeigt einige der wichtigsten Medientypen. Heute am meisten in der Praxis im Einsatz ist `print` für die Druckerausgabe.

Medientyp	Beschreibung
`all`	Für alle Medien. Das ist die Standardeinstellung, wenn kein Medientyp festgelegt ist.
`aural`	Stylesheet für die Sprachausgabe mit Sprachbrowsern.
`braille`	Für die Ausgabe mit Braille-Geräten (Blindenschrift).
`handheld`	Für mobile Endgeräte.
`print`	Für die Druckerausgabe.
`projection`	Für Präsentationen auf Beamern.
`screen`	Für die Ausgabe auf dem Bildschirm.
`speech`	Für die Ausgabe mit Sprachsoftware. Dieser Medientyp ist neu und ergänzt ab CSS 2.1 aural.

Für die Arbeit mit Cascading Style Sheets ist die Kenntnis einer großen Zahl von CSS-Eigenschaften

notwendig, die das Aussehen von Text, Bildern, Tabellen und Formularen steuern. Um Ihnen bei Ihrer Arbeit zu helfen, finden Sie in diesem Anhang eine Zusammenfassung der Eigenschaften und ihrer Werte, mit denen Sie Ihre eigenen Stile erstellen können. Diese Liste enthält die Standardeigenschaften von CSS 2.1, die von den meisten aktuellen Browsern unterstützt werden.

Info

Auf die Behandlung von Eigenschaften, die von (fast) keinem Brawser unterstützt werden, wurde in diesem Anhang verzichtet. Es wird in den folgenden Beschreibungen jedoch ggf. darauf hingewiesen, mit welchen Browsernn die verschiedenen Eigenschaften funktionieren. Informationen aus erster Hand finden Sie in der vollständigen CSS 2.1 – Spezifikation des W3C Spezifikation des W3C.

CSS-Werte

Jede CSS-Eigenschaft besitzt einen oder mehrere passende Werte. So benötigt die Eigenschaft `color` zur Definition der Schriftfarbe beispielsweise einen Farbwert. Die Werte können je nach Verwendungszweck der Eigenschaft natürlich unterschiedlich sein. Allgemein lassen sich diese jedoch in vier Kategorien einteilen: Farben, Längenangaben, Schlüsselwörter und URLs.

Farben

Farben können für verschiedene Eigenschaften als Wert verwendet werden, beispielsweise für Schriftfarben, Hintergründe und Begrenzungslinien (Rahmen).

Schlüsselwörter

Ein Schlüsselwort ist einfach der englische Name einer Farbe wie black oder white. Momentan gibt es 17 anerkannte Schlüsselwörter: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, pur ple, red, silver, teal, white und yellow. Zwar verstehen inzwischen so gut wie alle Browser die CSS 3-Farbwerte, jedoch sind diese noch nicht offiziell anerkannt und eine Validierung schlägt fehl. In der Zukunft werden Sie aber wohl wesentlich mehr Schlüsselwörter verwenden können. Eine vollständige Liste finden Sie unter CSS3-Color.

Farbwerte

Computermonitore verwenden eine Mischung aus rotem, grünem und blauem Licht, um daraus Farben zu erzeugen. Auf diese Weise lässt sich (fast) das gesamte Farbspektrum darstellen. So gut wie jedes Design-, Illustrations- und Grafikprogramm bietet die Möglichkeit, Farben als RGB-(Rot, Grün, Blau) oder Hexadezimalwert anzugeben, den Sie direkt für die CSS-Eigenschaften übernehmen können. In CSS gibt es neben den Schlüsselwörtern verschiedene Möglichkeiten, eine Farbe zu beschreiben:

- **Hexadezimalwerte**. Am häufigsten werden im Web hexadezimale (base 16) Werte wie `#FF0033` verwendet, um Farben anzugeben. Diese bestehen aus drei Paaren mit jeweils zwei Zeichen, die den Anteil der Farben Rot, Grün und Blau wiedergeben. Diesen sechs Zeichen wird ein Doppelkreuz

vorangestellt, damit der Webbrowser den Wert korrekt interpretieren kann.

Info

Bestehen alle drei Teilwerte aus doppelten Zeichen, können Sie den Wert auch in einer verkürzten Schreibweise angeben. #361 und #336611 sind gleichbedeutend.

- **RGB-Prozentwerte**. Sie können die Farbe ebenfalls in Form von Prozentwerten angeben, z.B. `rgb(100%, 0%, 33%)`. In den meisten Bildbearbeitungsprogrammen ist es möglich, Farben als Prozentwerte zu definieren, die Sie dann für die CSS-Eigenschaft übernehmen können.
- **Dezimalwerte**. Außerdem können Sie Farben als dezimale RGB-Werte ausdrücken. Das Format ähnelt den Prozentwerten. Allerdings verwenden Sie diesmal Zahlen zwischen 0 und 255, um die Anteile von Rot, Grün und Blau anzugeben, z.B. `rgb(255, 0, 33)`.

Alle drei Methoden funktionieren gleichermaßen gut. Aus Konsistenzgründen sollten Sie sich in einem Stylesheet allerdings für eine Methode entscheiden. In allen drei „großen“ Betriebssystemen (Windows, Mac OS X, Linux) gibt es Hilfsprogramme zur Farbwahl, mit denen Sie aus einer Palette von Millionen Farben den perfekten Ton wählen und als RGB- oder Hex-Wert angeben können. Alternativ können Sie auch einen der frei verfügbaren Farbwähler im Web verwenden, beispielsweise selfhtml-Farben.

Info

In vielen Mac-Programmen wie TextEdit können Sie den Farbwähler mit dem Tastatulbefehl cmd-Shift-C öffnen.

Längenangaben

In CSS gibt es etliche Möglichkeiten, die Schriftgröße, die Breite einer Box oder die Dicke einer Begrenzungslinie anzugeben. So kann die Schriftgröße in den Maßeinheiten Inch („Zoll“. in), Pica (pc), Punkt (pt), Zentimeter (zm), Millimeter (mm), em-Einheit (em), ex-Einheit (ex), Pixel (px) sowie in Prozentwerten angegeben werden. Obwohl es viele Optionen gibt, sind nur einige davon wirklich für die Bildschirmdarstellung geeignet. (Die Gründe finden Sie auf Hexadezimale Werte.) Die wichtigsten drei sind Pixel, em-Einheiten und Prozentwerte.

Pixelwerte

Ein Pixel ist ein einzelner Bildpunkt auf dem Computerbildschirm. Dadurch besitzen Sie mit Pixeln eine konsistente Methode, Längen und Schriftgrößen unabhängig vom Computer oder Browser anzugeben. 72 Pixel sind immer 72 Pixel. Das heißt allerdings nicht, dass die tatsächliche Länge auch immer gleich ist. Da die Monitore unterschiedliche Auflösungen haben (800 x 600, 1024 x 768, 1600 x 1200 usw.), können 72 Pixel manchmal 3 Zentimetern entsprechen, manchmal auch 5 Zentimetern. Trotzdem geben Ihnen Pixelwerte die konsistenteste Kontrolle über eine Präsentation.

Info

Bei der Verwendung von Pixelwerten gibt es allerdings einen Nachteil: Benutzer des Internet Explorer 6 können in Pixeln angegebene Schriftgrößen nicht selbst anpassen. Wenn die Schrift für manche Augen zu klein ist, kann diese im IE 6 nicht vergrößert werden. (Weitere Informationen zu Pixelwerte).

em-Einheiten

Die em-Einheit stammt ursprünglich aus der Welt der Typografie, wo sie der Höhe des großgeschriebenen Buchstabens M der verwendeten Schrift entspricht. In Webseiten entspricht eine em-Einheit der Höhe der Standardschriftgröße des Browsers, das sind normalerweise 16 Pixel. Allerdings lässt sich die Standardgröße leicht anpassen, sodass 1em für den einen Benutzer 16 Pixeln entspricht, in einem anderen Browser sind es dagegen 24 Pixel. Daher bezeichnet man em-Einheiten auch als relative Maßeinheit.

Zusätzlich zur Standardschriftgröße des Browsers können em-Einheiten Größeninformationen auch von umgebenden Tags erben. Eine Schriftgröße von 0,9 em-Einheiten (0.9em) entspricht 14 Pixeln, wenn der Browser eine Standardschriftgröße von 16 Pixeln hat. Enthält allerdings ein <p>-Tag mit 0.9em ein `<strong>`-Tag, für das ebenfalls eine Größe von 0.9em definiert wurde, ist die berechnete Schriftgröße für das `<strong>`-Tag nicht 14 Pixel, sondern nur 12 Pixel (16 x 0,9 x 0,9). Sie sollten daher bei der Verwendung von em-Einheiten die Vererbung nicht vergessen.

Prozentwerte

In CSS werden Prozentwerte zur Definition vieler Dinge eingesetzt: Textgrößen, Breite und Höhe von Elementen oder die Platzierung von Hintergrundbildern, um nur ein paar zu nennen. Auf welcher Grundlage dieser Prozentwert berechnet wird, ist abhängig von der Eigenschaft. Für Schriftgrößen wird der Prozentwert basierend auf dem vererbten Wert berechnet. Angenommen, die Standardschriftgröße beträgt 16 Pixel. Wenn Sie eine spezielle Regel erstellt haben, in der die Schrift 200 Prozent groß sein soll, wird der Text mit einer tatsächlichen Höhe von 32 Pixeln dargestellt. Prozentwerte für Breitenangaben werden dagegen basierend auf der Breite des Ansichtsbereichs oder eines anderen Vorfahren-Elements mit einer definierten Breite berechnet. Prozentwerte werden als Zahl gefolgt von einem Prozentzeichen angegeben: 100%.

Schlüsselwörter

Anstelle von Farbe oder Größe besitzen viele Eigenschaften eigene spezielle Werte, die festlegen, wie ein Element dargestellt wird. Diese Werte werden anhand von Schlüsselwörtern ausgedrückt. Mit der Eigenschaft `text-align` können Sie beispielsweise festlegen, wie Text ausgerichtet werden soll. Hierfür kann eines der vier Schlüsselwörter `right`, `left`, `center` und `justify` verwendet werden. Da sich die Schlüsselwörter von Eigenschaft zu Eigenschaft unterscheiden, müssen Sie in der Beschreibung der jeweiligen Eigenschaft nachsehen, welche Begriffe tatsächlich verwendet werden können.

Ein Schlüsselwort ist aber für alle Eigenschaften gültig: `inherit`. Hiermit können Sie festlegen, dass ein

Wert auf jeden Fall vom Eltern-Element geerbt werden soll. Mit diesem Schlüsselwort können Sie selbst vererbungsresistente Eigenschaften dazu bewegen, den Wert eines Vorfahren-Elements zu übernehmen. Wurde für einen Absatz per `text-decoration` eine Unterstreichung definiert, wird diese standardmäßig auch unter enthaltenen Kind-Elementen angezeigt. Manche Browser haben allerdings Probleme mit der durchgehenden Darstellung. Daher kann es sinnvoll sein, enthaltene Kind-Elemente die Unterstreichung mithilfe des Schlüsselworts `inherit` ausdrücklich erben zu lassen, wie hier:

```
p em, p strong {
em, strong {
text-decoration: inherit;
}
```

Auf diese Weise erhalten und ausdrücklich den gleichen Wert für `text-decoration` wie das umgebende <p>-Tag. Dadurch kann es passieren, dass hervorgehobene Elemente nun doppelt unterstrichen werden (ein guter Grund dafür, dass der Wert dieser Eigenschaft normalerweise nicht vererbt werden sollte). Ändern Sie den `text-decoration`-Wert für das <p>-Tag in `overline`, würden die – und -Tags diesen Wert automatisch übernehmen und ebenfalls durchgestrichen erscheinen.

Info

Unter- und Überstreichungen sind kein besonders nützliches Beispiel. Das liegt hauptsächlich daran, dass `inherit` kein besonders nützlicher Wert ist. Aber diese Webseite wäre kein HTML-Info, wenn wir Ihnen irgendwelche Tatsachen vorenthahen würden.

URLs

Mit einer URL als Wert können Sie auf eine andere Datei im Web verweisen. So können Sie beispielsweise der Eigenschaft `background-image` eine URL übergeben, um eine Grafikdatei als Hintergrundbild eines Elements zu definieren. Diese Technik ist praktisch, wenn Sie ein gekacheltes Bild für den Seitenhintergrund oder ein eigenes Aufzählungszeichen für Listen verwenden wollen. (Webseite und Grafiken).

In CSS geben Sie eine URL so an: `url(bilder/kachel.gif)`. Mit der folgenden Regel wird ein Bild namens title.gif für den Hintergrund einer Seite definiert:

```
body {
background-image: url(bilder/kachel.gif);
}
```

Im Gegensatz zu HTML sind in CSS die Anführungszeichen um die URL optional. Das heißt, `url("bilder/kachel.gif")`, `url('bilder/kachel.gif')` und `url(bilder/kachel.gif)` funktionieren gleichermaßen.

Info

Die URL selbst funktioniert genauso wie der Wert des `href`-Attributs für Links. Sie können also absolute URLs wie http://www.html-info.eu/images/html/webseiten_grafik_011.png, Dateiverweise relativ zum Wurzverzeichnis des Webservers wie /images/html/webseiten_grafik_011.png oder auch URLs verwenden, die relativ zur CSS-Datei interpretiert werden, wie etwa ../../images/html/webseiten_grafik_011.png. Die Details finden Sie auf Bildwiederholungen steuern.

Texteigenschaften

Mit den folgenden Eigenschaften können Sie bestimmen, wie der Text einer Webseite formatiert wird. Die meisten Eigenschaften in dieser Kategorie werden vererbt, sodass es in der Regel ausreicht, sie einem umgebenden Tag (z.B. <body> zuzuweisen. Alle enthaltenen Elemente (z.B. <p> übernehmen den definierten Wert automatisch. Auf diese Weise lassen sich schnell die Grundeinstellungen für Textfarbe, Schrifttyp und ähnliche Dinge in einer Seite festlegen.

color (vererbt)

Legt die Textfarbe fest, Da der Wert vererbt wird, reicht es aus, ihn für das <body>-Tag zu definieren. Um den Text auf der gesamten Seite rot einzufärben, genügt es in der Regel, den `color`-Wert für das <body>-Tag entsprechend festzulegen. Der Text in den enthaltenen Elementen wird nun ebenfalls rot dargestellt.

- **Werte**: Ein beliebiger Farbwert.

```
color: #FFFFB;
```

Info

Die voreingestellten Werte für Linkfarben des <a>-Tags überschreiben die vererbte Farbe. Im obigen Beispiel würden die Links innerhalb des <body>-Tags auch weiterhin in Standardblau dargestellt. Auf der Seite Zeitgemässe Navigation wird beschrieben, wie Sie die Standard-Linkfarben anpassen können.

font (vererbt)

Dies ist eine Kurzschrift-Eigenschaft, mit der Sie die folgenden Einzeleigenschaften in einer gemeinsamen Deklaration mit Werten versorgen können: `font-style`, `font-variant`, `font-weight`, `font-size`, `line-height` und `font-family`. (Die einzelnen Eigenschaften werden im Folgenden beschrieben.)

Die einzelnen Werte werden durch ein Leerzeichen voneinander getrennt. Es muss mindestens ein Wert für `font-size` und `font-family` angegeben werden. Werden weitere Werte angegeben, müssen die Werte für `font-size` und `font-family` am Schluss der Deklaration stehen. Wird für eine Einzeleigenschaft kein Wert angegeben, verwendet der Browser seinen Standardwert, wodurch zuvor vorgenommene Einstellungen möglicherweise überschrieben werden.

- **Werte**: Siehe die einzelnen Eigenschaften. Wenn Sie einen Wert für `line-height` angeben, trennen Sie diesen durch einen Schrägstrich vom Wert für die Schriftgröße, z.B. 1.25em/150%.

```
font: italic small-caps bold 1.25em/150% Arial, Helvetica, sans-serif;
```

font-family (vererbt)

Gibt an, welche Schriftfamilie für die Darstellung von Text verwendet werden soll. Normalerweise werden Schriften als Folge von drei oder vier Optionen angegeben, für den Fall, dass eine oder mehrere Schriften auf dem Rechner des Benutzers nicht installiert sind. Details finden Sie auf Textformatierung.

- **Werte**: Eine durch Kommata getrennten Liste mit Schriftnamen. Enthält der Name einer Schrift selbst Leerzeichen, muss der Name mit Anführungszeichen umgeben werden. Meistens wird als letzte Option der Liste eine allgemeine Schrift angegeben, wodurch der Browser zumindest etwas ungefähr Passendes auswählen kann, falls keine der anderen Schriften gefunden werden konnte. Mögliche Angaben für die allgemeine Schrift sind: `serif` (Serifenschrift, z.B. Times), `sans-serif` (serifenlose Schrift, z.B. Arial), `monotype` (Festbreitenschrift, z.B. Courier) `cursive` (Schrift mit geneigten Buchstaben) und `fantasy`.

```
font-family: "Lucida Grande", Arial, sans-serif;
```

font-size (vererbt)

Definiert die Schriftgröße. Diese Eigenschaft wird vererbt, was besonders bei relativen Längeneinheiten wie Prozentwerten oder em-Einheiten zu verwirrenden Ergebnissen führen kann.

- **Werte**: Eine beliebige, in CSS gültige Maßeinheit sowie die folgenden Schlüsselwörter: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`, `larger` und `smaller`. Der Wert `medium` steht hierbei für die im Webbrowser definierte Standardschriftgröße. Die tatsächlichen Werte der anderen Schlüsselwörter werden anhand dieses Werts berechnet. Der genaue Faktor, um den eine Schrift vergrößert oder verkleinert wird, ist browserabhängig, beträgt aber in der Regel 1,2. So ist `large` 1,2-mal größer als `medium`. Aufgrund der Unsicherheit in der tatsächlichen Behandlung durch den Browser bevorzugen viele Designer die Verwendung von Pixeln, em-Einheiten oder Prozentwerten.

```
font-size: 1.25em;
```

Info

Wird der Wert der Eigenschaft `font-size` von einem anderen Tag geerbt, beziehen sich die Schlüsselwörter auf den tatsachlich geerbten Wert und nicht etwa auf die Standardschriftgröße des Browsers. Immerhin wird für die Skalierung der gleiche Multiplikationsfaktor wie für die vererbte Größe verwendet (in den meisten Browsern 1,2).

font-style (vererbt)

Bestimmt, ob Text geneigt (schräg) dargestellt wird. Der Wert `normal` sorgt dafür, dass als kursiv definierter Text wieder gerade dargestellt wird. Die Optionen `italic` und `oblique` funktionieren in den meisten Fällen gleich.

- **Werte**: `italic`, `oblique`, `normal`.

```
font-style: italic;
```

font-variant (vererbt)

Sorgt dafür, dass Text in sogenannten Kapitälchen dargestellt wird, wie hier: SONNENSCHEIN UND BADESTRAND. Der Wert `normal` legt fest, dass in Kapitälchen dargestellter Text wieder in Normalschrift angezeigt werden soll.

- **Werte**: `small-caps`, `normal`.

```
font-variant: small-caps;
```

font-weight (vererbt)

Ändert das Schriftgewicht Ihres Texts. Am häufigsten wird `font-weight` verwendet, um Textteile fett darzustellen – oder um genau das zu verhindern.

- **Werte**: Insgesamt gibt es in CSS 14 verschiedene Schlüsselwörter für `font-weight`. Allerdings werden nur ein paar tatsächlich von den aktuellen Browsern unterstützt. In der Regel beschränkt sich die Verwendung auf `hold` und `normal`.

```
font-weight: bold;
```

letter-spacing (vererbt)

Vergrößert oder verringert den Buchstabenabstand durch Entfernen oder Hinzufügen von Leerraum zwischen den Buchstaben.

- **Werte**: Eine beliebige CSS-Maßeinheit, wobei em-Einheiten und Pixelwerte am häufigsten verwendet werden. In den meisten Browsern funktionieren Prozentwerte nicht. Mit einem positiven Wert wird der Abstand zwischen den Buchstaben vergrößert, mit einem negativen Wert verkleinert (die Buchstaben „kuscheln“). Mit dem Wert `normal` können Sie die Einstellung für `letter-spacing` auf den Standardwert des Browsers (0) zurücksetzen.

```
letter-spacing: -1px; letter-spacing: 2em;
```

line-height (vererbt)

Sorgt für eine Änderung des Zeilenabstands (analog zu vielen Textverarbeitungsprogrammen), also des vertikalen Abstands zweier übereinanderstehender Textzeilen (z.B. in einem Absatz). Die normale Zeilenhöhe beträgt 120 Prozent der Schriftgröße (Ganze Absätze formatieren).

- **Werte**: Die meisten in CSS gültigen Maßeinheiten. Allerdings werden Pixel und em-Einheiten am häufigsten verwendet.

```
line-height : 200%;
```

text-align (vererbt)

Steuert die Textausrichtung. Der Text wird linksbündig, rechtsbündig, zentriert oder im Blocksatz (wie dieser Absatz) am umgebenden Element ausgerichtet.

- **Werte**: `left` (links), `center` (zentriert), `right` (rechts), `justify` (Blocksatz). Bei Verwendung der Option `justify` kann es zu Problemen mit der Leserlichkeit des Texts auf dem Monitor kommen.

```
text-align: center;
```

text-decoration

Erzeugt Unter-, Über- oder Durchstreichungen von Texten oder Textteilen. Hyperlinks werden beispielsweise häufig unterstrichen dargestellt. Daher sollte man in der Regel vermeiden, Text, der kein Link ist, mit Unterstreichungen zu versehen. Meistens sind Sie besser bedient, wenn Sie Text auf andere Weise hervorheben, z.B. durch die Verwendung der Eigenschaft `font-weight`. Für Netscape-Nostalgiker gibt es außerdem den Wert `blink`. Allerdings wird in der Spezifikation ausdrücklich darauf hingewiesen, dass dieser von den Browsern nicht unterstützt werden muss.

- **Werte**: `underline`, `overline`, `line-through`, `blink`, `none`. Der Wert `none` entfernt sämtliche eventuell vorhandenen Textausschmückungen. Hiermit können Sie beispielsweise die Unterstreichungen verschwinden lassen, mit denen Links normalerweise gekennzeichnet werden. Außerdem ist es möglich, mehrere Ausschmückungen zu definieren. Schreiben Sie hierfür die gewünschten Werte (außer `none`) durch Leerzeichen getrennt hintereinander.

```
text-decoration: underline overline line-through;
```

text-indent (vererbt)

Legt fest, wie weit die erste Textzeile eines Blocks eingerückt werden soll. Wie in vielen Büchern kann mit dieser Eigenschaft die erste Zeile eines Absatzes eingerückt dargestellt werden. Möglich ist aber auch eine sogenannte hängende Einzüge, bei der die erste Zeile nach links über den folgenden Text hinausragt.

- **Werte**: Eine beliebige, in CSS gültige Maßeinheit. Pixelwerte und em-Einheiten werden am

häufigsten verwendet; Prozentwerte funktionieren hier anders als bei der Eigenschaft `font-size`. Hier basieren die Prozentwerte auf der Breite der Box, die den Text enthält. Dies kann auch die Breite des im Browserfenster sichtbaren Bereichs sein. In diesem Fall würde der Wert 50% die erste Zeile um die Hälfte des Fensters einrücken. Um eine Zeile auszurücken (für den oben genannten hängenden Einzug), verwenden Sie einen negativen Wert. Diese Technik funktioniert gut zusammen mit einem positiven Wert für die Eigenschaft `margin-left`, die die linke Seite der übrigen Zeilen um einen bestimmten Wert einrückt.

Eine detaillierte Beschreibung finden Sie auf Höhe und Breite.

```
text-indent: 3em;
```

text-transform (vererbt)

Ändert die Groß- und Kleinschreibung von Text. Auf diese Weise ist es möglich, Text oder Textteile komplett in Groß- oder Kleinbuchstaben erscheinen zu lassen oder auch den ersten Buchstaben jedes Worts großgeschrieben darzustellen.

- **Werte**: `uppercase`, `lowercase`, `capitalize`, `none`. Durch die Option `none` wird der Text so dargestellt, wie er im HTML-Quelltext steht. Angenommen, der HTML-Code enthält die Zeichenkette aBCDefg, so kann diese durch den Wert `uppercase` beispielsweise als ABCDEFG dargestellt werden, ohne dass das HTML-Dokument hierfür geändert werden müsste. Mit dem Wert `none` können Sie einen möglicherweise geerbten Wert für `text-transform` wieder zurücksetzen, sodass der Text auf dem Bildschirm wieder als aBCDefg angezeigt wird.

```
text-transform: uppercase;
```

vertical-align

Richtet die Grundlinie eines Elements im Verhältnis zur Grundlinie des umgebenden Inhalts aus. Auf diese Weise können Sie bestimmte Zeichen wie ", * oder @ ober- oder unterhalb des umgebenden Texts anzeigen lassen. Bei der Anwendung von `vertical-align` auf eine Tabellenzelle steuern die Werte die Platzierung des Inhalts innerhalb der Tabellenzelle (Vertikale und horizontale Ausrichtung).

- **Werte**: `baseline`, `sub`, `super`, `top`, `text-top`, `middle`, `bottom`, `text-bottom`, ein Prozentwert oder ein absoluter Wert (z.B. in Pixeln oder em-Einheiten). Prozentwerte werden basierend auf der Einstellung für `line-height` des Elements berechnet.

```
vertical-align: top; vertical-align: -5px; vertical-align: 75%;
```

white-space

Steuert die Darstellung von Leerzeichen (Whitespace-Zeichen) durch den Browser. Wenn zwischen zwei Wörtern mehr als ein Leerzeichen steht, z.B. „Hallo, Dave“, fasst der Webbrowser diese standardmäßig zu einem Zeichen zusammen; „Hallo, Dave“. Mit dem Wert `pre` können Sie allerdings festlegen, dass sämtliche Leerzeichen entsprechend dem HTML-Tag <pre> beibehalten werden. Außerdem werden

Textzeilen standardmäßig umbrochen, wenn das Fenster nicht breit genug ist. Um das Umbrechen zu verhindern, können Sie den Wert `nowrap` verwenden. Dadurch wird sämtlicher Text eines Absatzes auf einer einzigen Zeile dargestellt. Sie sollten also nur sehr kurze Absätze verwenden (es sei denn, Sie möchten, dass Ihre Besucher endlos nach rechts scrollen müssen, um den gesamten Text zu sehen).

- **Werte**: `nowrap`, `pre`, `normal`. Daneben gibt es noch die Werte `pre-line` und `pre-wrap`, die allerdings nur von wenigen Browsern unterstützt werden.

```
white-space: pre;
```

word-spacing (vererbt)

Funktioniert wie die Eigenschaft `letter-spacing`. Anstelle des Buchstabenabstands wird hiermit allerdings der Abstand zwischen den Wörtern gesteuert.

- **Werte**: Eine beliebige in CSS gültige Längeneinheit, wobei em-Einheiten und Pixel am häufigsten verwendet werden. Prozentwerte werden nur von wenigen Browsern unterstützt. Mit einem positiven Wert wird der Wortabstand vergrößert, mit einem negativen Wert wird der Abstand kleiner. Der Wert `normal` setzt frühere Einstellungen wieder auf den Standardwert des Webbrowsers zurück (0).

```
word-spacing: -1px; word-spacing: 2em;
```

Listen-Eigenschaften

Die folgenden Eigenschaften steuern die Darstellung von ungeordneten (`<ul>`) und nummerierten Listen (`<ol>`).

list-style (vererbt)

Mit dieser Kurzschrift-Eigenschaft können Sie die Werte der drei im Folgenden beschriebenen Eigenschaften in einer gemeinsamen Deklaration festlegen. Die einzelnen Werte werden durch ein Leerzeichen voneinander getrennt. Sie können `list-style` auch als Abkürzung für einzelne Eigenschaften verwenden, z.B. `list-style: outside` anstelle von `list-style-position: outside`. Allerdings sollten Sie bedenken, dass nicht angegebene Werte automatisch durch die Standardwerte ersetzt werden. Möglicherweise an anderer Stelle definierte Werte werden unter Umständen überschrieben. Wenn Sie sowohl einen Wert für `list-style-type` als auch einen für `list-style-image` angeben, stellt der Browser das Standard-Aufzählungszeichen nur dar, wenn das Bild nicht geladen werden konnte. Auf diese Weise können Sie sicherstellen, dass auch dann Aufzählungszeichen angezeigt werden, wenn das Bild nicht gefunden werden konnte.

- **Werte**: Sämtliche für `list-style-type`, `list-style-image` und/oder `list-style-position` gültigen Werte.

```
list-style: disc url(bilder/bullet.gif) inside;
```

list-style-image (vererbt)

Definiert ein Bild, das als Aufzählungszeichen einer ungeordneten Liste (`<ul>`) verwendet werden soll.

- **Werte**: Eine URL oder `none`.

```
list-style-image: url(bilder/bullet.gif);
```

Info

Mit der Eigenschaft `background-image` können Sie ebenfalls ein eigenes Aufzählungszeichen definieren. Zudem haben Sie hierbei mehr Kontrolle Ober die Platzierung. (Details s. CSS und <img>-Tag.

list-style-position (vererbt)

Legt fest, wo die Aufzählungszeichen der Listenelemente angezeigt werden. Diese Markierungen erscheinen standardmäßig links außerhalb (`outside`) des Texts oder innerhalb (`inside`) des Texts selbst (direkt vor dem ersten Buchstaben des Texts).

- **Werte**: `inside`, `outside`.

```
list-style: inside;
```

list-style-type (vererbt)

Legt fest, welche Art von Aufzählungszeichen für eine Liste verwendet werden soll (Kreis, Quadrat, Zahlen usw.). Theoretisch können Sie mit dieser Eigenschaft also eine ungeordnete Liste (`<ul>`, Aufzählungszeichen) in eine Aufzählungsliste (`<ol>`, Zahlen, Buchstaben) verwandeln. Allerdings funktioniert diese Methode nicht in allen Browsern (z.B. nicht im Internet Explorer fur Windows). Mithilfe des Werts `none` können Sie festlegen, dass überhaupt kein Aufzählungszeichen angezeigt werden sollen.

- **Werte**: `disc`, `circle`, `square`, `decimal`, `decimal-leading-zero`, `upper-alpha`, `lower-alpha`, `upper-roman`, `lower-roman`, `lower-greek`, `none`.

```
list-style-type: square;
```

Rahmen, Begrenzungslinien, Innen- und Außenabstände

Mit den folgenden Eigenschaften werden der Leerraum um ein Element herum sowie die Begrenzungslinien (Rahmen) definiert.

border

Mit dieser Kurzschrift-Eigenschaft können Sie um ein Element herum einen Rahmen erzeugen.

- **Werte**: Die Breite (`border-width`) der Begrenzungslinie kann in einer beliebigen in CSS gültigen Längeneinheit (außer Prozentwerten) angegeben werden.

Damit der Rahmen dargestellt wird, müssen Sie außerdem einen Rahmenstil angeben. Sie können hier sämtliche Werte der Eigenschaft `border-style` verwenden: `solid`, `dotted`, `dashed`, `double`, `groove`, `ridge`, `inset`, `outset`, `none` und `hidden`. (In Abbildung 7 auf Rahmen finden Sie eine Aufstellung der verschiedenen Stile. Die Werte `none` und `hidden` sorgen dafür, dass ein bereits definierter Rahmen nicht dargestellt wird.

Zudem können Sie eine Farbe für den Rahmen angeben. Hierfür wird ein beliebiger in CSS gültiger Farbwert verwendet (z.B. ein Schlüsselwort wie `green` oder ein Hexadezimalwert wie `#33fc44`).

```
border: 2px solid #f33;
```

border-top, border-right, border-bottom, border-left

Stellt an einer bestimmten Seite eines Elements eine Begrenzungslinie dar. Die Eigenschaft `border-top` ist beispielsweise für die Oberkante des Elements zuständig.

- **Werte**: Die gleichen wie für `border`.

```
border-left: 1em dashed red;
```

border-color

Definiert die Farbe für alle vier Begrenzungslinien.

- **Werte**: Ein beliebiger in CSS gültiger Farbwert (z.B. ein Schlüsselwort wie `green` oder ein Hexadezimalwert wie `#33fc44`).

```
border-color: rgb(255,34,100);
```

border-top-color, border-right-color, border-bottom-color, border-left-color

Funktioniert wie die Eigenschaft `border-color`. Allerdings wird die Farbe nur für jeweils eine Seite des Elements festgelegt. Mit diesen Eigenschaften können Sie beispielsweise die mit der Eigenschaft `border` festgelegten Werte überschreiben und so die Darstellung einer bestimmten Begrenzungslinie speziell anpassen, während für die übrigen Seiten die allgemeinen Einstellungen von `border` gelten.

- **Werte**: Siehe `border-color`.

```
border-left-color: #333;
```

border-style

Legt den Rahmenstil für alle vier Begrenzungslinien auf einmal fest.

- **Werte**: Eines der Schlüsselwörter `solid`, `dotted`, `dashed`, `double`, `groove`, `ridge`, `inset`, `outset`, `none` und `hidden`. Die Werte `none` und `hidden` sorgen dafür, dass ein bereits definierter Rahmen nicht dargestellt wird.

Eine Darstellung der einzelnen Stile finden Sie in Abbildung 7 auf Rahmen.

```
border-style: inset;
```

border-top-style, border-right-style, border-bottom-style, border-left-style

Diese Eigenschaften funktionieren wie `border-style`. Allerdings wird der Rahmenstil nur für jeweils eine Seite des Elements festgelegt.

- **Werte**: Siehe `border-style`.

```
border-top-style: none;
```

border-width

Definiert die Dicke der Begrenzungslinie für alle vier Seiten auf einmal.

- **Werte**: Eine beliebige in CSS gültige Maßeinheit mit Ausnahme von Prozentwerten. Am häufigsten verwendet man em-Einheiten oder Pixel.

```
border-width: 1px;
```

border-top-width, border-right-width, border-bottom-width, border-left-width

Funktionieren wie `border-width`, werden aber nur auf jeweils eine Seite des Elements angewandt.

- **Werte**: Siehe `border-width`.

```
border-bottom-width: 3em;
```

outline

Mit dieser Kurzschrift-Eigenschaft können Sie die Werte von `outline-color`, `outline-style` und `outline-width` (im Folgenden behandelt) in einer Deklaration zusammenfassen. Im Gegensatz zur Eigenschaft `border` beanspruchen mit `outline` definierte Umrisse keinen eigenen Platz (das Element wird also nicht höher oder breiter). Der Umriss wird prinzipiell an allen vier Seiten eines Elements dargestellt. Die Eigenschaft wird nicht so sehr als Designdetail benutzt, sondern vielmehr, um Dinge hervorzuheben. Die Eigenschaft `outline` funktioniert in Firefox, Safari und Opera, im Internet Explorer allerdings erst ab der Version 8.

- **Werte**: Die gleichen Werte wie für `border` mit einer Ausnahme – siehe die Beschreibung von `outline-color` unten.

```
outline: 3px solid #F33;
```

outline-color

Definiert die Farbe des Umrisses (siehe auch `outline` oben).

- **Werte**: Ein beliebiger in CSS gültiger Farbwert sowie das Schlüsselwort `invert`, mit dem die Hintergrundfarbe einfach umgekehrt wird. Wird der Umriss vor einem weißen Hintergrund dargestellt, erscheint der Umriss demnach in Schwarz. Ansonsten funktioniert `outline-color` analog zu `border-color`.

```
outline-color: invert;
```

outline-style

Gibt, wie `border-style`, den Umrissstil an, z.B. durchgehend, gestrichelt, gepunktet usw.

- **Werte**: Die gleichen wie für `border-style`.

```
outline-style: dashed;
```

outline-width

Definiert die Dicke des Umrisses. Funktioniert wie `border-width`.

- **Werte**: Eine beliebige in CSS gültige Maßeinheit mit Ausnahme von Prozentwerten. Am häufigsten verwendet man em-Einheiten oder Pixel.

```
outline-width: 3px;
```

padding

Legt fest, wie viel Leerraum (Innenabstand) zwischen dem Inhalt und einer möglichen Begrenzungslinie bleiben soll. Hiermit können Sie um Text, Bilder und andere Inhalte etwas „Luft“ schaffen. (Eine Darstellung sehen Sie in Abbildung 1 auf Boxmodell).

- **Werte**: Eine beliebige in CSS gültige Maßeinheit, z.B. Pixel oder em-Einheiten. Prozentwerte werden basierend auf der Breite des Elements berechnet. Ist eine überschrift beispielsweise ein Kind-Element des `<body>`-Tags, wird die Breite des im Browser sichtbaren Bereichs verwendet. Hat das Browserfenster beispielsweise eine Breite von 1.000 Pixeln, erzeugt der Wert 20% einen Innenabstand von 200 Pixeln Breite. Ändert der Besucher die Größe des Browserfensters, wird der Innenabstand entsprechend angepasst. Geben Sie für `padding` nur einen Wert an, gilt dieser für alle vier Seiten des Elements. Alternativ können Sie `padding` für jede einzelne Seite angeben. Die Reihenfolge ist hierbei: oben, rechts, unten, links.

```
padding: 20px; padding: 2em 3em 2.5em 0;
```

padding-top

Funktioniert wie die Eigenschaft `padding`, definiert allerdings nur den Innenabstand für die Oberseite eines Elements.

```
padding-top: 20px;
```

padding-right

Funktioniert wie die Eigenschaft `padding`, definiert allerdings nur den Innenabstand für die rechte Seite eines Elements.

```
padding-right: 20px;
```

padding-bottom

Funktioniert wie die Eigenschaft `padding`, definiert allerdings nur den Innenabstand für die Unterseite eines Elements.

```
padding-bottom: 20px;
```

padding-left

Funktioniert wie die Eigenschaft `padding`, definiert allerdings nur den Innenabstand für die linke Seite eines Elements.

```
padding-left: 20px;
```

margin

Definiert den Leerraum (Außenabstand) zwischen dem Rahmen eines Elements und dem Außenabstand eines anderen Elements (s. Abbildung 1 auf Boxmodell). Hiermit können Sie zwischen zwei Elementen (z.B. zwei Bildern oder einer Seitenleiste und dem Hauptinhalt einer Seite) Leerraum erzeugen.

Info

Vertikale Außenabstände zwischen zwei Elementen können zusammengefasst werden (`collapse`). Hierbei verwenden Browser nur den oberen oder unteren Außenabstand, wodurch der leerraum schmaler ausfallen kann als erwartet (S. Kurzschrift-Eigenschaften).

- **Werte**: Eine beliebige in CSS gültige Längenangabe, z.B. in Pixeln oder em-Einheiten. Prozentwerte werden anhand der Breite des umgebenden Elements berechnet. Ist eine Überschrift beispielsweise ein Kind-Element des <body>-Tags, wird die Breite des im Browser sichtbaren Bereichs verwendet. Hat das Browserfenster beispielsweise eine Breite von 1.000 Pixeln, erzeugt der Wert 20% für die Überschrift einen Außenabstand von 200 Pixeln Breite. Ändert der Besucher die Größe des Browserfensters, wird der Außenabstand entsprechend angepasst. Geben Sie für `margin` nur einen Wert an, gilt dieser für alle vier Seiten des Elements. Alternativ können Sie, wie bei `padding`. Einzelwerte für die Seiten angeben. Die Reihenfolge ist erneut: oben. rechts. unten. links.

```
margin: 20px; margin: 2em 3em 2.5em 0;
```

margin-top

Funktioniert wie die Eigenschaft `margin`, wobei nur für die Oberseite des Elements ein Außenabstand festgelegt wird.

```
margin-top: 20px;
```

margin-right

Funktioniert wie die Eigenschaft `margin`, wobei nur für die rechte Seite des Elements ein Außenabstand festgelegt wird.

```
margin-right: 20px;
```

margin-bottom

Funktioniert wie die Eigenschaft `margin`, wobei nur für die Unterseite des Elements ein Außenabstand festgelegt wird.

```
margin-bottom: 20px;
```

margin-left

Funktioniert wie die Eigenschaft `margin`, wobei nur für die linke Seite des Elements ein Außenabstand festgelegt wird.

```
margin-left: 20px;
```

Hintergründe

In CSS gibt es mehrere Eigenschaften für die Darstellung des Hintergrunds eines Elements. Hierzu gehört die Definition von Hintergrundfarben und -bildern sowie die Steuerung von Platzierung und Wiederholung (Kachelung) des Hintergrundbilds.

background

Mit dieser Kurzschrift-Eigenschaft können Sie die Werte der fünf Hintergrundeigenschaften wie Farbe. Hintergrundbild und Platzierung des Bilds (im Folgenden beschrieben) in einer gemeinsamen Deklaration zusammenfassen. Wenn Sie einen der Werte nicht explizit angeben, wird dieser allerdings wieder auf seinen Startwert gesetzt. Haben Sie beispielsweise festgelegt, dass ein Hintergrundbild nicht gekachelt werden soll (z.B. um es als Aufzählungszeichen zu verwenden), kann es passieren, dass diese Anweisung vom Browser ignoriert wird, denn standardmäßig werden Hintergrundbilder vertikal und horizontal wiederholt (s. Bildwiederholungen steuern).

- **Werte**: Die Werte der im Folgenden beschriebenen Hintergrundeigenschaften. Die Reihenfolge der Werte spielt keine Rolle (außer für die unten beschriebene Positionierung). Üblich ist es jedoch, die Werte in der Reihenfolge `background-color`, `background-image`, `background-repeat`, `background-attachment`, `background-position` anzugeben.

```
background: #333 url(bilder/logo.gif) no-repeat fixed left top;
```

background-attachment

Gibt an, wie sich ein Hintergrundbild verhalten soll, wenn die Seite gescrollt wird. Entweder wird das Bild mit dem restlichen Inhalt bewegt, oder es bleibt an seiner Position fixiert. Sie können beispielsweise dafür sorgen, dass ein Logo auch bei sehr langen Webseiten immer sichtbar bleibt, indem Sie `background-attachment` den Wert `fixed` geben. (In Internet Explorer 6 und seinen Vorgängern funktioniert diese Methode nur mit dem <body>-Tag.)

- **Werte**: `scroll` oder `fixed`. Der Standardwert ist `scroll`: Das Bild wird mit dem übrigen Inhalt bewegt. Der Wert `fixed` sorgt dafür, dass das Bild nicht mitbewegt wird.

```
background-attachment: fixed;
```

background-color

Versieht ein Element mit einer bestimmten Hintergrundfarbe. Der Hintergrund wird hinter möglicherweise definierten Begrenzungslinien (Rahmen) und Hintergrundbildern dargestellt. Wenn Sie Rahmenstile (Eigenschaft `border-style`) wie `dashed` oder `dotted` verwenden, kann es passieren, dass die Hintergrundfarbe in den Zwischenräumen der Begrenzungslinien „durchscheint“.

- **Werte**: Ein beliebiger Farbwert.

```
background-color: #FFF;
```

background-image

Definiert ein Bild als Hintergrund eines Elements. Andere Inhalte werden vor dem Bild dargestellt. Daher sollten Sie darauf achten, dass z.B. Text auch bei Überschneidungen mit dem Hintergrundbild weiterhin lesbar bleibt. Alternativ können Sie den Inhalt mithilfe von Innenabständen vom Bild wegbewegen. Standardmäßig wird das Bild horizontal und vertikal wiederholt (gekachelt), bis der gesamte sichtbare Bereich des Elements ausgefüllt ist. Dieses Verhalten können Sie mit der Eigenschaft `background-repeat` (s.u.) steuern oder auch unterbinden.

- **Werte**: Die URL des zu verwendenden Bilds.

```
background-image: url(bilder/foto.jpg); background-image:
url(http://www.beispiel.org/foto.jpg);
```

background-position

Steuert die Platzierung des Hintergrundbilds. Standardmäßig wird das Bild in der linken oberen Ecke des Elements platziert. Wird das Bild gekachelt (s.u.). steuert `background-position` den Startpunkt der Bildwiederholungen. Platzieren Sie ein gekacheltes Hintergrundbild in der Mitte des Elements, wird das Bild auf beiden Seiten sowie ober- und unterhalb des Startbilds wiederholt. In den meisten Fällen hat eine exakte Platzierung eines gekachelten Bilds keine visuellen Auswirkungen. Interessant wird es dagegen, wenn Sie eigene Aufzählungszeichen und spezielle visuelle Effekte verwenden wollen.

- **Werte**: Eine beliebige in CSS gültige Maßeinheit. Schlüsselwörter und Prozentwerte. Die Werte werden normalerweise paarweise angegeben, wobei der erste Wert die horizontale und der zweite Wert die vertikale Position festlegt. Diese Schlüsselwörter sind für die horizontale Positionierung: `left` (links), `center` (mittig) und `right` (rechts). Für die vertikale Positionierung gibt es: `top` (oben), `center` (mittig) und `bottom` (unten). Entfernungen in Pixeln und em-Einheiten werden von der linken oberen Ecke des Elements aus berechnet. Um ein Hintergrundbild 5 Pixel nach links und 10 nach unten zu verschieben, könnten Sie also die Angabe 5px lOpx verwenden.

Prozentwerte beziehen sich gleichzeitig auf einen Punkt des Elements sowie des Hintergrundbilds selbst und werden von der linken oberen Ecke des Elements/ des Bilds aus angegeben. Die Angabe 50% 50% platziert das Bild also an einem Punkt, an dem die Bildmitte mit der Mitte des Elements zur Deckung kommt (s. Prozentwerte). Diese Werte lassen sich auch mischen. So können Sie beispielsweise die horizontale Platzierung in Pixeln angeben und für die vertikale Position einen Prozentwert nennen.

```
background-position: left top; background-position: 1em 3em; background-position: 10px 50%;
```

background-repeat

Legt fest, ob und wie ein Hintergrundbild wiederholt (gekachelt) wird. Standardmäßig wird ein Hintergrundbild von der linken oberen Ecke des Elements (für das das Bild definiert wurde) aus horizontal und vertikal wiederholt, bis der gesamte sichtbare Bereich des Elements ausgefüllt ist.

- **Werte**: `repeat`, `no-repeat`, `repeat-x`, `repeat-y`. Die Option `repeat` entspricht dem Standardverhalten: einer vertikalen und horizontalen Kachelung des Bilds. Der Wert `no-repeat` stellt das Hintergrundbild nur einmal dar, ohne es zu kacheln. Der Wert `repeat-y` wiederholt das Bild nur vertikal, sprich, von oben nach unten, und ist damit ideal für Seitenleisten geeignet. Der Wert `repeat-x` wiederholt das Hintergrundbild in der horizontalen Ebene, wodurch Sie beispielsweise eine spezielle Trennlinie für ein Element realisieren können.

```
background-repeat: no-repeat;
```

Eigenschaften für das Seitenlayout

Die folgenden Eigenschaften steuern die Platzierung und Größe von Elementen auf einer Webseite.

bottom

Diese Eigenschaft wird mit absoluter (`absolute`), fester (`fixed`) und relativer (`relative`) Positionierung genutzt. Bei Verwendung von absoluter oder fester Positionierung legt der Wert von `bottom` fest, wie weit das Element von der Unterkante des (in der Vererbungshierarchie) nächsten positionierten Vorfahren-Elements entfernt ist. Besitzt das Element keine positionierten Vorfahren, bezieht sich die Angabe auf die Unterkante des im Browser sichtbaren Bereichs („Viewport“). Mit dieser Eigenschaft können Sie beispielsweise eine Fußnote an der Unterkante des Ansichtsbereichs platzieren. Bei Verwendung von relativer Positionierung wird die Unterkante des positionierten Elements selbst als Referenz genommen. Nähere Informationen finden Sie auf Einstein und die absolute Positionierung.

- **Werte**: Eine beliebige in CSS gültige Maßeinheit, z.B. Pixel, em-Einheiten oder Prozentwerte. Prozentwerte werden basierend auf der Breite des umgebenden Elements berechnet.

```
bottom: 5em;
```

Info

Internet Explorer 6 und seine Vorgänger haben möglicherweise Probleme, wenn ein Element per `bottom` positioniert wird.

clear

Verhindert, dass ein als Float definiertes Element umflossen wird. Stattdessen wird das mit `clear` markierte Element unterhalb des Floats dargestellt.

- **Werte**: `left`, `right`, `both`, `none`. Die Option `left` sorgt dafür, dass nur mit `float: left;` definierte Elemente nicht umflossen werden. Das Gleiche gilt entsprechend für den Wert `right` und mit `float: right;` definierte Elemente. Verwenden Sie den Wert `both`, wird das Umfließen für beide Arten von Floats verhindert. Durch die Angabe von `none` können Sie zuvor mit `clear` markierte Werte dazu bringen, Floats wieder zu umfließen. Das kann praktisch sein, wenn das Umfließen nur in einem bestimmten Fall verhindert werden soll.

Die hierfür verwendete Regel muss eine höhere Spezifität aufweisen als die Regel, in der das Umließen definiert wurde (Die Kaskade unter Kontrolle behalten).

```
clear: both;
```

clip

Erzeugt einen rechteckigen Ausschnitt, der nur einen Teil eines Elements zeigt. Vielleicht wollen Sie Ihr altes Schulabschlussfoto ins Netz stellen – allerdings ohne den Klassentyrann. In diesem Fall können Sie das Bild per `clip` so beschneiden, dass nur der tyrannenfreie Teil angezeigt wird. Ihr Peiniger dagegen bleibt unsichtbar. Am effektivsten ist die Eigenschaft `clip`, wenn der Bildausschnitt per JavaScript animiert wird, etwa um anfangs nur einen kleinen Ausschnitt zu zeigen, der langsam vergrößert wird.

- **Werte**: Die Koordinaten für einen rechteckigen Bereich in Form von in CSS erlaubten Längenangaben oder das Schlüsselwort `auto`. Die Koordinaten werden mit einem paar runder Klammern umgeben, denen das Schlüsselwort `rect` vorangestellt wird, wie hier: `rect(5px,110px,40px,10px);`.

Die vier Werte beziehen sich auf die vier Ecken des Beschneidungsrechtecks. Sie werden in der Reihenfolge oben, rechts, unten, links von der linken oberen Ecke des Elements aus gemessen, für das die Beschneidung definiert wird. Mit den oben genannten Werten können Sie also eine Beschneidung mit einer Höhe von 35 Pixeln (40px – 5px) und einer Breite von 100 Pixeln (110px – 10px) definieren, die 5 Pixel von der Oberkante und 10 Pixel von der linken Seite des Elements beginnt.

```
clip: rect(5px, 110px,40px,10px); clip: rect(10px, auto. auto,10px);
```

Info

Da die Reihenfolge der Koordinaten etwas seltsam ist, verwenden die meisten Designer den ersten und den letzten Wert als Berechnungsgrundlage für die anderen beiden Angaben.

display

Definiert die sogenannte Darstellungsrolle eines Elements, z.B. das Block- oder Inline-Element (s. Inline- und Block-Elemente darstellen). Hiermit können Sie das Standardverhalten des Browsers für die Darstellung bestimmter Elemente verändern. Das kann beispielsweise praktisch sein, wenn der Link in einer Navigationsleiste im gesamten Bereich anklickbar sein soll, wie auf Navigationsleisten erstellen beschrieben.

- **Werte**: `block`, `inline`, `none`. Tatsächlich gibt es 17 verschiedene Werte für diese Eigenschaft, die aber nicht von allen Browsern konsistent unterstützt werden. So wird die Verwendung von `display` im Internet Explorer offiziell erst ab der Version 8 vollständig unterstützt. Die häufigsten und von allen aktuellen Browsern verstandenen Werte sind `block`, `inline` und `none`. Der Wert `block` sorgt dafür, dass vor und nach dem Element wie bei anderen Block-Elementen auch (z.B bei Absätzen und Überschriften) die Zeile umbrochen wird. Der Wert `inline` bewirkt, dass sämtlicher Inhalt des Elements auf der gleichen Zeile wie die umgebenden Elemente angezeigt wird (wie z.B. Text, der mit Tags wie `<strong>` oder `<em>` markiert wurde).

Der Wert `none` entfernt das Element komplett aus der Darstellung der Seite. Danach können Sie das Element mit der Pseudoklasse `:hover` oder etwas JavaScript-Code wieder zum Vorschein bringen (s. Pseudoklassen und Pseudoelemente)

```
display: block;
```

float

Definiert ein Element als sogenannten Float. Als Float definierte Elemente werden an den linken oder rechten Rand des umgebenden Elements (bzw. der Seite) verschoben. Sie sind vom normalen Textfluss des Dokuments ausgenommen und werden vom übrigen Inhalt der Seite (bzw. des umgebenden Elements) „umflossen“. Das heißt, Elemente, die im Quelltext der Seite auf das Float folgen, werden nach oben verschoben und nehmen den entstandenen Leerraum ein, wie beispielsweise der Fließtext in einem Zeitungsartikel. Mit der Eigenschaft `clear` (s.o.) können die auf das Float folgenden Elemente am Umfließen gehindert werden.

Mit Floats lassen sich einfache Effekte wie das Verschieben eines Bilds an den Seitenrand, aber auch komplexe Layouts (s. Float und CSS) realisieren.

- **Werte**: `left`, `right`, `none`. Der Wert `none` sorgt dafür, dass die Definition als Float fur ein Element wieder aufgehoben wird. Das kann praktisch sein, wenn ein Tag in einer allgemeinen Regel als Float definiert wurde. Mithilfe einer CSS-Definition mit höherer Spezifität können Sie diese Einstellung für spezielle Tags überschreiben.

```
float: left;
```

height

Legt die Höhe des Inbaltsbereichs eines Elements fest. Dies ist der Teil einer Box, in dem Text, Bilder oder auch andere Tags enthalten sind. Um die tatsächliche Höhe eines Elements zu ermitteln, müssen zum Wert von `height` die Werte für die oberen und unteren Innen- und Außenabstände sowie die Dicke der Begrenzungslinien (Rahmen) hinzuaddiert werden.

- **Werte**: Eine beliebige in CSS gültige Maßeinheit, wie z.B. Pixel, em-Einheiten oder Prozentwerte. Letztere werden basierend auf der Höhe des umgebenden Elements berechnet.

```
height: 50%;
```

Info

Es kann passieren, dass der Inhalt mehr Platz beansprucht als die per `height` angegebene Höhe – beispielsweise wenn ein Element besonders viel Text enthält oder der Benutzer die Textgröße im Browser erhöht. Browser gehen mit diesem Problem unterschiedlich um: IE 6 und seine Vorgänger vergrößern die Box einfach nach Bedarf; andere Browser stellen den Text standardmäßig auch außerhalb der Box dar. Mithilfe von `overflow` können Sie dieses Verhalten steuern, indem Sie z.B. Rollbalken erzeugen (s. Textüberlauf im Griff mit der Eigenschaft overflow).

left

Bei der Verwendung von absoluter oder fester Positionierung (CSS-Eigenschaften und Positionierung) ermittelt diese Eigenschaft die Position der linken Seite eines Elements anband der Platzierung des nächsten positionierten Vorfahren-Elements. Gibt es kein solches Vorfahren-Element, wird als Referenz die linke Kante des sichtbaren Bereichs des Browserfensters verwendet. Mit dieser Eigenschaft können Sie beispielsweise ein Bild um 20 Pixel vom Rand des Browserfensters verschieben. Bei der Verwendung von relativer Positionierung wird der Wert anhand der linken Kante des Elements selbst ermittelt (vor der Positionierung).

- **Werte**: Eine beliebige in CSS gültige Maßeinheit, wie Pixel, em-Einheiten oder Prozentwerte.

```
left: 5em;
```

max-height

Definiert die maximal erlaubte Höhe für ein Element. Die Box des Elements darf also niedriger sein als der angegebene Wert, aber keinesfalls höher. Braucht der Inhalt der Box mehr Platz, als mit `max-height` angegeben, läuft er aus der Box heraus. Anhand der Eigenschaft `overflow` (Textüberlauf im Griff mit der Eigenschaft overflow) können Sie steuern, wie der „überflüssige“ Text dargestellt werden soll. Internet Explorer 6 (und dessen Vorgänger) unterstützen `max-height` nicht.

- **Werte**: Eine beliebige in CSS gültige Maßeinheit wie Pixel, em-Einheiten oder Prozentwerte. Prozentwerte werden basierend auf der Höhe des Elements berechnet.

```
max-height: 100px;
```

max-width

Definiert die maximal erlaubte Breite für ein Element. Ist der Inhalt eines Elements breiter als der mit `max-width` angegebene Wert, fließt dieser über die Grenzen der Box hinaus. Mithilfe der Eigenschaft `overflow` können Sie steuern, wie dieser Inhalt dargestellt wird. Am häufigsten wird `max-width` für „flüssige“ Layout“ (Grundlagen des Float-basierten Layouts) verwendet, damit ein Seitendesign auch auf sehr großen Monitoren noch lesbar bleibt. Diese Eigenschaft wird von Internet Explorer 6 und seinen Vorgängern nicht unterstützt.

- **Werte**: Eine beliebige in CSS gültige Maßeinheit wie Pixel, em-Einheiten oder Prozentwerte. Prozentwerte werden basierend auf der Breite des Elements berechnet.

```
max-width: 950px;
```

min-height

Legt die minimale Höhe für ein Element fest. Die Box des Elements darf höher sein als diese Einstellung, aber keinesfalls kleiner. Ist der Inhalt der Box niedriger als der mit `min-height` definierte Wert, wird die Höhe der Box so weit erweitert, bis die angegebe Höhe erreicht ist. Diese Eigenschaft wird vom Internet Explorer 6 und seinen Vorgängern nicht unterstützt.

- **Werte**: Eine beliebige in CSS gültige Maßeinheit wie Pixel, em-Einheiten oder Prozentwerte. Prozentwerte werden basierend auf der Höhe des Elements berechnet.

```
min-height: 20em;
```

min-width

Definiert die minimale Breite eines Elements. Die Breite eines Elements darf höher sein als dieser Wert, keinesfalls aber geringer. Ist der Inhalt des Elements schmaler als der Wert von `min-width`, wird die Breite der Box entsprechend verringert. Sie können `min-width` auch in flüssigen Layouts verwenden, um sicherzustellen, dass das Layout auch bei geringen Fensterbreiten nicht „auseinanderfällt“. Ist das Browserfenster schmaler als der Wert von `min-height` für das <body>-Element, erzeugt der Browser automatisch horizontale Rollbalken, um den Inhalt zugänglich zu machen. Diese Eigenschaft wird vom Internet Explorer 6 und seinen Vorgängern nicht unterstützt.

- **Werte**: Eine beliebige in CSS gültige Maßeinheit wie Pixel, em-Einheiten oder Prozentwerte. Prozentwerte werden basierend auf der Breite des Elements berechnet.

```
min-width: 760px;
```

Info

In der Regel werden `max-width` und `min-width` gemeinsam bei der Erstellung flüssiger Layouts

verwendet. Details finden Sie in Elemente mit negativen Aussenabständen positionieren.

overflow

Mit dieser Eigenschaft können Sie steuern, was mit Inhalt geschehen soll, der nicht vollständig im Inhaltsbereich des Elements dargestellt werden kann (z.B. weil die Werte für `max-width`, `max-height` oder `clip` dies verhindern).

Info

IE 6 (und früher) behandeln diese Situationen anders als standardkonforme Browser. Details hierzu finden Sie auf Textüberlauf im Griff mit der Eigenschaft overflow.

- **Werte**: `visible`, `hidden`, `scroll`, `auto`. Der Standardwert `visible` sorgt dafür, dass Inhalt, der außerhalb der Box dargestellt werden muss. sichtbar bleibt. Hierdurch kann es allerdings passieren, dass Begrenzungslinien (Rahmen) und andere Seitenelemente mit dem Inhalt der Box überlagert werden. IE 6 und seine Vorgänger vergrößern in diesem Fall einfach die Box, bis der Inhalt hineinpasst. Der Wert `hidden` versteckt die Teile des Inhalts, die über die Grenzen der Box hinausragen würden. Mit `scroll` wird der Browser angewiesen, das Element prinzipiell mit Rollbalken zu versehen, wodurch der versteckte Inhalt zugänglich bleibt. Durch den Wert `auto` werden die Rollbalken nur angezeigt, wenn dies auch nötig ist.

```
overflow: hidden;
```

position

Legt fest, auf welche Art der Browser ein Element auf der Seite positionieren soll.

- **Werte**: `static`, `relative`, `absolute`, `fixed`, `static` entspricht dem Standardverhalten des Browsers. Die Blockelemente werden in der Reihenfolge des HTML-Quellcodes nach- und übereinander auf der Seite dargestellt. Relativ positionierte Elemente werden um einen bestimmten Wert von ihrem ursprünglichen Platz in der Seite verschoben. Dieser wird nicht von anderen Elementen eingenommen, sondern bleibt frei. Der Wert `absolute` nimmt ein Element komplett vom Textfluss der Seite aus; dieses kann mit den Eigenschaften `left`, `right`, `top` und `bottom` (s. Hinweis) an einer bestimmten Position auf der Seite platziert werden. Absolut positionierte Elemente können andere Teile der Seite überlagern. Außerdem können Sie ein Element `relativ` zu einem mit `absolute`, `relative` oder `fixed` positionierten Element platzieren. Die Einstellung `fixed` sorgt dafür, dass ein derart positioniertes Element (ähnlich den HTML-Frames) an einer bestimmten Position auf der Seite sichtbar bleibt, auch wenn diese gescrollt wird. Der Wert `fixed` wird vom Internet Explorer 6 (und früher) nicht unterstützt.

```
position: absolute;
```

Info

Normalerweise verwenden Sie mit `relative`, `absolute` und `fixed` definierte Positionierungen zusammen mit den Eigenschaften `left`, `right`, `top` und `bottom`. Die Details finden Sie in CSS und

Webseiten ausdrucken.

right

Bei der Verwendung von absoluter oder fester Positionierung (Wie CSS-Eigenschaften für die Positionierung funktionieren) definiert diese Eigenschaft die Entfernung zur rechten Kante des nächsten positionierten Vorfahren-Elements oder des Browserfensters (falls es kein positioniertes Vorfahren-Element gibt). Hiermit können Sie beispielsweise eine Seitenleiste um einen bestimmten Wert von der rechten Kante des Ansichtsbereichs entfernt darstellen. Bei der Verwendung von relativer Positionierung wird als Referenz die rechte Kante des positionierten Elements selbst (vor der Verwendung von `right`) benutzt.

- **Werte**: Eine beliebige in CSS gültige Maßeinheit wie Pixel, em-Einheiten oder Prozentwerte.

```
right: 5em;
```

Info

Internet Explorer 6 (und seine Vorgänger) haben gelegentlich Probleme bei der Positionierung von Elementen mit der Eigenschaft `right`.

top

Macht das Gegenteil der Eigenschaft `bottom`. Bei der Verwendung von absoluter oder fester Positionierung definiert der Wert dieser Eigenschaft, wie weit die Oberkante des Elements von der Oberkante des nächsten positionierten Vorfahren-Elements bzw. des Ansichtsbereichs entfernt ist. Mit dieser Eigenschaft können Sie beispielsweise ein Logo um einen bestimmten Abstand von der Oberkante des Ansichtsbereichs entfernt platzieren. Bei der Verwendung von relativer Positionierung wird die Entfernung von der Oberkante des Elements selbst aus berechnet (bevor es positioniert wird).

- **Werte**: Eine beliebige in CSS gültige Maßeinheit wie Pixel, em-Einheiten oder Prozentwerte.

```
top: 5em;
```

visibility

Legt fest, ob ein Element sichtbar sein soll. Hiermit können Sie Teile einer Seite, etwa einen Absatz oder ein -Tag. „unsichtbar“ machen. Im Gegensatz zur Deklaration `display: none;` wird bei der Deklaration `visibility: hidden;` das Element nicht vom Textfluss der Seite ausgenommen. Das heißt, an der Stelle, an der das Element eigentlich stehen soll, bleibt eine Lücke. Aus diesem Grund wird die Eigenschaft `visibility` meistens für absolut positionierte Elemente verwendet, die sowieso vom normalen Textfluss ausgenommen sind.

Das Verstecken eines Elements bringt nur etwas, wenn Sie es auch wieder zum Vorschein bringen können. Häufig wird hierfür JavaScript verwendet. Alternativ können Sie den Wert von `visibility` aber auch über die Pseudoklasse `:hover` (Den Link-Zustand verstehen) ändern, wenn der Besucher den Mauszeiger über einen bestimmten Teil der Seite bewegt.

- **Werte**: `visible`, `hidden`. Mithilfe des Werts `collapse` können Sie außerdem die Zeilen oder Spalten einer Tabelle ein- und ausblenden.

```
visibility: hidden;
```

width

Definiert die Breite des Inhaltsbereichs eines Elements (den Teil, der den Text, das Bild oder auch andere Tags enthält). Der tatsächlich fur ein Element benutzte Bereich hängt davon ab, ob seitliche Innen- oder Außenabstände bzw. Begrenzungslinien (Rahmen) definiert wurden. IE 6 und seine Vorgänger gehen eigene Wege, wenn der Inhalt breiter ist als der angegebene Wert (s. Textüberlauf im Griff mit der Eigenschaft overflow).

- **Werte**: Eine beliebige in CSS gültige Maßeinheit wie Pixel, em-Einheiten oder Prozentwerte.

```
width: 250px;
```

z-index

Mit dieser Eigenschaft können Sie die Reihenfolge steuern, in der Elemente, deren position-Wert `absolute`, `relative` oder `fixed` ist, vor- oder hintereinander dargestellt werden (Wie CSS-Eigenschaften für die Positionierung funktionieren). Elemente mit einem höheren Wert fur `z-index` erscheinen näher am Benutzer.

- **Werte**: Ein ganzzahliger Wert wie 1, 2 oder 10. Auch negative Werte sind möglich, allerdings werden diese von den Browsern unterschiedlich interpretiert. Je größer die Zahl, desto weiter rückt das Element nach vorne. Ein Element mit einem `z-index` von 20 erscheint daher hinter einem Element, dessen `z-index` den Wert 100 hat (wenn sich die Elemente überschneiden). Befindet sich das Element innerhalb eines anderen positionierten Elements, benutzt es den „Positionierungskontext“ des umgebenden Elements und wird möglicherweise nicht vor einem anderen Element dargestellt, egal wie hoch der `z-index` ist.

```
z-index: 12;
```

Info

Die Reihenfolge der Werte muss nicht in ganzzahligen Schritten ertolgen. Hat Element A den `z-index 1`, muss das folgende Element B nicht zwingend den Wert 2 bekommen, um es vor Element A darzustellen. Sie können als Wert auch 5, 10 oder etwas anderes verwenden, um den gleichen Effekt zu erzielen. Hauptsache, die Zahl ist größer. Wenn Sie später neue Elemente in die Seite einbauen, brauchen Sie auf diese Weise nicht die Werte aller anderen Elemente zu ändern. Um sicherzustellen, dass ein Element immer vor anderen Elementen dargestellt wird, geben Sie ihm einfach einen sehr hohen Wert für `z-index`, z.B. 10000.

Tabelleneigenschaften

In CSS gibt es ein paar Eigenschaften, die nur für die Darstellung von HTML-Tabellen zuständig sind. Vollständige Anweisungen für die Benutzung finden Sie in Tabellen.

border-collapse

Legt fest, ob die Tabellenzellen jeweils mit einem eigenen Rahmen versehen werden oder ob die Rahmen zu einem Gitter zusammengefasst werden. Hat jede Zelle einen eigenen Rahmen, fügt der Browser zwischen den Zellen etwas Leerraum ein. Wenn Sie diesen Leerraum über das HTML-Attribut `cellspacing` auf null setzen, werden alle Rahmen dargestellt. Die untere Begrenzungslinie einer Zelle schließt beispielsweise direkt an die obere Begrenzungslinie der nächsten Zelle an, wodurch die Rahmen verdoppelt erscheinen. Erhält `border-collapse` den Wert `collapse`, verschwinden nicht nur die Zwischenräume zwischen den Zellen, sondern auch die doppelten Rahmen (Zellen und Spalten gestalten). Diese Eigenschaft funktioniert nur, wenn sie auf das `<table>`-Tag angewendet wird.

- **Werte**: `collapse`, `separate`.

```
border-collapse: collapse;
```

border-spacing

Definiert, wie groß der Zwischenraum zwischen den einzelnen Zellen sein soll. Diese Eigenschaft soll das HTML-Attribut `cellspacing` ersetzen. Der Internet Explorer versteht `border-spacing` allerdings erst ab Version 8. Daher ist es unter Umständen nötig, weiterhin mit `cellspacing` zu arbeiten, damit der Zwischenraum in allen Browsern gleich dargestellt wird.

Info

Wenn Sie den Zwischenraum entfernen wollen, den Browser standardmäßig zwischen den einzelnen Zellen darstellen, setzen Sie einfach die Eigenschaft `border-collapse` (s.o.) auf den Wert `collapse`.

- **Werte**: Zwei in CSS gültige Längenangaben. Der erste Wert definiert den horizontalen Abstand (den seitlichen Leerraum), der zweite Wert ist für die vertikale Trennung (den Leerraum ober- und unterhalb einer Zelle) zuständig.

```
border-spacing: 0 10px;
```

caption-side

Wird diese Eigenschaft auf eine Tabellenbeschriftung angewendet, legt sie fest, ob die Beschriftung über oder unter der Tabelle angezeigt wird. (Da das `<caption>`-Tag gemäß den HTML-Regeln direkt auf das öffnende `<table>`-Tag folgen muss, würde die Beschriftung ansonsten prinzipiell über der Tabelle erscheinen.)

- **Werte**: `top`, `bottom`.

```
caption-side: bottom;
```

Info

Leider wird diese Eigenschaft vom Intemet Explorer erst ab Version 8 unterstützt. Daher ist es unter Umständen sicherer, die Darstellung der Beschriftung mit dem entsprechenden HTML-Attribut zu steuem: `<caption align="bottom">` oder `<caption align="top">`.

empty-cells

Diese Eigenschaft steuert, ob ein Browser vollkommen leere Tabellenzellen (`<td></td>`) darstellen soll. Mit dem Wert `hide` können Sie diese Zellen von der Darstellung ausnehmen. Stattdessen ist nur ein leerer Platzhalter zu sehen. Eventuell definierte Rahmen, Hintergrundbilder oder -farben werden nicht dargestellt. Diese Eigenschaft wird auf das `<table>`-Tag angewendet.

- **Werte**: `show`, `hide`.

```
empty-cells: show;
```

Info

Hat die Eigenschaft `border-spacing` den Wert `collapse`, hat `empty-cells` keine Auswirkungen auf die Darstellung.

table-layout

Regelt, wie der Webbrowser eine Tabelle darstellt, und kann kleinere Auswirkungen auf die Darstellungsgeschwindigkeit haben. Mit dem Wert `fixed` wird der Browser angewiesen, alle Spalten mit der gleichen Breite darzustellen wie die der ersten Spalte. Das kann (aus sehr komplizierten Gründen) die Darstellung der Tabelle beschleunigen. Der Standardwert `auto` sorgt dagegen dafur, dass der Browser einfach „sein Ding“: macht. Wenn die Darstellungsgeschwindlgkeit Ihrer Tabelle für Sie schnell genug ist, werden Sie diese Eigenschaft nicht brauchen. Die Eigenschaft `table-layout` funktioniert nur mit Regeln, die das `<table>`-Tag formatieren.

- **Werte**: `auto`, `fixed`.

```
table-layout: fixed;
```

Verschiedene Eigenschaften

Neben den bereits vorgestellten gibt es in CSS 2.1 ein paar zusätzliche Eigenschaften, die möglicherweise interessant für Sie sind. Mit ihnen können Sie spezielle Inhalte in Webseiten einfügen oder eigene Cursor definieren. Außerdem können Sie steuern, wie eine Seite ausgedruckt wird, und noch einiges andere. (Leider ist die Browserunterstützung für diese Eigenschaften ziemlich dürftig.)

content

Mit dieser Eigenschaft können Sie Inhalte (Text, Bilder) definieren, die mit den Pseudoelementen `:before` und `:after` vor oder nach einem Element dargestellt werden. Diese Eigenschaft wird erst ab Internet Explorer 8 unterstützt, was das Anwendungsgebiet möglicherweise etwas einschränkt.

- **Werte**: Text in Anführungszeichen „wie hier“ und die Schlüsselwörter `normal`, `open-quote`, `close-quote`, `no-open-quote`, `no-close-quote`. Außerdem ist es möglich, den Wert eines HTML-Attributs zu verwenden. Außerdem ist es möglich, per `url()` eine URL anzugeben, die beispielsweise auf eine Grafikdatei verweist, oder (zumindest theoretisch) mit `counter()` einen Zähler zu realisieren.

```
p.anzeige: before {
content: "Und jetzt mal etwas vollkommen anderes..."; }

a:after {
content: " (" attr(href) ") ";
}
a:before {
content: url("link.gif");
}
```

Info

Auf diese Weise hinzugefügte Inhalte werden als erzeugter Inhalt bezeichnet. Eine einfache Erklärung für erzeugte Inhalte finden Sie beispielsweise unter css4you.de. Eine ausführliche Beschreibung (in Englisch) gibt es hier: w3.org.

cursor

Hiermit können Sie das Aussehen des Mauszeigers verändern, z.B. wenn dieser über ein bestimmtes Element bewegt wird. Vielleicht wollen Sie den Mauszeiger in ein Fragezeichen verwandeln, wenn dieser über einen Link bewegt wird, der auf weitere Informationen zu einem Thema (z.B. die Definition eines Worts) verweist.

- **Werte**: `auto`, `default`, `crosshair`, `pointer`, `move`, `e-resize`, `ne-resize`, `nw-resize`, `n-resize`, `se-resize`, `sw-resize`, `s-resize`, `w-resize`, `text`, `wait`, `help`, `progress`. Daneben können Sie auch eine URL angeben, die auf eine eigene Cursorgrafik verweist. (Mehr dazu im unten stehenden Hinweis.)

```
cursor: help; cursor: url(bilder/cursor.cur);
```

Info
URLs für eigene Cursor werden nur vom Internet Explorer und Firefox unterstützt. Weitere Informationen finden Sie unter echoecho.com und quirksmode.org.

orphans

Gibt die minimale Anzahl von Textzeilen innerhalb eines Elements an, die für sich am unteren Ende einer Seite stehen dürfen. Der Wert dieser Eigenschaft kann die Platzierung von Seitenumbrüchen innerhalb des Elements beeinflussen. Angenommen, Sie drucken Ihre Webseite auf einem Laserdrucker aus, und ein fünfzeiliger Absatz wird ohne weitere Angaben auf zwei Seiten verteilt. Die erste Zeile steht auf der ersten Seite, die übrigen vier auf der nächsten. Da eine einzelne Zeile am Seitenende wie ein Waisenkind (engl. orphan) wirkt, können Sie den Browser anweisen, den Zeilenumbruch nur durchzuführen, wenn z.B. mindestens drei Zeilen am unteren Ende stehen bleiben.

- **Werte**: Ein ganzzahliger positiver Wert wie 1, 2, 3 oder 5.

```
orphans: 3;
```

page-break-after

Legt fest, ob (beim Ausdrucken) nach einem bestimmten Element ein Seitenumbruch erzeugt werden soll. Auf diese Weise können Sie z.B. sicherstellen, dass ein bestimmter Absatz immer als Letztes aufeiner Seite erscheint.

- **Werte**: `auto`, `always`, `avoid`, `left`, `right`. Der Standardwert `auto` überlässt das Setzen von Seitenumbrüchen dem Browser. Die Einstellung `always` erzwingt direkt nach dem Element einen Seitenumbruch. Das folgende Element erscheint auf jeden Fall auf der folgenden Seite. Der Wert `avoid` verhindert, dass nach dem Element ein Seitenumbruch erzeugt wird. Diese Einstellung ist sinnvoll, wenn beispielsweise eine Überschrift und der folgende Absatz immer auf einer Seite stehen sollen. Leider wird dies jedoch nicht von allen Browsern verstanden. Die Werte `left` und `right` legen fest, ob das folgende Element auf einer links oder rechts angeordneten Seite (wie in einem Buch) ausgegeben wird. Es kann also passieren, dass der Browser eine zusätzliche Seite ausgibt. Momentan werden die letzten beiden Werte allerdings von keinem Browser verstanden. Sie brauchen sich also keine Sorgen um verschwendetes Papier zu machen. Die meisten Browser verhalten sich bei `left` und `right`, als wäre der Wert `always` angegeben worden.

```
page-break-after: always;
```

page-break-before

Funktioniert wie `page-break-after`, nur dass der Seitenumbruch vor dem Element erzeugt wird, sodass es als Erstes auf der folgenden Seite ausgegeben wird. Mit dieser Eigenschaft können Sie sicherstellen, dass überschriften für bestimmte Bereiche einer Webseite immer am Anfang der Druckseite stehen.

- **Werte**: Die gleichen wie für `page-break-after`.

```
page-break-before: always;
```

page-break-inside

Verhindert, dass ein Element auf zwei Druckseiten verteilt wird. Wenn Sie ein Foto und seine Beschriftung auf jeden Fall auf derselben Seite ausdrucken wollen, umgeben Sie das Foto und die Beschriftung mit einem -Tag. Auf dieses wenden Sie dann die Eigenschaft `page-break-inside` an.

- **Werte**: `avoid`

```
page-break-inside: avoid;
```

widows

Diese Eigenschaft ist das Gegenteil von `orphans`. Sie gibt die minimale Anzahl von Zeilen an, die am Anfang einer gedruckten Seite stehen müssen. Vielleicht können am Ende der vorigen Seite nur vier von fünf Zeilen eines Absatzes ausgegeben werden. Die fünfte Zeile erscheint allein wie eine Witwe (engl. widow) auf der folgenden Seite. Mit der Eigenschaft `widows` können Sie die Anzahl der Zeilen festlegen, die am Anfang einer neuen Seite stehen sollen. Diese Eigenschaft wird nur von Opera und IE 8 unterstützt.

- **Werte**: ein ganzzahliger Wert wie 1, 2, 3 oder 5.

```
widows: 3;
```

Um CSS optimal nutzen zu können, muss Ihr **HTML-Code** ein solides und gut gebautes Fundament bieten. In diesem Kapitel lernen Sie, wie Sie besseren und CSS-freundlicheren HTML-Code schreiben können. Das Gute daran ist, dass das Schreiben von HTML durch die Verwendung von CSS automatisch leichter wird. Sie brauchen sich keine Gedanken mehr darüber zu machen, wie Sie HTML für Designzwecke einsetzen, obwohl es niemals dafür gedacht war. Stattdessen bietet Ihnen CSS sämtliche benötigten grafischen Designelemente. Ihre Arbeit wird erleichtert, weil für CSS optimierte HTML-Seiten weniger Code und Tipparbeit bedeuten und außerdem deutlich leichter zu erstellen sind. Zudem laden Ihre Seiten schneller – ein willkommener Bonus, für den Ihnen die Besucher Ihrer Site dankbar sein werden.

HTML: Vergangenheit und Zukunft

HTML und dessen Nachfolger **XHTML** bilden das Grundgerüst fast jeder Seite, die Sie im World Wide Web finden. Sobald CSS ins Spiel kommt, ändert sich allerdings auch die Verwendung von HTML. Verabschieden Sie sich endgültig davon. HTML-Tags zu verbiegen, nur um bestimmte grafische Effekte zu erzielen. Einige Tags und Attribute wie das berüchtigte <font>-Tag können Sie ab sofort komplett vergessen. Die folgenden Abschnitte erklären Ihnen, warum.

Info

Alles was Sie in diesem Kapitel über **HTML** lesen, gilt gleichermaßen auch für XHTML. Es gibt mehr Varianten von (X)HTML als Regenbogenfarben. Am Ende müssen Sie sich für eine Variante entscheiden und dafür sorgen, dass Ihre Webseite darüber informiert, welche Version das ist. Ansonsten kann es passieren, dass die Browser Ihrer Besucher Ihre mit viel Aufwand erstellte Seite komplett zerlegen. Später in diesem Kapitel werden Sie lernen, wie Sie CSS mitteilen, welche Geschmacksrichtung von (X)HTML verwendet wird.

HTML in der Vergangenheit: Hauptsache, es sieht gut aus

Als eine Handvoll Wissenschaftler das Web erfand, um technische Dokumentationen leichter gemeinsam nutzen und nachverfolgen zu können, hat niemand die Grafikdesigner nach ihrer Meinung gefragt. Die Wissenschaftler brauchten HTML nur, um Informationen zum leichteren Verständnis klar zu strukturieren. So steht das Tag `<hl>` für eine wichtige überschrift („ersten Grades“), während `<h2>` weniger Gewicht hat und meist als Untertitel für `<hl>`-Überschriften verwendet wird. Ein weiterer Favorit ist das `<ol>`-Tag (ordered list), das eine nummerierte Liste erzeugt, z.B. für „1000 Dinge, die man bei Schwerelosigkeit tun kann“.

> Als auch andere Leute begannen, HTML zu benutzen, wollten diese, dass ihre Seiten gut aussähen.

Also begannen die **Webdesigner**, die Tags zum Steuern der Darstellung zu verwenden und nicht mehr nur zum Strukturieren der Inhalte. So können Sie beispielsweise das `<blockquote>`-Tag (eigentlich gedacht, um längere Zitate zu markieren) für beliebigen Text verwenden, den Sie ein wenig einrücken wollen. Mithilfe der Tags für Überschriften können Sie bestimmte Textteile größer und fett gedruckt darstellen – unabhängig davon, ob es sich tatsächlich um eine überschrift handelt oder nicht.

Noch raffinierter war die Verwendung des `<table>`-Tags, z.B. um mehrspaltigen Text darzustellen oder Text und Bilder möglichst genau auf einer Seite zu platzieren. Eigentlich war das `<table>`-Tag dafür gedacht, tabellarische Daten zu markieren und darzustellen, z.B. Messergebnisse, Zugfahrpläne usw. Weil es nichts Vergleichbares gab, benutzten die Designer das Tag auf unzweckmäßige Weise. Hierbei wurden nicht selten mehrere Tabellen ineinander verschachtelt, damit die Seiten gut aussahen.

In der Zwischenzeit erfanden die Browserhersteller neue Tags und Attribute, nur um das Aussehen einer Seite zu verbessern (manche Webdesigner wachen heute noch schweißgebadet auf, weil sie wieder einmal von `<blink>` und `<marquee>` geträumt haben). Das `<font>`-Tag ermöglicht die Angabe einer bestimmten Schrift, einer Schriftgröße (in sieben möglichen Schritten) und einer Schriftfarbe. (Vermutlich haben Sie schon gemerkt, dass das etwa 100 Schritte weniger sind, als beispielsweise Microsoft Word anbietet.)

> Wenn selbst der übelste Tag-Missbrauch nicht zum gewünschten Ergebnis führte, hatten die Webdesigner

immer noch einen letzten Ausweg: die Verwendung von Grafiken. Um das automatische Wiederholen („Kacheln“) von Hintergrundbildern zu verhindern, wurden beispielsweise übergroße Grafiken als Hintergrund verwendet; andersherum wurden große Grafiken gern in kleinere Einzelbilder zerschnitten (das berühmte „Slicing“) und mithilfe von Tabellen wieder zusammengeschustert.

Zwar geben die oben genannten Techniken – die kreative, aber nicht unbedingt zweckmäßige Verwendung von **HTML-Tags** und die ausufernde Verwendung von Grafiken – Ihnen eine große Kontrolle über die Seitengestaltung. Gleichzeitig bedeutete dies aber auch wesentlich mehr HTML-Code (und mehr neue Stirnfalten, die nicht einmal ein ganzes Leben in der Sonne hervorbringen kann). Lange Zeit waren diese Techniken die einzige Möglichkeit, ein „anständiges“ Seitenlayout zu erreichen. Mit der Einführung von CSS hat sich diese Sichtweise jedoch radikal verändert.

HTML heute: Das Grundgerüst für CSS

Es macht keinen Unterschied, ob eine Seite nun den Veranstaltungskalender Ihres Angelvereins, eine Anfahrtsbeschreibung zum nächsten IKEA oder die Bilder vom letzten Kindergeburtstag enthält. Am Ende entscheidet das **Design** darüber, ob die Seite einen professionellen Eindruck macht oder sich eher wie das Werk eines Hobbybastlers präsentiert. Gutes Design unterstützt den Inhalt Ihrer Seite, hilft Besuchern, das Gesuchte zu finden, und legt fest, wie der Rest der Welt Ihre Website betrachtet. Aus diesem Grund haben Webdesigner die oben genannten Verrenkungen überhaupt auf sich genommen, um eine Seite gut aussehen zu lassen. Indem CSS diese Designausgaben übernimmt, kann HTML wieder das tun, wofür es eigentlich gedacht war: Inhalte mit einer Struktur zu versehen.

Die Verwendung von HTML zur Steuerung des Aussehens von Text und anderen Elementen einer Webseite ist veraltet. Machen Sie sich keine Sorgen, wenn die Darstellung des `<hl>`-Tags für Ihren Geschmack zu groß ist oder die Abstände in einer Aufzählungsliste Ihnen nicht gefallen. Wie Sie später sehen werden, lässt sich das Problem mithilfe von CSS ganz einfach lösen. Stellen Sie sich HTML einfach als eine Möglichkeit vor, den Inhalt, den Sie im Web veröffentlichen wollen, mit einer ordentlichen Struktur zu versehen. Kurz gesagt: Verwenden Sie HTML, um Ihre Inhalte zu organisieren, und CSS, um sie auch noch gut aussehen zu lassen.

HTML-Code für die Verwendung mit CSS schreiben

Wenn **Webdesign** für Sie etwas Neues ist, brauchen Sie vermutlich einige Orientierungshilfen bei Ihren Ausflügen in die Welt von HTML. (Halten Sie sich dabei prinzipiell von veralteten HTML-Techniken fern!) Wenn Sie sich dagegen schon länger mit der Gestaltung von Webseiten befassen, haben Sie unterwegs vermutlich ein paar schlechte Gewohnheiten beim Schreiben von HTML angenommen, die Sie besser möglichst schnell vergessen. Der Rest dieses Kapitels macht Sie mit ein paar Regeln zum Schreiben von

HTML bekannt, die Ihnen dabei helfen werden, möglichst viel aus CSS herauszuholen. Ihre Mutter wird stolz auf Sie sein!

HTML = Struktur

HTML gibt den Textteilen Ihrer Seite eine Bedeutung, indem es sie logisch aufteilt und je nach Funktion in der Seite mit entsprechenden Markierungen versieht. So enthält der Text innerhalb der <hl>-Tags die Hauptüberschrift für Ihre Seite. Mit anderen Überschriften können Sie den Inhalt in andere, weniger wichtige, aber verwandte Abschnitte unterteilen. Wie das Buch, das Sie gerade lesen, sollte auch eine Webseite eine logische Struktur aufweisen. Jedes Kapitel dieses Buchs besitzt einen Titel (sprich: <h1> und mehrere Abschnitte (<h2>), die ihrerseits in kleinere Unterabschnitte aufgeteilt sind. Stellen Sie sich vor, wie schwer diese Webseite zu lesen wäre, wenn es keine Absätze, Abschnitte oder Überschriften gäbe, sondern nur einen riesigen Textblock.

Info

Eine gute Ressource zum Thema (X)HTML ist das Buch HTML & XHTML – Dos umfossende Handbuch (O'Reilly) von Chuck Musciano und Bill Kennedy. Eine gute deutschsprachige Anleitung zum Thema finden Sie unter HTML. Eine Liste aller verfügbaren (X)HTML-Tags und ihrer Attribute finden Sie unter HTML-Referenz oder auf Englisch auch unter w3schools-tags.

Neben den Tags für Überschriften besitzt HTML eine Vielzahl weiterer Tags zum Markieren (das „M" in HTML) bestimmter Textteile entsprechend ihrer Funktion in einer Seite. Mit am beliebtesten ist hier das <p>-Tag, mit dem Sie Textteile als „Absatz" kennzeichnen können, und <ul> zum Erzeugen einer ungeordneten (nicht nummerierten) Liste. Weniger bekannte Tags können sehr genau bezeichnen, um welche Art von Inhalt es sich handelt, z.B. <abbr> für Abkürzungen und <code> für die Kennzeichnung von Programmiercode.

> Wenn Sie Ihren HTML-Code für die Verwendung von CSS optimieren, sollten Sie das HTML-Tag wählen,

das möglichst genau die jeweiligen Rolle beschreibt, die ein bestimmter Textteil in der Seite spielt (Überschrift, Absatz, Adresse, Link usw.), anstatt zu bestimmen, wie dieser Teil dargestellt werden soll. So ist eine Sammlung von Links in einer Navigationsleiste eigentlich keine Überschrift und natürlich auch kein normaler Textabsatz. Am ehesten handelt es sich um eine ungeordnete Liste mit möglichen Optionen, für die das -Tag eine gute Wahl ist. An dieser Stelle könnten Sie einwenden: „Mag ja sein, aber ich möchte die Links nicht untereinander, sondern nebeneinander in einem horizontalen Navigationsmenü anordnen". Keine Sorge. Mit ein bisschen CSS-Magie (und wir reden hier noch lange nicht vom richtigen Voodoo) können Sie die normalerweise vertikale Liste in eine elegante (horizontale oder vertikale) Navigationsleiste verwandeln.

Zwei neue HTML-Tags

Trotz der Vielfalt an HTML-Tags wird es Situationen geben, in denen genau das Tag, das Ihren Inhalt exakt beschreibt, vermutlich fehlt. Sicher ist `<code>` gut geeignet, um den **Quellcode** eines Computerprogramms zu kennzeichnen, aber die meisten Anwender würden `<kochrezept>` praktischer finden. Leider gibt es so eins nicht. Glücklicherweise besitzt HTML zwei Tags, mit denen Sie Ihren Inhalt besser beschreiben und im gleichen Schritt auch noch mit passenden CSS-Anweisungen verknüpfen können.

Die Tags `<div>` und `<span>` sind wie leere Gefäße, die Sie mit beliebigem Inhalt füllen können. Diese Elemente besitzen keine eigenen visuellen Eigenschaften. Sie können ihnen mit CSS ein beliebiges Aussehen geben. Das `<div>`-Tag (für engl. division, Unterteilung) kennzeichnet einen eigenständigen Teil des Seiteninhalts, ähnlich einem Absatz oder einer Überschrift. Sie können das `<div>`-Tag aber auch verwenden, um beliebige andere Teile des Inhalts damit zu umgeben, z.B. eine Überschrift, mehrere Absätze oder eine Liste. Das `<div>`-Tag eignet sich daher hervorragend, um eine Seite in mehrere logische Bereiche zu unterteilen, beispielsweise für das Logo, den Navigationsbereich, den Haupttext, die Fußzeile, usw. Mit Hilfe von CSS können Sie die so markierten Bereiche später positionieren und auf diese Weise anspruchsvolle Seitenlayouts zu erstellen.

Das `<span>`-Tag wird für sogenannte Inline-Elemente verwendet. Das können einzelne Wörter oder Sätze sein, die Teil eines längeren Absatzes oder einer Überschrift sind. Sie können `<span>` verwenden wie andere Inline-Tags auch. z.B. `<a>` (für das Einfügen von Links) oder `<strong>` (zum Hervorheben eines Worts in einem Absatz). Das `<span>`-Tag kann beispielsweise benutzt werden, um einen Firmennamen zu markieren. Anschließend können Sie CSS verwenden, um den Namen mit einer eigenen Schrift, Farbe oder Ähnliches hervorzuheben. Hier ein Beispiel für diese Tags in Aktion, inklusive eines kleinen Ausblicks auf die Attribute `id` und `class`, die häufig verwendet werden, um bestimmte Teile einer Seite mit Stildefinitionen zu versehen:

```
<div id="fusszeile">
<p>Copyright 2009. <span class="firmenname">CosmoFarMer.com</span></p>
<p>Wir sind telefonisch unter der Nummer 555-555-5501 für Sie zu erreichen.</p>
</div>
```

Diese Tags werden Ihnen auch außerhalb dieser kurzen Einführung noch häufig begegnen. Besonders in CSS-lastigen Websites werden sie oft und gerne benutzt. Im weiteren Verlauf dieses Kapitels werden Sie lernen, wie Sie diese Tags in Kombination mit CSS verwenden können, um die kreative Kontrolle über Ihre Website voll auszunutzen.

HTML-Code, den Sie am besten vergessen

CSS vereinfacht das Schreiben von HTML aus einem wichtigen Grund: Sie brauchen keine Berge von Tags und Attributen mehr, um Ihre Seiten besser aussehen zu lassen. Das <font>-Tag ist hierfür das strahlendste Beispiel. Sein einziger Zweck besteht darin, Farbe, Größe und Schrift eines Texts festzulegen. Zum Verständnis der Seitenstruktur trägt er jedoch überhaupt nicht bei.

Hier eine Liste der Tags und Attribute, die Sie einfach durch CSS ersetzen können:

- Die Verwendung des <font>-Tag für die Darstellung von Text gehört auf den Müll. CSS kann wesentlich besser (und vielfältiger) mit Text umgehen.
- Hören Sie damit auf, <b> und <i> zu benutzen, um Text fett oder kursiv darzustellen. Mit CSS können Sie jedes Tag fett oder kursiv darstellen. Diese formatspezifischen Tags sind also nicht mehr nötig. Wenn Sie dagegen tatsächlich ein bestimmtes Wort oder einen Satz hervorheben wollen, verwenden Sie dafür das -Tag (das von den meisten Browsern sowieso fett dargestellt wird). Sollen Textteile nur leicht hervorgehoben werden, verwenden Sie (für eng!. emphasis – Betonung). Die meisten Browser stellen Text innerhalb dieser Tags kursiv dar.

Info

Soll der Titel einer Publikation kursiv dargestellt werden, schlägt das <cite>-Tag zwei Fliegen mit einer Klappe. Der Titel wird kursiv dargestellt, und der Text wird als Zitat gekennzeichnet, was wiederum die Suchmaschinen freut. Dieses Tag sollten Sie sich merken.

- Verwenden Sie für Seitenlayouts auch nicht mehr das <table>-Tag! Benutzen Sie es nur, um tatsächlich tabellarische Informationen darzustellen, z.B. Daten aus einer Tabellenkalkulation, die Bundesligatabelle, eine Aufstellung der günstigsten Telefonanbieter mit Zeiten und Preisen usw. Sie können Ihr gesamtes Seitenlayout mit CSS erstellen, und das Ganze noch in wesentlich kürzerer Zeit als mit „<table>-dance“.
- Eliminieren Sie die unbeholfenen Attribute für das <body>-Tag, die sich nur auf die Präsentation Ihres Inhalts auswirken; `background`, `bgcolor`, `text`, `link`, `alink` und `vlink` legen Farben und Hintergrundbilder für Seite, Text und Links fest. Mit CSS lässt sich die Aufgabe wesentlich besser bewältigen. Und wenn Sie schon dabei sind, trennen Sie sich auch gleich von den browserspezifischen Attributen für die Seitenränder: `leftmargin`, `topmargin`, `marginwidth` und `marginheight`. Auch hier ist CSS deutlich im Vorteil.
- Missbrauchen Sie keine
-Tags. Wenn Sie damit aufgewachsen sind, das
-Tag zu verwenden (
 in HTML), um innerhalb von Absätzen Zeilenumbrüche herbeizuführen, haben wir etwas Besonderes für Sie. (Browser fügen automatisch und oftmals zum Ärger des Webdesigners einen gewissen Leerraum zwischen Absätzen, aber auch zwischen Absätzen und <p>-Tags ein. In der Vergangenheit haben Webdesigner ausgeklügelte Methoden entwickelt, um das Erzeugen dieser automatischen Zwischenräume zwischen den <p>-Tags zu umgehen. Sie benutzten mehrere
-Tags hintereinander und drehten dann per <font> die erste Zeile so sehr durch die Mangel, dass sie aussah wie eine überschrift.) CSS hält eine Reihe von Möglichkeiten bereit, Innen- wie Außenabstände genau einzustellen – und zwar nicht nur für Absätze, sondern auch für andere Block-Elemente wie beispielsweise Überschriften.

Allgemein gilt: Sämtliche Tag-Attribute für die Definition von Farben, Rahmen, Hintergrundbildern oder die Ausrichtung (das gilt auch für Tabellen) gelten komplett als veraltet und sollten nicht mehr benutzt werden. Das Gleiche gilt für die Ausrichtung von Bildern (z.B. `align=left`) und das Zentrieren von Text innerhalb eines Absatzes (z.B. `align=center` oder noch schlimmer: `<center>` und von Tabellenzellen. Für alle diese Aufgaben bietet CSS einen mehr als angemessenen Ersatz.

Ein paar Tipps für unterwegs

Es ist immer gut, eine Karte zu haben, wenn man sich nicht auskennt. Wenn Sie sich immer noch nicht sicher sind, wie Sie HTML zum Erstellen gut strukturierter Webseiten verwenden können, haben wir hier ein paar Tipps, die Sie auf den Weg bringen sollen:

- Verwenden Sie immer nur ein `<hl>`-Tag pro Seite und nehmen Sie es nur dafür, um damit das Hauptthema der Seite zu kennzeichnen. Stellen Sie es sich als eine Art Kapitelüberschrift vor. Die korrekte Verwendung von `<hl>` trägt dazu bei, dass die Seite korrekt von Suchmaschinen indiziert werden kann.
- Verwenden Sie Überschriften, um wichtige Textstellen zu markieren. Stellen Sie sich vor, Sie wollten Ihr Kapitel in Abschnitte unterteilen. Wenn zwei Überschriften das gleiche Gewicht haben, werden sie mit Tags der gleichen Ebene gekennzeichnet <h2>, <h3> usw.). Ist eine Überschrift weniger wichtig oder steht sie für ein Unterthema, verwenden Sie eine Überschrift der nächsten Hierarchieebene: Auf <h2> folgt <h3> usw. Andersherum sollte <h5> nicht direkt auf ein <h2>-Tag folgen.
- Verwenden Sie das <p>-Tag nur für Absätze. (Schließlich steht das p für „paragraph“, also das englische Wort für Absatz.)
- Verwenden Sie ungeordnete Listen (`<ul>`), wenn Sie eine Liste mit verwandten Dingen erstellen wollen, z.B. Navigationslinks, Überschriften oder eine Aufzählung mit Tipps wie diese.
- Nehmen Sie stattdessen nummerierte Listen (`<ol>`), um die Listenelemente in einerbestimmten Reihenfolge darzustellen, wie z.B. „Die 10 schlimmsten Schlager der Weltgeschichte“ oder „25 Dinge, die Sie nicht bei Schwerelosigkeit tun sollten“.
- Um ein Glossar mit Begriffen und deren Definitionen zu erstellen, ist das `<dl>`-Tag (definition list) zusammen mit den Tags `<dt>` (definition term – zu definierender Begriff) und `<dd>` (Definition) am besten geeignet.
- Für ein besonderes Stück Text oder ein Zitat (z.B. eine Filmkritik oder einen weisen Ausspruch Ihres Großvaters) eignet sich das `<blockquote>`-Tag am besten. Kurze (einzeilige) Abschnitte markieren Sie dagegen mit dem <q>-Tag.
- Nutzen Sie auch weniger bekannte Tags wie `<cite>`, mit denen Sie Buchtitel, Zeitschriftenartikel oder Referenzen auf andere Websites markieren können, und `<address>`, um Kontaktinformationen zum Autor einer Seite zu kennzeichnen (besonders in Copyright-Hinweisen).
- Wie schon bereits besprochen, sollten Sie sich von allen Tags und Attributen fernhalten, die nur dafür da sind, das Erscheinungsbild von Text oder Bildern zu verändern. Wie Sie sehen werden, kann CSS das auch – nur viel besser.

- Wenn es einmal kein HTML-Tag gibt, das Ihren Inhalt genau beschreibt, Sie aber ein oder mehrere Seitenelemente markieren wollen, können Sie dies mit den Tags `<div>` und `<span>` tun.
- Denken Sie immer daran, dass Tags bis auf wenige Ausnahmen paarweise auftreten. Ein öffnendes `<p>` benötigt immer auch ein schließendes `</p>`. Andere Tags „schließen sich selbst“, wie z.B. `<br />` und `<img />`.
- Gewöhnen Sie sich an, Ihre Seiten prinzipiell mit dem W3C-Validator zu überprüfen. Schlecht geschriebener oder mit Tippfehlern versetzter HTML-Code kann zu unvorhersehbarem Browserverhalten führen.

Die Bedeutung der Dokumenttyp-Definition

Wie schon besprochen, folgt HTML bestimmten Regeln. Diese Regeln stehen in einer **Dokumenttyp-Definition** kurz: DTD. Bei der DTD handelt es sich um eine einfache, im Internet verfügbare Datei im XML-Format, die beschreibt, welche Tags, Attribute und Werte für bestimmte Arten von (X)HTML gültig sind. Für jede HTML-Version gibt es eine eigene DTD. Möglicherweise fragen Sie sich jetzt, was das alles mit CSS zu tun hat.

Die Antwort lautet: „Alles!“, jedenfalls wenn Sie wollen, dass Ihre Webseiten korrekt und unabhängig vom verwendeten Browser konsistent dargestellt werden. Damit der Browser „weiß“, welche HTML- oder XHTML-Version Sie verwenden, wird am Anfang einer Webseite eine sogenannte Dokumenttyp-Deklaration eingefügt. Sie ist die erste Zeile einer HTML-Datei. Neben Informationen über die verwendete HTML-Version (z.B. HTML 4.01 Transitional) kann sie auch einen Verweis auf die die entsprechende DTD-Datei im Web enthalten. Ein Tippfehler in der Dokumenttyp-Deklaration versetzt einige Browser in einen speziellen Zustand, der als Quirles-Modus („Macken-Modus“) bezeichnet wird.

> Dieser Quirks-Modus ist der Versuch der Browserhersteller, Ihre Programme dazu zu bringen, Webseiten so darzustellen,

wie es die Browser ungefahr 1999 (zu Zeiten von Netscape 4 und Internet Explorer 5) getan haben. Stößt ein moderner Browser auf eine Seite mit fehlendem oder fehlerhaften Dokumenttyp-Angaben, denkt er sich: „Hmmm. Diese Seite scheint ziemlich alt zu sein. Ich will doch mal versuchen, ob ich sie nicht so darstellen kann, wie es die verquasten Browser von damals getan haben.“ Aus diesem Grund werden Ihre liebevoll mit CSS gestalteten Seiten auch ohne korrekten Dokumenttyp nicht so aussehen, wie sie es gemäß den aktuellen Standards eigentlich sollten. Möglicherweise betrachten Sie eine Webseite bei der Überprüfung in einem Browser versehentlich im Quirks-Modus. Dann kann es passieren, dass Sie einen fehlerhaften Dokumenttyp womöglich für Fehler in Ihrem **HTML**– oder **CSS-Code** halten und versuchen, Probleme zu lösen, die gar keine sind.

Info

Weitere (technische) Informationen finden Sie unter anderem auf der Seite von quirksmode.

Glücklicherweise gibt es eine einfache Methode, den richtigen Dokumenttyp zu verwenden. Hierfür müssen Sie nur wissen, welche (X)HTML-Version Sie benutzen. Mit größter Wahrscheinlichkeit haben Sie bereits Webseiten mit HTML 4 erstellt. Möglicherweise nehmen Sie sogar schon XHTML für Ihre Site.

Die beliebtesten HTML- und XHTML-Versionen sind heutzutage **HTML 4.01 Transitional** und **XHTML 1.0 Transitional**. Mit diesen Versionen sind Tags für die Präsentation wie `<font>` weiterhin zugelassen. Sie erlauben einen übergang (engl. transition) von älteren HTML-Versionen zu den neueren und strengeren HTML- und XHTML-Varianten. Auch wenn es – wie Sie inzwischen wissen sollten – besser ist, diese Tags überhaupt nicht benutzen, sind sie in den Transitional-Versionen noch erlaubt. Sie können die Umstellungen also in Ihrem eigenen Tempo vornehmen. In den strikten (X)HTML-Versionen funktionieren einige der älteren Tags überhaupt nicht mehr.

Info

Allgemein verbieten die strikten Versionen von HTML und XHTML Tags und Attribute, die nur das Aussehen von Seiten verändern, wie das notorische `<font>`-Tag oder das `align`-Attribut für Absätze (z.B. `<p align="center">`. Außerdem sind verschiedene, früher einmal beliebte Eigenschaften wie das `target`-Attribut für Links (zum Öffnen von Links in einem neuen Browsertenster) nicht mehr erlaubt.

Wenn Sie HTML 4.01 Transitional verwenden, muss die folgende Dokumenttyp-Deklaration am Anfang jeder von Ihnen erstellten Seite stehen:

```
<!OOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/htm14/loose.dtd">
```

Die Dokumenttyp-Deklaration für XHTML 1.0 Transitional ist ähnlich, verweist aber auf eine andere DTD. Es ist außerdem empfehlenswert, das öffnende `<html>`-Tag mit ein paar zusätzlichen Informationen zu den im Dokument verwendeten Sprachen zu versehen:

```
<!OOCTYPE html PUBlIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="de">
```

Info

Wenn Sie in Ihren Webseiten Frames einsetzen, müssen Sie den entsprechenden Dokumenttyp für Frames verwenden. Eine vollständige liste der möglichen Dokumenttypen finden Sie unter w3.org.

Auch wenn diese Erläuterungen bei Ihnen eher für Kopfschmerzen und langsam zufallende Augen sorgen, sollten Sie auf jeden Fall darauf achten, den richtigen Dokumenttyp anzugeben und diese Angabe immer in die erste Zeile in Ihre (X)HTML-Dokumente (noch vor dem öffuenden `<html>`-Tag) zu schreiben. Es ist empfehlenswert, eine leere HTML-Seite mit den korrekten Dokumenttyp-Deklarationen auf Ihrem Computer bereitzuhalten. Auf diese Weise können Sie schnell eine Kopie machen, wenn Sie eine neue Seite erstellen wollen.

Info

Die meisten visuellen Programme zum Erstellen von Webseiten, wie Dreamweaver, Golive und der FrontPage-Nachfolger Microsoft Expression, togen automatisch die korrekte Dokumenttyp-Deklaration in neu erstellte Webseiten ein. Viele HTML-fähige Texteditoren besitzen ebenfalls Abkürzungen zum Hinzufügen von Dokumenttyp-Deklarationen.

Je komplexer Ihre Stildefinitionen werden, desto mehr fragen Sie sich vermutlich, wie ein Seitenelement trotz scheinbar widersprüchlicher Anweisungen das korrekte Aussehen erhält. Wie im vorigen Kapitel besprochen, sorgt die Möglichkeit der Vererbung in CSS dafür, dass ein Tag bestimmte Eigenschaftswerte von den umgebenden Tags erhen kann. So kann das <body>-Tag Formatierungen an einen enthaltenen Absatz weitergeben, und der Absatz gibt diese Anweisungen an einen enthaltenen Link weiter. Das heißt aber auch, dass der Link CSS-Eigenschaftswerte sowohl vom <body> – als auch vom <p>-Tag erben kann. Ohne weitere Anweisungen würde diese Möglichkeit zu einer Art „Die Fliege“-Effekt führen, der auf unvorhersehbare Weise Teile aus beiden CSS-Regeln miteinander kombiniert.

Manchmal kommt es aber auch vor, dass in mehreren Stilregeln für das gleiche HTML-Element unterschiedliche Werte für die gleiche Eigenschaft definiert werden (beispielsweise wenn in einem externen und einem internen Stylesheet unterschiedliche Schriftfarben für das <p>-Tag festgelegt werden). In solchen Fällen können recht seltsame Dinge passieren, etwa dass Text hellblau dargestellt wird, obwohl Sie in einer CSS-Klasse ausdrücklich eine rote Textfarbe zugewiesen haben. Tatsächlich sehen Sie hier ein grundsätzliches CSS-Prinzip bei der Arbeit: die Kaskade, von der die Cascading Style Sheets übrigens ihren Namen haben. Sie bestimmt, wie die verschiedenen Stildefinitionen zusammenwirken und welche Stile bei einem Konflikt vor anderen Vorrang (Präzedenz) bekommen.

Info

In diesem Kapitel geht es um Probleme bei der Erstellung komplexer Stylesheets, die stark von der Vererbung und zusammengesetzten Selektoren, wie z.B. mehreren verschachtelten Nachfahren-Selektoren, Gebrauch machen. Die Regeln sind zwar logisch, aber ungefähr so interessant und spannend wie die Gelben Seiten.

Wie Stile kaskadiert werden

Die Kaskade ist eine Reihe von Regeln, die festlegen, welche CSS-Eigenschaften (und deren Werte) auf ein bestimmtes HTML-Element angewandt werden. Sie legt fest, wie Webbrowser mit unterschiedlichen Eigenschaftswerten für das gleiche Tag umgehen sollen und was zu tun ist, wenn CSS-Eigenschaften sich gegenseitig in die Quere kommen. Für Stilkonflikte gibt es zwei mögliche Ursachen: wenn durch Vererbung die gleiche Eigenschaft von verschiedenen Vorfahren geerbt wird und wenn mehrere Stilregeln auf das gleiche Element zutreffen (z.B. wenn es in einem externen und in einem internen Stylesheet eine Regel für <p>-Tags gibt.)

Vererbte Stile werden zusammengefasst

Wie Sie aus dem letzten Kapitel wissen, sorgt die Vererbung von CSS dafür, dass verwandte Elemente, etwa alle Wörter eines Absatzes inklusive Links oder anderer enthaltener Tags, gleich formatiert werden können. Auf diese Weise bleibt es Ihnen erspart, für jedes Tag einer Seite separate Stilregeln defmieren müssen. Da ein Tag seine Eigenschaften aber von allen Vorfahren erben kann, ist es gelegentlich nicht ganz einfach zu ermitteln, warum ein Tag eine bestimmte Formatierung erhält. Stellen Sie sich beispielsweise vor, für das <body>-Tag wird eine bestimmte Schriftart festgelegt, für <p>-Tags eine Schriftgröße und für <a>-Tags eine Schriftfarbe. Jedes <a>-Tag würde also die Schriftart vom <body>-Tag erben, die Schriftgröße jedoch vom <p>-Tag. Die endgültige Darstellung des <a>-Tags ist eine Art Hybridstil – eine Kombination aus den von den Vorfahren-Elementen geerbten Stile.

Die in Abbildung gezeigte Seite enthält drei Stildefinitionen: eine für <body>, eine für das <p>-Tag und eine für . Der CSS-Code sieht folgendermaßen aus:

```
body {
font-family: Verdana, Arial, Helvetica, sans-serif;
}
p {
color: #999999;
}
strong {
font-size: 24px;
}
```

Das -Tag befindet sich innerhalb eines Absatzes, der wiederum vom <body> umgeben wird. Das -Tag erbt von beiden Vorfahren. Von <body> erhält es die Einstellungen für die Eigenschaft `font-family` (die Schriftart) und von <p> die mit der Eigenschaft `color` festgelegte Schriftfarbe. Zusätzlich erhält das -Tag über die CSS-Eigenschaft `font-size` eine Schriftgröße von 24 Pixeln zugewiesen. Das endgültige Aussehen des Tags ist eine Kombination aller drei Stildefmitionen. Anders gesagt: Das -Tag erscheint so, als hätten Sie eine CSS-Regel wie diese definiert:

```
strong {
font-family: Verdana, Arial, Helvetica, sans-serif;
color: #999999;
font-size: 24px;
}
```

Der nächste Vorfahr gewinnt

Im vorigen Beispiel wurden mehrere vererbte und direkt zugewiesene Eigenschaftswerte zu einer Formatierung zusammengefasst. Was passiert aber, wenn die vererbten Eigenschaftswerte einander widersprechen? Stellen Sie sich eine Seite vor, in der <body> und <p> unterschiedliche Schriftfarben haben. Außerdem befände sich innerhalb des Absatzes ein -Tag. Welche Schriftfarbe erhält es wohl? Die Farbe für <body> oder die Farbe für den Absatz? Der Gewinner ist hier der Absatz. Gemäß der CSS-Spezifikation verwendet der Webbrowser die Stildefinition, die dem jeweiligen Tag gemäß den Verwandtschaftsverhältnissen im HTML-Baum am nächsten liegt.

In diesem Beispiel sind die vom <body>-Tag geerbten Stile eher allgemeingültig. Sie beziehen sich auf alle Tags. Ein für das <p>-Tag definierter Stil ist dagegen wesentlich spezieller. Die hier verwendeten CSS-Eigenschaften beziehen sich nur auf Absätze und darin enthaltene Tags.

> Kurz gesagt: Wenn einem Tag kein eigener Stil zugewiesen wurde, gewinnt bei Konflikten mit geerbten Eigenschaftswerten der nächste Vorfahr.

Um das Konzept zu vertiefen, hier noch ein weiteres Beispiel. Nehmen wir mal an, in einem CSS-Stil wurde eine Textfarbe für das <table>-Tag definiert, und in einer anderen Stildefmition wurde dem <td>-Tag innerhalb der Tabelle eine andere Farbe zugewiesen. In diesem Fall verwenden HTML-Elemente (Absätze, Überschriften, Listen), die sich innerhalb des <td>-Tags befinden, die Farbe von <td>, weil dies der nächste Vorfahre für diese Elemente ist.

Direkt zugewiesene Stildefinitionen gewinnen

Zieht man aus der „Nächster Vorfahr“-Regel den logischen Schluss, gibt es im CSS-Stammbaum einen ständigen Hauptgewinner: einen Stil, der einem bestimmten Element direkt zugewiesen wurde. Stellen wir uns vor, es wurde für <body>-, <p>– und -Tags jeweils ein Wert für die Eigenschaft `font-color` definiert. Zwar ist der Stil für den Absatz spezifischer als der für <body>, allerdings ist der direkt für das -Tag definierte Stil noch eindeutiger als die anderen beiden und erhält daher den Vorrang. Er wurde ausdrücklich und ausschließlich für -Tags formuliert, wodurch die vererbten und miteinander konkurrierenden Eigenschaftswerte der Vorfahren-Elemente keine Geltung mehr haben. Einfach gesagt:

> CSS-Eigenschaften, die einem HTML-Element ausdrücklich zugewiesen werden, haben vor vererbten Eigenschaften Vorrang.

Diese Regel erklärt auch, warum manche Eigenschaftswerte scheinbar nicht vererbt werden, obwohl sie das eigentlich sollten. Ein Link innerhalb eines Absatzes mit rotem Text wird trotzdem im Standardblau des Browsers dargestellt, weil der Browser eigene, vordefinierte Stilregeln für das <a>-Tag verwendet und so

möglicherweise vererbte Werte überschreibt.

Ein Tag, viele Stile

Vererbung ist nur eine Möglichkeit, wie ein Tag von mehreren Stildefinitionen beeinflusst werden kann. Daneben ist es auch möglich, dass einem Tag mehrere konkurrierende Eigenschaftswerte direkt zugewiesen werden. Dies kann zum Beispiel passieren, wenn für <p>-Tags sowohl ein internes wie auch ein externes Stylesheet unterschiedliche Werte für eine Eigenschaft festlegen. Um die Sache noch interessanter zu machen, wurde einem der <p>-Tags zusätzlich noch eine CSS-Klasse mit weiteren Werten zugewiesen. Dieses Tag besitzt jetzt also drei konkurrierende Stile, die direkt zugewiesen wurden. Für welchen Stil bzw. welche Stile wird der Browser sich entscheiden?

Die Antwort lautet: Das kommt daraufan. Je nachdem, um welche Art von Stildefinition es sich handelt und aus welcher Quelle sie stammt, kann der Browser eine oder mehrere Regeln anwenden. Hier ein paar Situationen, in denen mehrere Stile fur das gleiche Tag zutreffen können:

- Das Tag wird von einem Typ-Selektor und von einem Klassenselektor ausgewählt. Nehmen wir beispielsweise einen Typ-Selektor für das <h2>-Tag. eine Klasse namens .haupt_ueberschrift und den folgenden HTML-Code: `<h2 class="haupt_ueberschrift">Entdecken Sie Ihre Zukunft!</h2>`. Hier würden ohne weitere Stile beide Definitionen aufdas <h2>-Tag angewendet.

Info

Wenn Sie sich fragen, was bei konkurrierenden Stilen passiert. haben Sie bitte noch etwas Geduld. Die lösung ist bereits in Sicht.

- Ein Klassen- oder ID-Name wird im selben Stylesheet mehrmals verwendet. Vielleicht wird eine CSS-Klasse .haupt_ueberschrift neben Ihrer Definition noch mal in einer Gruppe von Selektoren verwendet, wie hier: .haupt_ueberschrift, .sekundaere_ueberschrift, .artikel_ueberschrift. Beide Regeln enthalten Anweisungen für die Darstellung von Elementen, denen die Klasse .haupt_ueberschrift zugewiesen wurde.
- Einem Tag wurde sowohl ein klassenbasierter als auch ein ID-basierter Stil zugewiesen. Nehmen wir beispielsweise eine ID namens #kopfteil, eine Klasse namens .news und folgenden HTML-Code: `<div id="kopfteil" class="news">`. In diesem Fall werden sowohl die für kopfteil als auch die für news definierten Stile verwendet.
- Es gibt mehr als ein Stylesheet, das Stile mit dem gleichen Namen definiert. Stildefinitionen mit dem gleichen Klassen- oder ID-Namen können gleichzeitig in externen wie internen Stylesheets definiert werden.
- Es gibt komplexe Selektoren. die auf das gleiche Tag zutreffen. Diese Situation kann häufig auftreten, wenn Sie Selektoren für Nachfahren-Elemente verwenden. Das kann beispielsweise der Fall sein, wenn Sie ein `<div>`-Tag verwenden und sich darin ein Absatz mit einem class-Attribut befindet, z.B. `<p class="autor">`. Die folgenden Selektoren können Stile für diesen Absatz

definieren:

- #hauptteil p
- #hauptteil p.autor
- p.autor
- .autor

Werden mehrere Stile für das gleiche Element definiert, kombiniert der Webbrowser die einzelnen Anweisungen, sofern es keinen Konflikt gibt. Ein Beispiel soll dieses Konzept verdeutlichen. Stellen Sie sich vor, Ihre Webseite enthält einen Absatz mit dem Namen des Autors inklusive Link und E-Mail-Adresse. Der HTML-Code könnte in etwa so aussehen:

```
<p class="autor">Geschrieben von <a href="mailto:jean@cosmofarmer.com">Jean Graine de Pomme</a></p>
```

Das Stylesheet enthält drei Regeln. die den Link formatieren:

```
a {
color: #6378df;
}
p a {
font-weight: bold
}
.autor a {
text-decoration: none;
}
```

Die erste Regel färbt alle <a>-Tags hellblau (in der Webfarbe „`powder blue`„) ein, die zweite Regel legt fest, dass alle <a>-Tags innerhalb von <p>-Tags fett dargestellt werden sollen, und die dritte Regel entfernt die standardmäßigen Unterstreichungen für Links, die sich innerhalb von Tags mit der Klasse autor befinden.

Alle drei Regeln gelten für das <a>-Tag. Da in allen drei Regeln unterschiedliche CSS-Eigenschaften benutzt werden, gibt es in diesem Fall keine Konflikte.

Info

Die richtigen Kopfschmerzen fangen erst an, wenn Sie feststellen, dass die Formatierung für den Link auch noch von vererbten Eigenschaftswerten abhängen kann. So würde der Link etwa eine dem umgebenden Absatz zugewiesene Schriftart übernehmen.

Spezifität: Welcher Stil gewinnt?

Das vorige Beispiel ist leicht zu verstehen. Was passiert aber, wenn die drei Regeln oben für die Eigenschaft `font-family` unterschiedliche Schriftarten definieren? Welche der drei Schriften würde schließlich im Webbrowser angezeigt?

Wenn Sie dieses Kapitel sorgfaltig gelesen haben, wissen Sie, dass die Kaskade eine Reihe von Vorschriften zur Verfügung stellt, die dem Webbrowser helfen, Konflikte zwischen CSS-Eigenschaften aufzulösen. In erster Linie heißt das: Regeln mit der höchsten Eindeutigkeit (Spezifität) haben Vorrang vor allgemeinen Regeln. Aber manchmal ist nicht klar, welcher Stil am eindeutigsten ist. Glücklicherweise stellt CSS eine Formel bereit, nach der sich die Spezifität einer CSS-Regel bestimmen lässt. Hierbei werden den verschiedenen Selektor-Arten (wie Typ-Selektor, Klassen-Selektor, ID-Selektor usw.) unterschiedlich hohe Werte zugeordnet. Das System funktioniert so:

- Typ-Selektor: 1 Punkt
- Klassen-Selektor: 10 Punkte
- ID-Selektor: 100 Punkte
- Inline-Stildefinition: 1000 Punkte

Je höher die Zahl, desto höher die Spezifität. Hier ein paar Beispiele:

- ein Typ-Selektor für das <img>-Tag (Spezifität = 1)
- ein Klassen-Selektor mit dem Namen .hervorhebung (Spezifität = 10)
- ein ID-Selektor namens #logo (Spezifität = 100)

Enthält Ihre Webseite den HTML-Code `<img id="logo" class="hervorhebung" src="logo.gif" />` und Sie definieren z.B. für die Eigenschaft `border` in allen drei CSS-Regeln unterschiedliche Werte, gewinnt immer der Wert des ID-Stils (#logo).

Info

Pseudoelemente (wie z.B.: `first-child`) werden wie Typ-Selektoren behandelt und sind demnach einen Punkt wert. Pseudoklassen wie: `link` werden wie normale Klassen behandelt und erhalten zehn Punkte.

Da Nachfahren-Selektoren aus mehreren einzelnen Selektoren bestehen können (z.B. `#inhalt p` oder `h2 strong`), wird die Berechnung etwas komplizierter. Um die Spezifität eines Nachfahren-Selektors zu berechnen, werden die Einzelwerte der enthaltenen Selektoren addiert.

Info

Vererbte Eigenschaften besitzen keine eigene Spezifität. Selbst wenn ein Tag Eigenschaftswerte von einer

Regel mit einer höhen Spezifität erbt, z.B. #kopfteil, haben direkt für das Tag definierte Stile auf jeden Fall Vorrang.

Das Zünglein an der Waage: Der zuletzt definierte Stil gewinnt

Um dem Ganzen die Krone aufzusetzen, kann es trotz aller Regeln passieren, dass zwei Eigenschaften die gleiche Spezifität besitzen. Diese spezielle Form der Zwickmühle kann auftreten, wenn der gleiche Selelktor an zwei verschiedenen Stellen im selben Stylesheet defmiert wird oder zwei verschiedene Stile einfach die gleiche Spezifität besitzen. In solchen Fällen gewinnt einfach der zuletzt definierte Stil.

Hier ein kniffliges Beispiel, das den folgenden HTML-Code verwendet:

```
<p class="autor">Geschrieben von <a class="email"
href="mailto:jean@cosmofarmer.com">Jean Graine de Pomme</a></p>
```

Das Stylesheet für die Seite mit obigem Absatz und Link enthält die folgenden CSS-Regeln:

```
p a.email {
color: blue;
}
p.autor a {
color: red;
}
```

Beide Stile besitzen eine Spezifität von 12 (jeweils 10 für die Klasse und 2 für die Typ-Selektoren), und beide treffen auf das <a>-Tag zu. Gleichstand. Welche Farbe verwendet der Browser? Die Antwort lautet „Rot“ (`red`), weil dies die zweite (und letzte) Definition im Stylesheet ist.

Jetzt stellen Sie sichvor, das Stylesheet sähe stattdessen so aus:

```
p.autor a {
color: red;
}
p a.email {
color: blue;
}
```

In diesem Fall würde der Link blau (`blue`) dargestellt werden. Da `p a.email` hinter `p.autor a` steht, gewinnt auch hier die zweite Definition.

Was passiert mit konkurrierenden Regeln in einem externen und einem internen Stylesheet? In diesem Fall

ist die Platzierung innerhalb Ihrer Stylesheets (in der HTML-Datei) entscheidend. Wenn Sie zuerst ein internes Stylesheet mittels `<style>`-Tags definieren und danach ein externes Stylesheet per `<link>` einbinden, gewinnt die Regel aus dem externen Stylesheet. (Das ist das gleiche Prinzip, das wir gerade besprochen haben: Die zuletzt im Stylesheet definierte Regel gewinnt.) Es ist daher empfehlenswert, externe Stylesheets immer zuerst einzubinden und erst danach die internen Stile zu definieren.

Info

Externe Stylesheets die mit einer `@import`-Regel eingebunden werden, müssen vor den internen Stildefinitionen zwischen den `<style>`-Tags stehen.

Die Kaskade unter Kontrolle behalten

Wie Sie sehen, ist die Gefahr, Formatierungsprobleme zu bekommen, abhängig von der Anzahl der erstellten CSS-Stile. Vielleicht schreiben Sie einen klassenbasierten Stil, der eine bestimmte Schriftart und -größe definiert. Wenn Sie die Regel aber auf einen Absatz anwenden, passiert gar nichts! Probleme dieser Art haben normalerweise mit der Kaskade zu tun. Selbst wenn Sie glauben, dass die direkte Zuweisung einer Klasse ausreicht, um ihre Formatierung auf das Tag anzuwenden, kann es immer noch einen Stil mit höherer Spezifität geben.

Dieses Problem lässt sich auf verschiedene Arten umgehen. Zum einen können Sie !important verwenden, um dafür zu sorgen, dass die Eigenschaft immer angewandt wird. Allerdings kann die `!important`-Methode schnell nach hinten losgehen, weil Sie nicht wissen, ob Sie nicht eines Tages eine `!important`-Regel überschreiben müssen. Im Folgenden finden Sie zwei weitere Möglichkeiten, die Kaskade auszutricksen.

Die Spezifität ändern

Im oberen Teil sehen Sie ein Beispiel dafür, wie die Regeln für ein bestimmtes Tag trotz recht hoher Spezifität überschrieben werden. Meistens ist es jedoch recht einfach, die Spezifität für eine der Regeln anzupassen. So können Sie sich die `!important`-Methode für die echten Notfälle sparen. Im oberen Teil gibt es zwei Regeln für den ersten Absatz. Die Klasse .einleitung hat eine geringere Spezifität als `#seitenleiste p`. Dadurch werden die Eigenschaftswerte von .einleitung nicht auf den Absatz angewendet. Allerdings können Sie die Spezifität des Klassen-Selektors erhöhen, indem Sie die ID des umgebenden Elements einbeziehen: #seitenleiste .einleitung.

Selektives Überschreiben

Die Feinabstimmung für Ihr Design können Sie auch vornehmen, indem Sie die Stile auf bestimmten Seiten selektiv überschreiben. Nehmen wir beispielsweise ein externes Stylesheet namens allgemein.css, das Sie in jede Seite Ihrer Site einbinden. Das Stylesheet enthält die allgemeinen Darstellungsregeln wie Schriftart

und -farbe für <h1>-Tags, das Aussehen für Formularelemente usw. Vielleicht sollen die primären Überschriften auf der Homepage aber anders aussehen als auf der übrigen Site, zum Beispiel fetter und größer. Oder der Text in den Absätzen soll auf der Homepage kleiner sein, damit Sie mehr Informationen unterbringen können. Anders gesagt: Die meisten Stildefinitionen sollen aus allgemein.css übernommen werden; für einige Eigenschaften und Tags <h1>, <p> usw.) sollen eigene Einstellungen verwendet werden.

Eine Möglichkeit ist, einfach ein internes Stylesheet anzulegen, das bestimmte CSS-Regeln aus dem externen Stylesheet überschreibt. Angenommen, allgemein.css enthält die folgende Regel:

```
h1 {
font-family: Arial, Helvetica, sans-serif;
font-size: 24px;
color: #000;
}
```

Dann können Sie das <h1> größer und in roter Schrift darstellen, indem Sie die folgende Regel in das interne Stylesheet der Homepage schreiben:

```
h1 {
font-size: 36px;
color: red;
}
```

Daraufhin wird das <h1>-Tag auf der Homepage in der Schrift Arial (aus dem externen Stylesheet), aber in der Farbe Rot und 36 Pixel groß (aus dem internen Stylesheet) dargestellt.

Info

Stellen Sie sicher, dass das externe Stylesheet im Kopfteil der HTML-Seite vor dem internen Stylesheet eingebunden wird. Hierdurch wird sichergestellt, dass die Stile aus dem internen Stylesheet höher bewertet werden, falls die Spezifität zweier Stile einmal gleich sein sollte.

Eine weitere Möglichkeit besteht darin, ein zusätzliches externes Stylesheet anzulegen, z.B. startseite.css. Diese Datei enthält die Namen der Stile und Eigenschaften, die Sie überschreiben wollen. Damit diese Methode funktioniert, muss startseite.css nach allgemein.css in die HTML-Seite eingebunden werden, wie hier gezeigt:

```
link rel="stylesheet" type="text/css" href="css/allgemein.css">
link rel="stylesheet" type="text/css" href="css/startseite.css">
```

Oder Sie verwenden @import:

```
<style type="text/css">
@import url(css/allgemein.css);
@import url(css/startseite.css);
</style>
```

Info

Außerdem können Sie Ihr Design seitenweise anpassen, indem Sie den <body>-Tags der einzelnen Seiten spezielle IDs zuweisen, z.B. <body id="startseite">, <body id="kontakt"> usw. Mithilfe von Nachfahren-Selektoren (z.B. #5tartseite .hauptteil h1) können Sie dann das Aussehen der einzelnen Tags einer bestimmten Seiten anpassen.

Jede **CSS-Stildefinition** besteht aus zwei Teilen: einem Selektor und einem Deklarationsblock. Der Deklarationsblock enthält die Formatierungsanweisungen, wie Textfarbe, Schriftgröße usw. Das ist aber nur die eine Hälfte. Die eigentliche Magie von CSS liegt in den Zeichen links von der öffnenden geschweiften Klammer – den Selektoren. Erst indem Sie CSS mithilfe der Selektoren sagen, was formatiert werden soll, erhalten Sie die vollständige Kontrolle über das Aussehen Ihrer Seite. Wenn Sie gern verallgemeinern, können Sie einen Selelktor benutzen, der sich auf mehrere Seitenelemente gleichzeitig auswirkt. Für Leute, die sich lieber auf die Details konzentrieren, gibt es ebenfalls spezielle Selektoren, mit denen Sie ein bestimmtes Element oder eine Reihe ähnlicher Elemente auswählen können. **CSS-Selektoren** geben Ihnen die Kontrolle. Den richtigen Gebrauch lernen Sie in diesem Kapitel.

Typ-Selektoren: Stile für die ganze Seite

Typ-Selektoren (auch Element- oder Tag-Selektoren genannt) sind äußerst effiziente Werkzeuge für die Definition von Stilen, da sie sich auf jedes Vorkommen eines bestimmten HTML-Tags einer Seite auswirken. Mit Typ-Selektoren können Sie mit äußerst geringem Aufwand umfassende Änderungen an einer oder mehreren Seiten vornehmen. Sollen beispielsweise alle Absätze die gleiche Schrift, Schriftfarbe und -größe haben, reicht es, eine Regel zu erstellen, die den p-Selektor (analog zum <p>-Tag) verwendet. Man kann sagen, der Typ-Selektor legt fest, wie der Browser ein bestimmtes Tag darstellen soll (daher der alternative Name).

Bevor es **CSS** gab, mussten Sie den zu formatierenden Text mit <font>-Tags umgeben. Um jedem Absatz das gleiche Aussehen zu verleihen, musste <font> teilweise mehrere Hundert Male pro Webseite benutzt werden. Das war nicht nur eine Menge Arbeit, sondern erhöhte durch den zusätzlichen HTML-Code auch

die Ladezeit der Seiten. Bei der Verwendung von Typ-Selektoren brauchen Sie überhaupt keine Änderungen am HTML-Code vorzunehmen – erstellen Sie einfach eine **CSS-Regel** und lassen Sie den Browser den Rest für Sie erledigen.

> Typ-Selektoren sind in einer CSS-Regel leicht zu erkennen, da sie nach dem **HTML-Tag** benannt sind,

auf das sie angewendet werden sollen: `p`, `h1`, `table`, `img` usw.

Typ-Selektoren haben aber auch gewisse Nachteile. Wie sollen Sie vorgehen, wenn einige Absätze eine eigene Formatierung erhalten sollen, die von der Standarddefinition abweicht? Ein einfacher Typ-Selektor funktioniert hier nicht, weil der Webbrowser nicht zwischen den einzelnen <p>-Tags unterscheiden kann. Er weiß nicht, welche Absätze in Lila und fett gedruckter großer Schrift hervorgehoben werden sollen und welche die schwarze Standardschrift behalten sollen. Glücklicherweise hat CSS auch hierfür eine Lösung: Klassen-Selektoren, die im folgenden Abschnitt beschrieben werden.

Klassen-Selektoren: zielgenaue Kontrolle

Wenn Sie nicht wollen, dass jedes Vorkommen eines Absatzes oder einer Überschrift gleich dargestellt wird, können Sie in CSS den sogenannten Klassen-Selektor (dessen Namen Sie frei wählen können) verwenden und damit spezielle Regeln für bestimmte Teile des HTML-Codes festlegen. Möglich wäre beispielsweise eine Klasse mit dem Namen `.copyright`, die nur für den Absatz gilt, der den Copyright-Hinweis enthält, ohne dass andere Absätze hiervon betroffen wären.

Mit Klassen-Selektoren können Sie unabhängig vom Tag spezielle Regeln für einzelne HTML-Elemente definieren. Soll beispielsweise nicht das gesamte <p>-Tag verändert werden, sondern nur ein bestimmter Satz darin, können Sie diesen über einen Klassen-Selektor separat gestalten, Sie können mit dem Klassen-Selektor sogar mehrere Elemente ansprechen, die in unterschiedlichen HTML-Tags stehen. Auf diese Weise können z.B. für Absätze und sekundäre überschriften „h2“ gemeinsame Regeln festgelegt werden, beispielsweise die Farbe und eine bestimmte Schrift, mit der Sie spezielle Informationen hervorheben wollen. Im Gegensatz zu den Einschränkungen der HTML-Tags können Sie so viele Klassen-Selektoren erstellen, wie Sie wollen, und Sie an beliebigen Stellen in Ihrer Seite einsetzen.

Info

Wenn Sie nur ein paar Wörter innerhalb eines anderen Tags mit einem Klassen-Selektor ansprechen wollen, brauchen Sie die Hilfe des -Tags.

Möglicherweise haben Sie schon bemerkt, dass dem Namen von Klassen-Selektoren immer ein Punkt (`.`) vorangestellt wird, wie etwa bei `.copyright` oder `.spezial`. Dies ist eine der folgenden Regeln, die Sie bei der Benenung einer Klasse beachten müssen:

- Sämtliche Klassen-Selektoren müssen mit einem Punkt beginnen. Auf diese Weise können Webbrowser Klassen-Selektoren von anderen Selektoren in Ihrem Stylesheet unterscheiden.
- Die Namen von CSS-Klassen dürfen nur aus Buchstaben (A bis Z, a bis z, keine Umlaute usw.), Bindestrichen (`-`) und Unterstrichen _) bestehen.
- Nach dem Punkt muss der Name prinzipiell mit einem Buchstaben beginnen. Dinge wie ._zwei-und-vierzig, .-42 oder .42-ist-die-antwort funktionieren also nicht. Schreiben Sie dagegen .die-antwort-ist_42, ist alles in Ordnung.
- Bei Klassennamen wird zwischen Groß- und Kleinschreibung unterschieden. CSS sieht .SEITENLEISTE und .seitenleiste als zwei unterschiedliche Klassennamen an.

Abgesehen von den Namensregeln verwenden Sie Klassen-Selektoren auf die gleiche Weise wie Typ-Selektoren. Schreiben hinter den Selektor einfach einen Deklarationsblock, der sämtliche Stildefinitionen enthalten kann, die Sie wünschen:

```
.spezial {
color: #FF0000;
font-family:"Monotype Corsiva";
}
```

Da mit Typ-Selektoren definierte Stilregeln auf alle Tags einer Webseite anwendbar sind, brauchen Sie diese nur im Kopfteil einer Seite (oder in einem externen Stylesheet) zu defmieren – die dafür benötigten HTML-Tags sind ja schon da. Die zusätzliche Freiheit bei der Verwendung von Klassen-Selektoren bedeutet etwas mehr Arbeit und geschieht in zwei Schritten: Nachdem Sie die Regel erstellt haben, müssen Sie angeben, worauf diese Regeln angewendet werden sollen. Hierfür erweitern Sie das **HTML-Tag**, das formatiert werden soll, um ein `class`-Attribut.

Nehmen wir einmal an, Sie haben die Klasse `.spezial` erzeugt, um damit bestimtme Seitenelemente hervorzuheben. Um diesen Stil auf einen bestimmten Absatz anzuwenden, versehen Sie dessen <p>-Tag mit folgendem `class`-Attribut:

```
<p class="spezial">
```

Info

Im **HTML-Code** wird der Klassenname ohne vorangestellten Punkt geschrieben. Der Punkt wird nur benötigt, damit CSS die Klasse von anderen Selektoren unterscheiden kann.

Sieht ein Webbrowser dieses Tag, weiß er, dass die Formatierungsregeln der .spezial-Definition auf den markierten Absatz angewendet werden sollen. Wenn Sie das -Tag verwenden, können Sie eine CSS-Klasse auch nur auf Teile eines Absatzes, einer Überschrift oder was auch immer anwenden.

> Sobald Sie eine klassenbasierte Stildefinition erstellt haben, können Sie diese auf jedes beliebige Tag

der Seite(n) anwenden. Auch wenn Ihnen damit fast grenzenlose Gestaltungsmöglichkeiten zur Verfügung stehen, sind Klassen nicht immer das beste Mittel nur ein CSS-basiertes Seitenlayout. Und hier kommt der ID-Selektor ins Spiel. Mit ihm können Sie eine Formatierungsregel festlegen, die auf jeder Seite immer nur genau einmal verwendet werden darf, wie im folgenden Abschnitt beschrieben.

ID-Selektoren: Regeln für ein bestimmtes Seitenelement

In CSS ist der ID-Selektor für die Identifizierung eines bestimmten Teil seiner Seite reserviert. Dies kann beispielsweise ein Logo sein, eine Navigationsleiste oder auch der Hauptteil. Er darf nur einmal benutzt werden und kann auf diese Weise ein Element eindeutig identifizieren. Die Folge ist, dass der ID-Selektor pro HTML-Seite immer nur einmal verwendet werden kann. Anstelle eines Punkts wird dem Namen im Stylesheet ein Doppelkreuz (#) vorangestellt. Davon abgesehen, funktioniert ein ID-Selektor analog zum Klassen-Selektor. Wie schon beschrieben wurde, kommen ID-Selektoren speziell bei bestimmten JavaScriptbasierten oder sehr langen HTML-Seiten zum Einsatz. Aber darüber hinaus gibt es nur wenige Gründe, ID-Selektoren anstelle von Klassen zu benutzen.

> Die folgenden Faustregeln können als Entscheidungshilfen dienen, ob eine Klasse oder eine ID verwendet werden soll:

- Soll eine Stildefinition mehr als einmal auf einer Seite benutzt werden, müssen Sie Klassen verwenden, z.B. wenn Ihre Seite mehrere Fotos enthält, die alle mit dem gleichen Rahmenversehen werden sollen.
- Verwenden Sie IDs, um verschiedenen Teile einer Seite zu markieren, die nur einmal pro Seite vorkommen. CSS-basierte Layouts verwenden oftmals IDs, um einmalige Teile einer Seite zu markieren, etwa eine Seitenleiste oder eine Fußzeile.
- Gelegentlich können Sie ID-Selektoren verwenden, um Probleme zu umgehen, da Webbrowser den Anweisungen in ID-Selektoren eine höhere Priorität einräumen als Klassen-Selektoren. Wurden für ein Tag z.B. in einem Klassen- und in einem ID-Selektor unterschiedliche Hintergrundfarben

definiert, gewinnt der ID-Selektor.

Info

Obwohl IDs auf einer HTML-Seite nur einmal vorkommen sollen, werden beim Browser nicht gleich die Alarmsirenen heulen, weil Sie versehentlich die gleiche 10 an zwei Tags vergeben haben. Die meisten Browser stellen in solchen Fällen beide Tags mit den gleichen Stilen dar. Allerdings sollten Sie sich nicht wundern, dass Ihr HTML-Code in diesem Fall nicht validiert und Ihre Webdesignerfreunde plötzlich nicht mehr mit Ihnen sprechen.

ID-Selektoren sind genauso einfach zu benutzen wie Klassen-Selektoren. Anstelle des Punkts verwenden Sie für ID-Selektoren einfach ein Doppelkreuz (#) vor dem Namen der ID. Davon abgesehen, gelten die gleichen Namensregeln für die Benennung wie bei Klassen. Dieses Beispiel definiert Hintergrundfarbe, Breite und Höhe für das HTML-Element mit der ID kopfteil:

```
#kopfteil {
background: #CC0000;
height: 300px;
width: 720px;
}
```

Die Verwendung der IDs im HTML-Code ist ebenfalls der von Klassen ähnlich. Allerdings kommt anstelle von dass hier logischerweise das Attribut `id` zum Einsatz. Soll beispielsweise der letzte Absatz einer Seite als Copyright-Vermerk dienen, können Sie eine Stildefinition für `#copyright` anlegen und sie wie hier gezeigt dem <p>-Tag zuweisen:

```
<p id="copyright">
```

Info

Wie bei klassenbasierten Stilen wird das entsprechende Symbol (# nur im Stylesheet, nicht aber im ID-Anribut des HTML Codes verwendet – wie hier: `<div id="kopfteil">`.

Stildefinitionen für verschachteHe Tags

Die Entscheidung, ob Sie die Stildefinitionen für Ihre Seite mit Typ-Selektoren oder mit Klassen-Selektoren vornehmen, ist nicht immer einfach. Typ-Selektoren sind schnell und einfach zu benutzen, aber sie lassen jedes Vorkommen eines Tags gleich aussehen. Das kann durchaus gewünscht sein. Klassen- und ID-Selektoren geben Ihnen dagegen die Flexibilität, die einzelnen Seitenelemente individuell zu gestalten. Allerdings bedeutet das Anlegen eigener Stilregeln für die Hintergrundfarbe einer einzigen Überschrift zusätzliche Arbeit – und HTML-Code. Sie bräuchten eine Möglichkeit, die Einfachheit von Typ-Selektoren mit der Präzision von Klassen und IDs zu kombinieren. Wie sich zeigt, besitzt CSS auch hierfür eine Lösung: Nachfahren-Selektoren (auch Kontext-Selektoren genannt).

Mithilfe von Nachfahren-Selektoren können Sie mehrere Tags gleichzeitig mit denselben Stildefinitionen versorgen

(wie mit Typ-Selektoren), aber nur, wenn sie sich an einer bestimmten Stelle der Webseite befmden. Das ist, als würden Sie sagen: „He du, <a>-Tag in der Navigationsleiste, ich habe hier ein paar Formatierungen für dich. Alle anderen <a>-Tags bitte weitergehen, hier gibt es nichts zu sehen.“

Mit Nachfahren-Selektoren können Sie ein Tag basierend auf seinem Kontext zu anderen Tags formatieren (daher der alternative Name Kontext-Selektor). Um zu verstehen, was hier vor sich geht, müssen wir uns den HTML-Code etwas genauer ansehen. Das Gute daran ist, dass Ihnen das Verständnis der Funktionsweise von Nachfahren-Selektoren hilft, auch gleich eine Reihe anderer Selektor-Typen zu verstehen, die wir später in diesem Kapitel behandeln.

Der HTML-Stammbaum

Der **HTML-Code**, aus dem jede Webseite besteht, hat eine gewisse Ähnlichkeit mit einem Familienstammbaum, bei dem die HTML-Tags den einzelnen Familienmitgliedern entsprechen. Das erste HTML-Tag einer Seite, <html>, ist quasi der Uropa aller anderen Tags. Das <htm> umgibt die <head>-und <body>-Tags. Damit gilt <html> als Vorfahren-Element für die anderen beiden. Andersherum werden <head> und <body> als Nachfahren-Elemente von <htm> bezeichnet. Das <title>-Tag ist seinerseits ein Nachfahre von <head>, aber gleichzeitig auch von <html>

```
<html>
<head>
<title>Ein einfaches Dokument</title>
</head>
<body>
<h1>Oberschrift</h1>
<p>Ein Absatz mit etwas <strong>wichtigem</strong>Text.</p>
</body>
</html>
```

Wenn Sie diesen HTML-Code betrachten, werden die verwandtschaftlichen Verhältnisse der Tags schnell klar. Das <html>-Tag hat zwei Nachkommen: <head> und <body>. Diese beiden Tags enthalten wiederum andere Tags, die ihrerseits weitere Tags enthalten können. Diese Art von Diagramm lässt sich für jede Webseite erstellen.

Baumdiagramme wie dieses können dabei helfen, die Beziehungen der Elemente einer Seite leichter zu verstehen.

Viele in diesem Kapitel behandelten Selektoren, inklusive der Nachfahren-Selektoren, basieren auf diesen Beziehungen. Die wichtigsten Beziehungen sind:

- Vorfahren-Element. Wie bereits zu Beginn dieses Abschnitts erklärt wurde, werden HTML-Tags, die andere Tags enthalten, als deren Vorfahren-Elemente bezeichnet. Das `<html>`-Tag ist der Vorfahre aller anderen Tags, während `<body>` der Vorfahre sämtlicher in ihm enthaltenen Tags ist, in unserem Beispiel `<h1>`, `<p>` und `<strong>`.
- Nachfahren-Element. Ein Tag, das sich innerhalb eines oder mehrerer anderer Tags befindet, bezeichnet man als Nachfahren-Element. Das `<body>`-Tag ist ein Nachfahre von `<html>`, während das `<title>`-Tag sowohl von `<head>` als auch von `<html>` abstammt.
- Eltern-Element. Als Eltern-Element bezeichnet man den nächsten direkten Vorfahren eines Elements. Das erste Tag ist immer eine direkte Verbindung zu einem in der Hierarchie tiefer stehenden Tag. Das bedeutet, `<html>` ist das Eltern-Element für `<head>` und `<body>`, aber nicht für die anderen Tags. Außerdem ist `<p>` das Eltern-Element für das `<strong>`-Tag.
- Kind-Element. Der nächste direkte Nachfahre eines Eltern-Elements wird als Kind-Element bezeichnet. Es sind sowohl `<h1>` als auch `<p>` Kind-Elemente des `<body>`-Tags. Das `<strong>`-Element ist dagegen kein Kind von `<body>`, sondern von `<p>`.
- (Benachbarte) Geschwister-Elemente. Tags, die Kind-Elemente des gleichen Tags sind, werden als Geschwister-Elemente bezeichnet. wie Brüder und Schwestern. In einem HTML-Diagramm stehen Geschwister-Elemente nebeneinander und haben die gleichen Eltern. Es sind `<head>` und `<body>` Geschwister, genau wie `<h1>` und `<p>`.

Damit ist die Familienidylle in CSS 2.1 glücklicherweise erschöpft. Sie müssen sich also kein Sorgen um Onkel, Tanten oder entfernte Cousinen dritten Grades machen. (Es gibt allerdings Gerüchte. dass CSS 10 auch Schwiegermütter und Schwippschwager unterstützen soll.)

Selektoren für Nachfahren-Elemente verwenden

Mit Nachfahren-Selektoren können Sie sich die Vorteile des HTML-Stammbaums zunutze machen, indem Sie Tags unterschiedlich formatieren, wenn Sie innerhalb anderer Tags stehen. Angenommen, Sie wollten innerhalb einer `<h1>`-Überschrift mittels eines `<strong>`-Tags ein Wort besonders hervorheben. Die meisten Browser stellen aber sowohl Überschriften als auch als `<strong>` markierte Textteile fett dar, sodass für den Betrachter kein Unterschied zwischen dem hervorgehobenen Wort und der restlichen Überschrift zu sehen ist. Die Verwendung eines Typ-Selektors zum Ändern der Farbe des `<strong>`-Tags ist hier keine gute Wahl, weil Sie dadurch – ob Sie wollen oder nicht – auch die Farbe aller anderen `<strong>`-Tags verändern. Mit dem Selektor für Nachfahren-Elemente können Sie dagegen gezielt nur solche `<strong>`-Tags verändern, die innerhalb von `<h1>`-Tags stehen.

Der CSS-Code, der unser `<h1>` und `<strong>`-Dilemma löst. sieht so aus:

```
h1 strong {
color: red;
}
```

Durch diese Schreibweise werden nur Vorkommen von rot dargestellt, die sich innerhalb von <h1>-Tags befinden. Alle anderen -Tags sind nicht betroffen.

Selektoren für Nachfahren-Elemente wirken nur auf Elemente, die in andere Elemente verschachtelt sind. Sie folgen dabei exakt dem Muster von Vor- und Nachfahren des HTML-Stammbaums.

Einen Nachfahren-Selektor erstellen Sie, indem Sie den Vorfahren und den Nachfahren gemäß dem HTML-Stammbaum nebeneinanderstellen. Hierbei steht der „älteste“ Vorfahre ganz links und der „jüngste“ Nachfahre am weitesten rechts. Wenn Sie beispielsweise drei Links <a>, die sich innerhalb einer ungeordneten Liste befinden. sowie einen weiteren Link innerhalb eines Absatzes (<p>). Damit die Links innerhalb der Liste eine vom Link im Absatzunterschiedliche Formatierung erhalten, können Sie den folgenden Nachfahren-Selektor einsetzen:

```
li a {
font-family: Arial;
}
```

Diese Regel lautet auf Deutsch übersetzt: „Stelle alle <a>-Tags (Links) innerhalb eines Listeneintrags (<li>) in der Schrift Arial dar.“ Hierbei kann ein Nachfahren-Selektor auch mehr als zwei Elemente enthalten. Die folgenden Beispiele funktionieren alle als gültige Selektoren für die <a>-Tags innerhalb der ungeordneten Liste:

```
ul li a
body li a
html li a
html body ul li a
```

Info

Ein Grund für die Verwendung von zusätzlichen Elementen in Nachfahren-Selektoren kann die Definition verschiedener Regeln zur Formatierung des gleichen Tags sein. Die Formatanweisungen eines sehr deutlich aus formulierten Nachfahren-Selektors (z.B. das vierte Beispiel oben) können einfache Klassen- oder Typ-Selektoren überschreiben.

Diese vier Selektoren, die eigentlich alle das Gleiche tun, zeigen, dass Sie nicht unbedingt die gesamte

„Abstammungslinie“ des zu formatierenden Tags zu nennen brauchen. Im zweiten Beispiel,`body li a`, wird `ul` gar nicht gebraucht. Der Selektor funktioniert, sobald es ein <a>-Tag gibt, das irgendwo in seinem Stammbaum ein `<li>`-Tag zum Vorfahren hat (das seinerseits ein Nachfahre des `<body>`-Tags ist). Das bedeutet, dass dieser **Selektor** genauso gut auf ein <a>-Tag zutreffen kann, das sich innerhalb eines -Tags befindet, das sich in einem `<strong>`-Tag befindet, das von einem `<li>` umgeben wird – und so weiter.

> Diese Technik ist nicht einmal auf Typ-Selektoren beschränkt.

Sie können komplexe Nachfahren-Selektoren erzeugen, indem Sie verschiedene Selektoren kombinieren. Vielleicht sollen nur die Links gelb dargestellt werden, die sich innerhalb des einleitenden Absatzes (markiert mit der Klasse einleitung) befinden. Die folgende Stildefinition macht genau das:

```
p.einleitung a {
color: yellow;
}
```

Die Kurzfassung: „Wende diese Stilregel auf jeden Link (a) an, der ein Nachfahre eines Absatzes (p) ist, dem die Klasse einleitung zugewiesen wurde.“ Beachten Sie, dass zwischen p und `.einleitung` kein Leerzeichen steht. Auf diese Weise teilen Sie CSS mit, dass die Klasse `.einleitung` ausdrücklich mit dem <p>-Tag verbunden sein muss.

Wenn Sie dagegen ein Leerzeichen einfügen. erhalten Sie einen anderen Effekt:

```
p .einleitung a {
color: yellow;
}
```

Diese nur scheinbar winzige Änderung sorgt dafür, dass der Selektor nun auf <a>-Tags zutrifft, die sich innerhalb eines beliebigen anderen Tags befinden, der mit der Klasse .einleitung markiert wurde. Dieser beliebige andere Tag soll seinerseits von einem <p>-Tag umgeben sein.

Lassen Sie den Namen des letzten Vorfahren weg (in unserem Fall das p). erhalten Sie eine flexiblere Stildefmition:

```
.einleitung a {
color: yellow;
}
```

Diese Regel wird auf alle <a>-Tags angewandt. die sich innerhalb eines beliebigen anderen Tags befmden. , <h1>. <table> oder was auch immer, denen die Klasse .einleitung zugewiesen wurde. Das Vorfahren-Element dieser Tags ist unwichtig.

Nachfahren-Selektoren sind ziemlich schwere Geschütze in Ihrem CSS-Arsenal, die sehr vielseitig eingesetzt werden können.

Stile für mehrere Tags auf einmal definieren

Gelegentlich müssen Sie schnell mal mehrere verschiedene Elemente mit den gleichen Stilen versehen. Sollen beispielsweise alle Überschriften die gleiche Schrift und Farbe bekommen, ist die Defmition eigener Stile für jede Überschrift. <h1>. <h2> usw., deutlich zu viel Arbeit. Bei einer späteren Änderung müssen Sie außerdem entsprechend viele Stildefinitionen anpassen, die im Grunde die gleichen Regeln enthalten. Ein besserer Ansatz besteht darin, die Selektoren zu gruppieren. um so die gleiche Regel für mehrere Selektoren nur einmal defmieren zu müssen.

Selektoren gruppieren

Um mehrere Selektoren zu gruppieren, erstellen Sie einfach eine durch Kommata getrennte Liste. Wenn Sie für alle überschriften die gleichen Deklarationen verwenden wollen, können Sie das folgendermaßen schreiben:

```
h1, h2. h3, h4, hS. h6 {
color: #FICD33;
}
```

Dieses Beispiel besteht einfach aus einer Gruppe von Typ-Selektoren, Sie können aber auch jeden anderen gültigen Selelktor (oder eine beliebige Kombination von Selektoren) in die Liste aufnehmen. Die folgende Deklaration gilt für alle primären Überschriften, Absätze, sämtliche mit der Klasse `.copyright` markierten Tags und das Tag mit der ID kopfteil:

```
h1, p, .copyright, #kopfteil {
color: #F1CD33;
}
```

Info

Manchmal sollen bestimmte Seitenelemente nur einen Teil der Formatierungen miteinander teilen. Gleichzeitig soll es für bestimmte Tags aber auch eigene Regeln geben. In solchen Fällen können Sie die gemeinsamen Anweisungen gruppieren und danach mit speziellen Regeln für die einzelnen Elemente

präzisieren. Die Fähigkeit, mehrere Regeln für das gleiche Element festzulegen, ist eine sehr mächtige Eigenschaft von CSS.

Der universelle Selektor (Asterisk, *)

Wenn Ihnen die Möglichkeit, Selektoren zu gruppieren, immer noch nicht ausreicht, besitzt CSS auch noch den universellen Se1ektor: Dabei handelt es sich um einen Asterisk (*), der in seiner Grundform einfach alle Tags einer Webseite auswählt.

Vielleicht sollen aus irgendeinem Grund ja alle Tags Ihrer Seite in fetter Schrift dargestellt werden. Dann könnten Sie das tun, indem Sie eine Gruppe aller Selektoren anlegten:

```
a, p, img, h1, h2, h3, h4, h5 ... und so weiter ... {
font-weight: bold;
}
```

Der Asterisk bietet dagegen eine wesentlich kürzere Schreibweise, um CSS anzuweisen, alle HTML-Tags auszuwählen:

```
* {
font-weight: bold;
}
```

In seiner kultivierteren Form kann der universelle Se1ektor auch als Teil eines Nachfahren-Selektors oder anderer komplexer Konstruktionen verwendet werden. Wenn Sie einen Stil auf alle Elemente anwenden wollen, die vom Tag mit der ID kopfteil abstammen, können Sie beispielsweise schreiben:

```
#kopfteil * {
font-weight: bold;
}
```

Da der universelle Selektor sich auf kein bestimmtes Tag bezieht, ist es gelegentlich schwer, vorherzusagen, ob er auf alle Seiten und deren Tags in einer Website die gleiche Wirkung hat. Um viele verschiedene Seitenelemente zu formatieren, verlassen sich die Webdesign-Gurus gern auch auf die sogenannte Vererbung – ein Merkmal von CSS.

Pseudoklassen und Pseudoelemente

Manchmal soll eine Regel für Teile einer Webseite gelten, die kein eigenes Tag besitzen, aber trotzdem leicht zu identifizieren sind, wie beispielsweise die erste Zeile eines Absatzes oder ein Link, über dem sich gerade der Mauszeiger befindet. **CSS** besitzt eine Handvoll Selelktoren für diese Dinger, namentlich

Pseudoklassen und Pseudoelemente.

Stildefinitionen für Links

Allein für die Formatierung von Links gibt es vier verschiedene Pseudoklassen. Abhängig davon. wie der Benutzer mit den Links interagiert, können Sie unterschiedliche Stile festlegen. Die entsprechenden Pseudoklassen lauten:

- **a:link** trifft auf einen Link zu, dem der Benutzer noch nicht gefolgt ist. Der Mauszeiger befindet sich nicht über dem Link, und der Link wird momentan nicht angeklickt. Mit `a:link` definierte Stile gelten für normale, noch nicht angeklickte Weblinks.
- **a:visited** bezeichnet einen Link, der (gemäß dem Verlauf des Webbrowsers) bereits besucht wurde. Sie können diesen Links ein anderes Aussehen geben, um Ihren Besuchern zu verdeutlichen, dass sie hier schon einmal waren.
- **a:hover** findet Links, über denen sich gerade der Mauszeiger befindet. Diese sogenannten Rollover-Effekte sind nicht nur zum Spaß da, sie können für Buttons oder Navigationselemente sinnvolle visuelle Rückmeldungen geben.

Sie können die Pseudoklasse `:hover` übrigens nicht nur für Links verwenden. So ist es durchaus möglich, auch Textteile in einem bestimmten Absatz oder einem `<div>`-Container hervorzuheben, wenn der Besucher mit der Maus darüberfahrt.

Info

Im Internet Explorer 6 und früheren Versionen funktioniert `:hover` nur für Links. Wenn Sie partout nicht auf Version 7 oder 8 aktualisieren wollen (oder dürfen), sprechen wir Ihnen unser herzliches Beileid aus und empfehlen Ihnen, das Problem mit dem vorgestellten JavaScript zu umgehen.

- **a:active** definiert Stilregeln für den Moment, in dem der Link gerade angeklickt wird. So definierte Stile gelten also nur für den Augenblick zwischen Klicken und Loslassen der Maustaste.

Info

Vermutlich können Sie ein langes und gesundes Leben führen, ohne jemals mit den Selektoren in den folgenden Abschnitten zu tun zu haben. Viele Webdesigner benutzen sie überhaupt nicht. Die Selektoren die Sie bisher kennen für Typ, Klasse, ID, Nachfahren und so weiter, reichen meistens vollkommen aus, um schöne, praktische und leicht wartbare Webserren zu erstellen.

Noch mehr Pseudoelemente und -klassen

In der CSS-Speziftkation werden neben bereits behandelten weitere mächtige Pseudoklassen und -

elemente deftniert. Leider unterstützt der am zweithäuftgsten benutzte Browser – Internet Explorer 6 für Windows – diese so gut wie gar nicht. Obwohl diese Merkmale offIziell Teil von CSS sind, werden viele Websurfer mit diesen Mitteln erstellte Designelemente nicht sehen können (es sei denn, Sie aktualisieren auf Internet Explorer 7 oder höher bzw. benutzen einen wirklich standardkonformen Browser wie z.B. Firefox). Ist es Ihnen nicht möglich, ihren Browser zu wechseln oder zu aktualisieren, finden Sie im JavaScript eine Möglichkeit zum Umgehen des Problems.

:before

Das Pseudoelement `:before` besitzt eine Fähigkeit, die Sie bei normalen Selektoren vergeblich suchen. Es kann Inhalte vor einem gegebenen Element einfügen. Vielleicht wollen Sie vor einem wichtigen Absatz die Worter „HEISSER TIPP!“ einfügen, um hervorzuheben. Anstatt diesen Text in den HTML-Code einzubauen, können Sie dies auch mit dem Selektor `:before` erledigen. Dieser Ansatz hält nicht nur Ihren HTML-Code klein, sondern ermöglicht auch spontane Anpassungen der Nachricht, ohne hierfür den HTML-Code anfassen zu müssen. Die Änderung von „HEISSER TIPP!“ zu „Wissenswerte Dinge“ lässt sich so mit ein wenig CSS für die gesamte Website auf einmal vornehmen. (Der Nachteil ist, dass diese spezielle Nachricht für Browser, die kein CSS verstehen oder :before nicht unterstützen, unsichtbar bleibt.)

Um `:before` zu verwenden, müssen Sie erst einmal eine Klasse anlegen (z.B. `.tipp`) und diese den Absätzen zuweisen, denen der Text vorangestellt werden soll, wie hier: `<p class="tipp">`. Danach fügen Sie den Nachrichtentext in Ihr Stylesheet ein:

```
p.tipp:before {
content: "HEISSER TIPP!"
}
```

Sobald ein Webbrowser die Klasse .tipp in einem <p>-Tag findet, stellt er dem Absatz pflichtgemäß den Text „HEISSER TIPP!“ voran.

Das Stichwort für Text, der auf diese Weise eingefügt wird, heißt erzeugter Inhalt, weil die Webbrowser diesen nach Bedarf erzeugen. Im **HTML-Quellcode** der Seite werden Sie diesen Text nicht finden. Auch wenn Ihnen das vielleicht nicht aufafllt, erzeugen Webbrowser ständig ihren eigenen Inhalt, z.B. Markierungen oder Zahlen vor Listeneinträgen. Wenn Sie unbedingt wollen, könnten Sie mit :before sogar eigene Aufzählungszeichen definieren.

Trotz seines sicherlich großen Potenzials ignorieren Internet Explorer 6 und dessen Vorgänger dieses Pseudoelement geflissentlich.

:after

Wie sein Zwillingsbruder `:before`, fügt auch das Pseudoelement `:after` erzeugten Inhalt in ein Dokument ein – diesmal allerdings nach dem Element. Auf diese Weise können Sie beispielsweise schließende Anführungszeichen nach einem Zitatblock erzeugen.

:first-child

Erinnern wir uns kurz, was in der Analogie des HTML-Stammbaums ein Kind-Element ist: ein beliebiges Tag, das von einem anderen Tag umgeben ist. (<h1>, <p>, <h2> und <ul> sind Kind-Elemente des <body>-Tags.) Mit dem Pseudoelement `:first-child` können Sie spezielle Stile für das erste der vielen Kind-Elemente festlegen.

So kann das Tag <ul>, mit dem eine ungeordnete Liste erzeugt wird, beispielsweise eine ganze Reihe von Listeneinträgen als Kind-Elemente besitzen. Um nur den ersten Eintrag fett darzustellen, können Sie die folgende Regel verwenden:

```
li:first-child {
font-weight: bold;
}
```

Da `:first-child` nur den Namen des Kind-Elements enthält (aber niemals den des Eltern-Elements), formatiert der obige Stil beliebige -Tags, die das erste Kind-Element irgendeines Tags sind, also nicht nur von <ul>. Listenelemente kommen immer in Listen vor. Daher können Sie davon ausgehen, dass der Selektor `li:first-child` für alle Listen einer Seite gilt – geordnet oder ungeordnet. Bei anderen Tags ist die Verwendung von `first-child` dagegen nicht ganz so einfach. So hätte der Selektor `p;first-child` keinerlei Auswirkungen, wenn das <p>-Tag zwar ein Kind-Element von <body> ist, aber nicht das erste; das ist <h1>.

Da sich die Eltern-Kind-Beziehung bei jeder Anpassung der Webseite ändern kann, ist es unter Umständen schwierig, vorherzusagen, wie sich der Selelctor `:first-child` bei einer Weiterentwicklung Ihrer Site verhält. Hinzu kommt, dass er im Internet Explorer 6 und früheren Versionen nicht funktioniert – ein weiterer Grund. ihn nur zu benutzen, wenn es absolut notwendig ist. (Immerhin bietet die Version 7 eine gewisse Unterstützung für `:first-child`.)

:focus

Die Pseudoklasse `:focus` funktioniert ähnlich wie `:hover`. Während `:hover` aktiv wird, sobald ein Benutzer mit dem Mauszeiger über einen Link fahrt, wird `focus` auf das momentan aktive Element angewandt. „Den Fokus“ erhält ein Element beispielsweise durch Anklicken oder durch Betätigen der Tabulatortaste. Dieser Klick ist normalerweise der einzige Hinweis darauf, welches Element gerade aktiv ist.

Daher ist der als Pseudoklasse angelegte Selektor `:focus` in erster Linie sinnvoll, um Ihren Benutzern Rückmeldungen zu geben. Dies kann beispielsweise die Änderung der Hintergrundfarbe des Textfelds sein, in das gerade Eingaben gemacht werden. (Einzeilige Text- und Passwortfelder. aber auch größere

<textarea>-Felder sind häufige Einsatzgebiete für den :focus-Selektor.) Das folgende Beispiel ändert die Hintergrundfarbe eines Texteingabefelds in Gelb, wenn ein Benutzer es per Mausklick oder Tabulatortaste aktiviert:

```
input:focus {
background-color: #FFFFCC;
}
```

Die Stildefinitionen haben nur so lange Gültigkeit, wie das betreffende Element den Fokus hat. Sobald der Benutzer in ein anderes Feld oder zu einem anderen Element der Seite wechselt, verliert die Textbox den Fokus wieder und damit auch die entsprechenden :focus-Stilregeln.

Info

Die Schreibweise von Selektoren zu lernen, erinnert gelegentlich etwas an das Lernen von Hieroglyphen. Eine Übersetzung der CSS-Selektoren in (englische oder spanische) Menschensprache finden Sie auf der Website von Selectorade unter selectoracle. Ähnlich wie beim **CSS-Validator** des W3C können Sie CSS-Code oder die Adresse einer Webseite bzw. eines **Stylesheets** eingeben. Als Antwort erhalten Sie eine Beschreibung im Klartext zurück, die Ihnen sagt, auf welche Elemente einer Seite sich eine Stildefinition auswirkt.

Selektoren für Fortgeschrittene

Die CSS-Spezifikation enthält weitere mächtige Selektoren, mit denen Sie die Gestaltung Ihrer Website noch genauer steuern können. Ähnlich einigen bereits vorgestellten Selektoren funktionieren auch diese erst ab dem Internet Explorer 7. (Für ältere Versionen haben Sie möglicherweise mit der vorgestellten JavaScript-Lösung Glück.)

Kind-Selektoren

Kind-Selektoren besitzen gegenüber den bereits in diesem Kapitel vorgestellten Nachfahren-Selektoren eine noch höhere Präzision. Hiermit können Sie die Kind-Elemente, also direkte Nachfahren. eines anderen Elements (Tags) mit Stilen versehen. Kind-Selektoren verwenden ein Größer-als-Zeichen (>), um die Beziehung zwischen den beiden Elementen zu verdeutlichen. Mit dem Selektor body > h1 können Sie beispielsweise Stile für alle <h1>-Tags definieren, die Kind-Elemente des <body>-Tags sind.

Im Gegensatz zum Selektor für Nachfahren-Elemente (der alle Nachkommen eines Tags auswählt, also Kinder, Enkel, Urenkel usw.), können Sie mit dem Kind-Selektor genau festlegen, welches Kind-Element welches Vorfahren Sie genau meinen. Mit dem bereits bekannten Nachfahren-Selektor body h2 werden beide <h2>-Tags ausgewählt. Obwohl sich beide sekundären Überschriften innerhalb der <body>-Tags befinden, ist nur die zweite ein direkter Nachfahr (ein „Kind“ von <body>. Das erste <h2>-Tag befindet

sich dagegen innerhalb eines -Paars und gilt daher als dessen Kind (und als Enkelkind von <body>. Wenn Sie ausdrücklich das erste <h2> auswählen wollen, benötigen Sie stattdessen den Kind-Selektor `div > h2`.

Benachbarte Geschwister-Elemente

Neben Eltern-Kind-Beziehungen gibt es auch noch andere Verwandtschaftsverhältnisse, die im HTML-Stammbaum vorkommen. Gelegentlich wollen Sie ein Tag nicht anhand seiner Vorfahren, sondern seiner Nachbarn auswählen. Hierbei werden Tags, die das gleiche Eltern-Element besitzen und im HTML-Code direkt aufeinanderfolgen, als benachbarte Geschwister-Elemente bezeichnet. So ist das -Tag ein benachbartes Geschwister-Element von <h1>. Das <p>-Tag ist ein benachbartes Geschwister-Element von <h2> usw.

Mit dem Selektor für benachbarte Geschwister-Elemente können Sie beispielsweise den ersten auf eine <h1>-Überschrift folgenden Absatz anders formatieren als die übrigen Absätze. Vielleicht soll zwischen der überschrift und dem Absatz kein Zwischenraum sein, wie sonst üblich. Oder der erste Absatz soll eine bestimmte Farbe oder Schrift verwenden, weil er als Einleitung dienen soll.

Das Symbol für den Selektor für benachbarte Geschwister-Elemente ist ein Pluszeichen (+), mit dem die Elemente verbunden werden. Um beispielsweise jeden Absatz (<a>) auszuwählen, der auf eine sekundäre Überschrift (<h2>) folgt, können Sie dies mit dem Selelctor `h2 + p` erledigen. Hierbei erhält das letzte Element (hier <p> die Formatierung – aber nur, wenn es sich direkt neben seiner Schwester, der sekundären Überschrift <h2>. befindet.

Attribut-Selektoren

Neben den bereits beschriebenen Methoden bietet CSS die Möglichkeit, Tags anhand ihrer Attribute auszuwählen. Vielleicht sollen nur die wichtigen Fotos einer Seite einen Rahmen erhalten. Ihr Logo, Buttons und andere kleine Dinge, die ebenfalls per <img>-Tag in die Seite eingebunden sind, sollen davon jedoch ausgenommen sein. Zum Glück erinnern Sie sich, dass Sie Ihren Fotos mit dem Attribut `title` kurze Beschreibungen mitgegeben haben. Dadurch können Sie jetzt einen Attribut-Selektor verwenden, um die wichtigen Bilder von den anderen zu unterscheiden.

Info

Das **HTML-Attribut** `title` bietet eine großartige Möglichkeit, links und Bilder mit Kurzinfos (oder Popup-Meldungen) zu versehen. Es ist außerdem eine Methode, Suchmaschinen mit nützlichen Informationen zu einer Webseite zu versorgen. Näheres dazu finden Sie beispielsweise unter webdesign.about.com.

Mit Attribut-Selektoren können Sie Tags anhand bestimmter Merkmale auswählen. Im folgenden Beispiel wird ein <img>-Tag ausgewählt. das ein `title`-Attribut besitzt:

```
img[title]
```

Der erste Teil des Selektors ist der Name des Tags (`img`). Der Name des Attributs steht in eckigen Klammern direkt dahinter: `[title]`.

Hierbei sind Sie nicht auf die Namen der Attribute beschränkt. Sie können sie auch mit Klassennamen verwenden. So wählt `.foto[title]` beispielsweise jedes Element aus, dem die Klasse `foto` zugewiesen wurde und das das HTML-Attribut `title` besitzt.

Sie können die Auswahl sogar noch weiter eingrenzen, indem Sie den Selekdor nur für Elemente formulieren, die einen genau definierten Wert für ein bestimmtes Attribut besitzen. Vielleicht sollen Links, die auf einen besonderen URL zeigen, hervorgehoben werden. Hier hilft der folgende Attribut-Selektor:

```
a[href="http://www.html-info.eu"]{
color:red;
font-weight:bold;
}
```

Die Verwendung bestimmter Werte in Attribut-Selektoren ist besonders bei der Arbeit mit Formularen nützlich. Viele Formularelemente verwenden das gleiche Tag, obwohl sie unterschiedlich aussehen und sich auch so verhalten. So werden Checkbox, Textfelder. Submit-Buttons und andere Formularfelder alle durch das `<input>`-Tag gekennzeichnet. Erst durch den Wert des Attributs `type` werden sie tatsächlich unterschieden. Der Code `<input type="text">` erzeugt beispielsweise ein Textfeld, `<input type="checkbox">` dagegen ein Checkbox.

Um nur die Textfelder eines Formulars auszuwählen, können Sie beispielsweise diesen Selektor verwenden:

```
input[type="text"]
```

Info

Der Entwurf für die CSS3-Spezifikation enthält noch eine ganze Reihe weiterer Attribut-Selektoren. Leider werden auch Attribut-Selektoren vom Internet Explorer erst ab der Version 7 unterstützt. Eine Übersicht darüber, welche gängigen Browser welche Attribute unterstützen finden Sie beispielsweise unter kimblim.dk.

Selbst die komplexesten und schönsten Websites beginnen mit einer einzelnen **CSS-Stildefinition**. Mit den richtigen CSS-Regeln und Stylesheets können Sie ein vollständiges Webdesign umsetzen, das Designer inspiriert und die Besucher Ihrer Website erstaunt. Sowohl der CSS-Novize wie auch der CSS-Samurai müssen bei der Erstellung von Stilen und **Stylesheets** allerdings ein paar Regeln befolgen. In diesem Kapitel beginnen wir am absoluten Nullpunkt und lernen die Grundlagen der Verwendung und Erstellung von Stilen und Stylesheets kennen.

Anatomie einer Stildefinition

Eine einzelne Stildefinition für das Aussehen eines bestimmten Webseiten-Elements ist ziemlich einfach aufgebaut. Im Prinzip ist es lediglich eine Regel, die dem Webbrowser mitteilt, wie das betreffende Element formatiert werden soll: Überschrift in Blau, ein roter Rahmen um ein Foto, einen 150 Pixel breiten Bereich in der Darstellung für die Anzeige einer Linkliste reservieren usw. Wenn eine Stildefinition sprechen könnte, würde sie etwas sagen wie: „Hallo Browser, das soll so aussehen.“ Eine **Stildefinition** besteht immer aus zwei Elementen: einem Selektor, der angibt, welches Element oder welche Elemente einer Seite der Browser formatieren soll, und den Deklarationsblock, der die Formatierungsanweisungen enthält. Ein Selektor kann beispielsweise eine Überschrift, ein Absatz oder ein Bild usw. sein.

Ein Deklarationsblock enthält eine oder mehrere Deklarationen, die dafür sorgen,

dass z.B. die Überschrift in roter Schrift erscheint, der Absatz zentriert wird und das Bild mit einem blauen Rahmen umgeben wird – die Möglichkeiten sind schier endlos.

Hinweis: Technisch orientierte Menschen folgen gern den Vorgaben des **W3C** und bezeichnen CSS-Stildefinitionen als Regeln. In diesem Kapitel werden beide Begriffe synonym verwendet.

- Leider sind CSS-Stile nicht wirklich in der Lage, in Menschensprache mit den Browsern zu sprechen. Ihre Sprache folgt eigenen, einfacheren Regeln. Um für die Absätze einer Webseite beispielsweise eine Standardschriftart, -farbe und -größe festzulegen, könnten Sie Folgendes schreiben:

```
p {
color: red;
font-size: 1.Sem;
}
```

was so viel heißt wie: „Stelle den Text in sämtlichen Absätzen (die mit dem <p>-Tag markiert sind) rot und in einer Größe von 1,5 em-Einheiten dar.“ (Ein em ist eine sogenannte relative Größenangabe, die auf der Standardschriftgröße des Browsers basiert.) Wie in Abbildung 2.2 zu sehen ist. Es können selbst einfache Regeln wie diese aus mehreren Einzelteilen bestehen:

- **Selektor**. Wie bereits gesagt, teilt der Selektor dem Webbrowser mit, auf welches Element oder welche Elemente die Regeln angewendet werden sollen, beispielsweise auf Überschriften, Absätze, Bilder, Links usw. Der Selektor auf der linken Seite der Definition (p) in Abbildung 2.2 bezieht sich auf das <p> Tag. Hierdurch werden Webbrowser angewiesen, die CSS-Regeln auf alle <p> Tags

anzuwenden. Mit der großen Zahl an möglichen Selektoren und ein wenig Kreativität werden Sie die Formatierung Ihrer Seiten schnell in den Griff bekommen. (Wir werden die Selektoren im folgenden Kapitel ausführlich besprechen.)

- **Deklarationsblock**. Der auf den Selektor folgende Code enthält die Formatierungsanweisungen für den betreffenden Selektor. Ein Deklarationsblock beginnt immer mit einer öffnenden geschweiften Klammer (`{`) und endet mit einer schließenden geschweiften Klammer (`}`).
- **Deklaration**. Zwischen den geschweiften Klammern des Deklarationsblocks können Sie eine oder mehrere Deklarationen, die Formatierungsanweisungen, angeben. Hierbei besteht jede Deklaration aus einer Eigenschaft und einem oder mehreren Werten. Eigenschaft und Wert werden durch einen Doppelpunkt (`:`) voneinander getrennt. Eine Deklaration wird prinzipiell mit einem Semikolon (`;`) beendet.
- **Eigenschaft**. CSS besitzt eine Vielzahl von Formatierungsoptionen, den sogenannten Eigenschaften. Eine Eigenschaft besteht aus einem oder mehreren durch Bindestriche verbundenen Wörtern, die sich auf einen bestimmten Aspekt oder Effekt beziehen. Die meisten Eigenschaften besitzen mehr oder weniger selbsterklärende Namen wie `background-color` (Hintergrundfarbe) oder `font-size` (Schriftgröße). Mit der Eigenschaft `background-color` definieren Sie beispielsweise (wer hätte das gedacht?) die Hintergrundfarbe für eines oder mehrere Elemente. Im Laufe dieses Kapitels werden Sie Unmengen von CSS-Eigenschaften kennenlernen.
- **Wert**. Um Ihrem kreativen Talent angemessen Ausdruck verleihen zu können, können Sie den Eigenschaften bestimmte Werte zuweisen. Diese sorgen dafür, dass ein Hintergrund beispielsweise blau, rot, purpur oder gelblich-grün (wenn es unbedingt sein muss) dargestellt wird. Wie Sie in den folgenden Kapiteln sehen werden, haben verschiedene CSS-Eigenschaften bestimmte Arten von Werten, z.B. eine Farbe (wie `red` oder `#ff0000`), eine Längenangabe (z.B. `18px`, `2in` oder `5em`), eine URL (z.B. bilder/hintergrund.gif) oder ein bestimtm es Schlüsselwort (beispielsweise `top`, `center` oder `bottom`).

Hierbei muss die Stildefinition nicht unbedingt auf einer Zeile stehen, wie in Abbildung 2.2. Viele Stildefinitionen enthalten mehrere Eigenschaften. Sie können die Lesbarkeit Ihrer Regeln deutlich erhöhen, wenn Sie sie auf mehrere Zeilen verteilen. Hierbei ist es üblich, den Selelctor und die öffnende geschweifte Klammer auf die erste Zeile zu stellen und jede Deklaration sowie die schließende geschweifte Klammer in jeweils eine eigene Zeile zu schreiben, wie hier:

```
p {
color: red;
font-size: 1.5em;
}
```

Es ist außerdem hilfreich, die Eigenschaften mithilfe von Tabulatorschritten oder ein paar Leerzeichen etwas einzurücken, um die Selektoren und die Eigenschaften visuell leichter voneinander unterscheiden zu können. Das Leerzeichen zwischen dem Doppelpunkt und dem Eigenschaftswert ist optional, erhöht aber ebenfalls die Lesbarkeit. Sie können so viele Leerzeichen einfügen, wie Sie wollen; `color:red`, `color:red` und `color: red` funktionieren gleichermaßen.

Stylesheets verstehen

Natürlich verwandelt eine Stildefinition eine Webseite nicht gleich in ein Kunstwerk. Vielleicht wird die Schrift in Ihren Absätzen jetzt rot dargestellt, aber das macht Ihre Website nicht gleich zu einer Design-Ikone – dafür werden mehrere Stildefinitionen gebraucht. Eine Sammlung von **CSS-Regeln** wird als Stylesheet bezeichnet. Je nachdem, ob das Stylesheet in der HTML-Seite selbst oder in einer externern Datei gespeichert wird, spricht man von internen oder externen Stylesheets.

Intern oder extern – die richtige Wahl

In den meisten Fällen werden Sie externe Stylesheets verwenden wollen, weil sie das Erstellen von Webseiten vereinfachen und eine Aktualisierung mehrerer Seiten auf einmal wesentlich schneller geht. Bei externen Stylesheets können alle Informationen in einer Datei gespeichert werden. Mit nur einer Codezeile können Sie das externe Stylesheet in Ihre HTML-Seiten einbinden und dadurch das Aussehen komplett verändern. Mit nur wenigen Handgriffen an einer einfachen Textdatei, dem Stylesheet, können Sie das Erscheinungsbild einer ganzen Website anpassen.

Beim Benutzer hat das zur Folge, dass Webseiten schneller geöffnet werden. Bei der Verwendung externer Stylesheets enthält die Webseite nur die nötigen HTML-Informationen, aber keine kostspieligen Tabellenlayouts. `<font>`-Tags oder direkt in die Seite eingebettete Stilregeln. Hinzu kommt, dass Webbrowser externe Stylesheets auf dem Rechner des Benutzers zwischenspeichern (in einem sogenannten Cache). um schneller daraufzugreifen zu können. Wenn der Besucher andere Seiten Ihrer Website aufruft, die dasselbe externe Stylesheet verwenden, muss der Browser die bereits gespeicherten Informationen nicht erneut aus dem Netz laden. Er kann direkt auf die Version im lokalen Zwischenspeicher, dem **Cache**, zugreifen, was deutliche Geschwindigkeitsvorteile bringen kann.

Info

Wenn Sie an einer Website arbeiten und eine Vorschau im Browser betrachten wollen, kann der `cache` gelegentlich ein Hindernis bedeuten.

Interne Stylesheets

Ein internes Stylesheet ist eine Sammlung von Stildefinitionen, die direkt in den **HTML-Code** einer Seite eingebettet werden. Interne Stylesheets stehen prinzipiell innerhalb der `<head>`-Sektion und werden mit einem `<style>...</style>`-Tag-Paar umgeben. Hier ein Beispiel:

```
<style type="text/css">
h1 {
color: #FF7643;
font-family: Arial;
```

```
}
p {
color: red;
font-size: 1.5em
}
</style>
</head>
<body>
/* Hier steht der Rest Ihrer Seite...*/
```

Info

Zwar können die `<style>`-Tags an beliebiger Stelle im Kopfteil einer Seite stehen (z.B. vor dem `<title>`-Tag). Üblicherweise platzieren Webdesigner interne Stylesheets allerdings direkt vor dem schließenden `</head>`-Tag.

Das `<style>`-Tag selbst ist jedoch Teil des HTML-Codes und kein CSS. Seine Aufgabe besteht darin, dem Browser mitzuteilen, dass es sich bei den enthaltenen Informationen um CSS-Code und nicht um HTML handelt. Um ein internes Stylesheet zu erstellen, reicht es aus, ein paar Stilregeln zwischen die `<style>`-Tags zu schreiben, wie oben gezeigt.

> Interne Stylesheets lassen sich leicht in eine Webseite integrieren

und sorgen dadurch für eine sofortige visuelle Bereicherung Ihres HTML-Codes. Wollen Sie dagegen eine komplette Website mit einer großen Zahl Einzelseiten erstellen, ist diese Methode nicht besonders effizient. So müssen Sie den Code in jede einzelne Seite erneut einfügen – eine zeitraubende Aufgabe, die außerdem unnötig Bandbreite verbraucht.

Noch ärgerlicher wird die Arbeit mit internen Stylesheets, wenn Sie das Aussehen der gesamten Website aktualisieren wollen. Vielleicht sollen `<h1>`-Überschriften, die ursprünglich in großer, grüner fetter Schrift dargestellt wurden, jetzt mit kleineren Buchstaben, dafür aber in Blau und in der Schrift Courier erscheinen. Bei der Verwendung von internen Stylesheets müssen Sie jetzt jede einzelne Seite ändern. Wer hat dafür schon Zeit? Glücklicherweise gibt es für dieses Dilemma eine einfache Lösung: externe Stylesheets.

Externe Stylesheets

Ein externes Stylesheet ist im Prinzip einfach nur eine Textdatei, die Ihre CSS-Stildefmitionen enthält. HTML-Code ist hier nicht erlaubt. Der Code darf in externen Stylesheets also nicht mit `<style>`-Tags umgeben sein, und als Dateiendung muss `.css` verwendet werden. Der Dateiname selbst kann frei

gewählt werden, sollte aber möglichst beschreibend für den Inhalt sein. Der Name allgemein.css eignet sich beispielsweise gut für ein Stylesheet, das für alle Seiten einer Site benutzt werden soll, während formulare.css die Regeln für die Gestaltung von Webformularen enthalten könnte.

Info

Um ein internes Stylesheet „auszulagern“, also in externes Stylesheet umwandeln zu wollen, brauchen Sie nur die folgenden Schritte durchzuführen: Schneiden Sie den Code zwischen den `<style>`-Tags aus (ohne die Tags), legen Sie eine neue leere Textdatei an und fügen Sie den ausgeschnittenen Code in die neue Datei ein. Sichern Sie die Datei mit der Endung `.css` z.B. allgemein.css. Danach können Sie das neue externe Stylesheet mit einer der im Folgenden beschriebenen Techniken in die Webseite (und alle anderen!) einbinden.

Damit ein externes Stylesheet die Darstellung eines HTML-Dokuments verändern kann, muss es in die betreffende Seite eingebunden werden. Hierfür gibt es zwei Möglichkeiten: Entweder Sie verwenden das spezielle HTML-Tag `<link>`, oder Sie wählen die CSS-eigene `@import-Direktive` – einen speziellen Befehl, der im Prinzip das Gleiche tut wie das `<link>`-Tag. Alle modernen Webbrowser behandeln diese Techniken auf die gleiche Weise; mit beiden können Sie Stylesheets in HTML-Seiten einbinden. Welche Variante Sie verwenden, ist also (bis auf eine Ausnahme, ältere Browser können mit dieser Directive nicht umgehen) eine Frage Ihrer Vorlieben.

Info

Die `@import-Direktive` kann eine Sache, zu der das `<link>`-Tag nicht fähig ist: externe Stylesheets in externe Stylesheets einbinden.

Stylesheets mit HTML einbinden

Eine Möglichkeit, ein externes Stylesheet in eine Webseite einzubinden, besteht in der Verwendung des `<link>`-Tags. Je nachdem, ob Sie HTML oder XHTML benutzen, gibt es einen winzigen (aber wichtigen) Unterschied in der Schreibweise des Tags. Hier sehen Sie die HTML-Variante:

```
<link rel="stylesheet" type="text/css" href="css/allgemein.css">
```

Und hier ist der entsprechende XHTML-Code:

```
<link rel="stylesheet" type="text/css" href="css/allgemein.css" />
```

Der einzige Unterschied besteht darin, wie das Tag beendet wird. Das `<link>`-Tag ist ein „leeres“ Element. Es gibt also kein schließendes `</link>`-Tag. In XHTML müssen allein stehende Tags mit einem Schrägstrich beendet werden (wie hier: `/>`). In HTML ist das dagegen nicht nötig.

Davon abgesehen ist der Name in HTML und XHTML gleich. Unabhängig von der Sprache werden, wie in den oben stehenden Beispielen gezeigt, drei Attribute benötigt:

- **rel** gibt an, um welche Art von Link es sich handelt, in unserem Fall um einen Link zu einem Stylesheet.
- **type** sagt dem Browser, welche Art von Daten die eingebundene Datei enthält: eine Textdatei mit CSS-Code.
- **href** gibt an, wo die externe CSS-Datei zu finden ist. Der Wert dieses Attributs ist eine URL, die auf die gleiche Weise funktioniert wie beispielsweise das `src`-Attribut für Bilder oder `href` für das Anlegen von Links.

Info

Es ist möglich, mehrere verschiedene Stylesheets in eine Webseite einzubinden (z.B. allgemein.css und formulare.css, indem Sie mehr als ein `<link>`-Tag verwenden. Dies ist eine hervorragende Möglichkeit, Ihre CSS-Stile zu ordnen.

Stylesheets mit CSS einbinden

CSS bringt außerdem seine eigene Methode mit, externe Stylesheets einzubinden: die `@import-Direktive`. Diese steht im HTML-Code innerhalb der `<style>...</style>`-Tags, wie hier gezeigt:

```
<style type="text/css">
@import url(css/allgemein.css);
</style>
```

Im Gegensatz zum `<link>`-Tag ist `@import` nicht Teil von HTML, sondern von CSS und hat als solches einige HTML-untypische Eigenheiten:

- Der Verweis auf die externe CSS-Datei wird nicht mit dem Attribut `href`, sondern mit dem Schlüsselwort `url` eingeleitet. Der Pfad (oder die URL) wird mit runden Klammern umgeben. In unserem Beispiel ist demnach css/allgemein.css der Pfad zum externen Stylesheet. Die Anführungszeichen sind optional; Sowohl url(css/allgemein.css) als auch url(„css/allgemein.css“) sind gültig.

Info

Neben der Tatsache, dass Ihre HTML-Datei immerhin um 2 Byte größer wird, wenn Sie den pfad mit Anführungszeichen umgeben, hat auch der Internet Explorer 5 für Macintosh Probleme bei deren Interpretation. Wir haben Sie gewarnt.

- Wie mit dem `<link>`-Tag können Sie auch mit `@import` mehrere externe Stylesheets einbinden:

```
<style type="text/css">
@import url(css/allgemein.css);
@import url(css/formulare.css);
</style>
```

- Da sich die `@import-Direktive` innerhalb der `<style>`-Tags befindet, können Sie auch ganz normale CSS-Regeln nach der Direktive einbauen. Auf diese Weise können Sie die im externen Stylesheet definierten allgemeinen Stile um spezielle Regeln für eine bestimmte Seite ergänzen oder diese auch überschreiben.

Info

Es ist möglich Regeln zu erstellen, die andere Stildefinitionen überschreiben. Außerdem ist es möglich, eine externe Stylesheet Datei anzulegen, die nur `@import-Direktiven` enthalt, die andere externe Stylesheets einbinden. Diese Technik wird häufig verwendet, um Ordnung in größere Mengen externer Stylesheets zu bringen.

Hier ein Beispiel:

```
<style type="text/css">
@import url(css/allgemein.css);
@import url(css/formulare.css);
p { color: red; }
</style>
```

- Eigentlich sollten Sie die `@import-Direktiven` vor allen anderen CSS-Regeln platzieren, wie hier gezeigt. Allerdings ist es nicht so schlimm, wenn Sie das einmal vergessen. Nonnalerweise sollten Browser Stylesheets ignorieren, die nach einer CSS-Regel importiert werden, allerdings wird diese Einschränkung von sämtlichen aktuellen Browsern ignoriert.

Kinder erben von ihren Eltern bestimmte Merkmale wie Augenfarbe, Größe. Ob und wann Männer eine Glatze bekommen und vieles mehr. Manchmal erben wir auch bestimmte Eigenschaften von früheren Vorfahren, z.B. unseren Groß- oder sogar Urgroßeltern. Wie Sie im vorigen Kapitel gesehen haben, wird die Metapher der Familienbeziehungen auch in HTML verwendet. Auf ähnliche Weise können **HTML-Tags** die Werte von CSS-Eigenschaften von ihren Vorfahren erben.

Was ist Vererbung?

Kurz gesagt, bezeichnet Vererbung den Prozess, mit dem die Werte von **CSS-Eigenschaften** an

verschachtelte Tags weitergegeben werden. So ist das <p>-Tag immer von einem <body>-Tag umgeben. Bestimmte dem <body>-Tag zugewiesene Eigenschaften werden automatisch an das <p>-Tag vererbt. Stellen Sie sich vor, Sie haben in einer Stildefinition für <body> definiert, dass die Schriftfarbe (z.B. `color: #b32400;`) Dunkelrot sein soll. Nachfahren-Tags von <body>, also alle Tags, die sich innerhalb der <body>...</body>-Tags befinden, werden diese Einstellungen aufgrund der Vererbung übernehmen. Das heißt, der Text in diesen Tags, <h1>, <h2>, <p> oder wo auch immer, wird nun ebenfalls in dunkelroter Schrift angezeigt.

Die Vererbung funktioniert sogar über mehrere Generationen hinweg. Wenn sich innerhalb des <p>-Tags ein – oder -Tag befindet, werden diese ebenfalls die für das <body>-Tag festgelegten Eigenschaften erben.

Info:

Wie schon besprochen, gilt jedes Tag innerhalb eines anderen Tag-Paars als dessen Nachfahre. Ein <p>-Tag innerhalb eines <body>-Tags ist demnach ein Nachfahre von <body>, während das <body> der Vorfahre des <p>-Tags ist. Nachfahren (Kinder, Enkel usw.) erben Eigenschaften von ihren Vorfahren (Eltern, Großeltern usw.).

Obwohl es anfangs ein wenig konfus klingen mag, können Sie dennoch mit der Vererbung in **CSS** eine Menge Zeit sparen. Würden die Eigenschaftswerte nicht vererbt, müssten diese für alle enthaltenen (verschachtelten) Tags neu definiert werden. Passiert das nicht, müssten Sie beispielsweise Stile für Hervorhebungen () innerhalb eines Absatzes (<p>) oder Links (<a>) innerhalb einer Liste von Hand zuweisen, obwohl sie eigentlich die gleiche Schriftart und -farbe verwenden sollen – eine langwierige und fehlerträchtige Arbeit. Ohne die Anpassungen würden diese verschachtelten Tags im langweiligen Browserstandard dargestellt werden.

Dabei ist die Vererbung nicht auf Stile beschränkt, die mit dem Typ-Selektor (für Tags) festgelegt werden. Sie funktioniert mit allen Selektoren. Wenn Sie für ein Tag einen klassenbasierten Stil definieren, erben alle enthaltenen Tags die Eigenschaften. Das Gleiche gilt auch für ID-basierte Regeln. Selektoren für Nachkommen und die anderen besprochenen Möglichkeiten, Elemente einer Seite auszuwählen.

Wie die Vererbung Stylesheets vereinfachen kann

Sie können die Vererbung verwenden, um Ihre **Stylesheets** zu vereinfachen. Wenn Sie beispielsweise auf der gesamten Seite die gleiche Schrift verwenden wollen, müssen Sie diese nicht für jedes Tag einzeln definieren. Aufgrund der Vererbung reicht es aus, eine Stilregel für das <body>-Tag festzulegen. (Alternativ können Sie natürlich auch eine klassenbasierte Stildefinition verwenden und diese dem <body>-Tag zuweisen.) Geben Sie in dieser Regel an, welcher Schrifttyp benutzt werden soll, und er wird automatisch an alle in <body> enthaltenen Tags vererbt. Schneller und einfacher geht es kaum. z.B.:

```
body {
font-family: Arial, Helvetica, sans-serif;
}
```

Mithilfe der Vererbung können Sie aber auch Stile für bestimmte Seitenbereiche festlegen. Viele **Webdesigner** benutzen `<div>`-Tags, um bestimmte Bereiche eindeutig als Logo. Seitenleiste oder Fußzeile zu kennzeichnen. Stile, die Sie diesen `<div>`-Behältern zuweisen, werden an alle enthaltenen Tags weitergereicht, ohne dabei Tags in anderen Teilen der Seite zu beeinflussen. Soll etwa sämtlicher Text einer Seitenleiste die gleiche Farbe haben, erstellen Sie eine Stilregel mit der CSS-Eigenschaft `color`, die Sie dann dem `<div>`-Tag zuweisen. Sämtliche im `<div>` enthaltenen Tags, `<p>`, `<h2>` usw., erhalten jetzt die gleiche Schriftfarbe.

Einschränkungen der Vererbung

Allerdings ist auch die Vererbung nicht allmächtig. Viele CSS-Eigenschaften geben ihre Werte nicht an HTML-Nachkommen weiter. Zum Beispiel wird die Eigenschaft `border` (mit der Elemente mit einem Rahmen versehen können) aus gutem Grund nicht vererbt. Ansonsten würde jedes verschachtelte Tag, das sich in einem Tag mit definiertem Rahmen befindet, ebenfalls einen eigenen Rahmen erhalten. Hätten Sie beispielsweise das `<body>`-Tag mit einem Rahmen versehen, würde jede Liste und auch jedes einzelne Listenelement einen Rahmen erhalten.

Hier ein paar Beispiele für Situationen, in denen die Vererbung nicht strikt angewandt wird:

- Allgemein gilt, dass Eigenschaften. die die Platzierung von Elementen auf der Seite beeinflussen. z.B. Außenabstände (`margin`), Hintergrundfarben und Rahmen, nicht vererbt werden.
- Webbrowser verwenden eigene Stylesheets, um den einzelnen Tags eine Standardformatierung zu geben: Überschriften werden groß und fett dargestellt, Links sind blau usw. Wenn Sie anhand eines Stils für das `<body>`-Tag eine Schriftgröße für die Seite festlegen, werden Überschriften auch weiterhin größer als Absätze angezeigt, und `<h1>`-Überschriften sind immer noch größer als `<h2>`-Überschriften. Das Gleiche gilt bei der Definition einer Schriftfarbe für `<body>`. Auch in diesem Fall werden die Links im guten alten Webbrowser-Blau angezeigt.
- Wurden zwei sich widersprechende Stile für das gleiche Tag definiert, gewinnt die genauer formulierte Regel. (Die Experten sprechen hier von „Spezifität“. Beispielsweise kann es sein, dass Sie für eine Liste (`<ul>`) eine bestimmte Schriftgröße festlegen, die mit vererbten Eigenschaften kollidiert – etwa mit einer für das `<body>`-Tag definierten Größe. In diesem Fall verwendet der Browser den speziell für `<ul>` definierten Wert.

Info:

Diese Konflikte treten sehr häufig auf. Die Regeln, nach denen ein Browser diese Konflikte auflöst, werden als Kaskade (engl. cascode) bezeichnet, von der die Cascading StyleSheets übrigens ihren Namen haben.

Bevor wir uns weiteren CSS-Eigenschaften widmen, ist ein wichtiger Theorieabschnitt notwendig. Nur wenn

Sie diese Grundlagen verstehen, können Sie souverän mit den anschließend beschriebenen CSS-Eigenschaften umgehen.

Das Boxmodell von CSS beschreibt den Erstreckungsraum und das Anordnungsverhalten von Block- und Inline-Elementen in HTML. Jedes Element wird dabei als rechteckige Box betrachtet.

Die Boxen der Basiselemente

Es beginnt beim `html`-Element und beim `body`-Element. Diese beiden stellen bereits eigene Boxen mit unterschiedlichen Eigenschaften dar. Ein Test soll das verdeutlichen. Das folgende Listing zeigt ein vollständiges HTML-Dokument, bei dem das `html`– und das `body`-Element mithilfe von CSS je einen Rahmen erhalten:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html style="border:solid 4px blue">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>Boxmodell</title>
</head>
<body style="border:solid 4px red">
<h1>Some body</h1>
<p>Some thing</p>
</body>
</html>
```

Info

Im <`html`>-Tag wird ein 4 Pixel dicker blauer Rahmen definiert und im <`body`>-Tag ein ebenso dicker roter Rahmen. Das Ergebnis im Browser ist in mehrfacher Hinsicht aufschlussreich:

Gut erkennbar ist, dass das `html`-Element eine andere Box darstellt als das `body`-Element. Das `html`-Element hat kein anderes Elternelement. Sein Bezugspunkt ist daher einfach das Browserfenster, im CSS-Fachjargon als **Viewport** bezeichnet. Der blaue Rahmen des `html`-Elements erstreckt sich, wie die Abbildung zeigt, direkt bis zu den Fensterrändern.

> Das `body`-Element ist dagegen ein Kindelement des `html`-Elements.

Seine Box hat ganz andere Eigenschaften. Zunächst fällt auf, dass es einen gewissen Abstand zur Box des

html-Elements hat, obwohl mit CSS kein solcher Abstand definiert wurde. Der Abstand zwischen dem blauen und dem roten Rahmen ist einfach ein Default-Abstand, den der Browser benutzt, damit der Inhalt eines HTML-Dokuments nicht direkt am Fensterrand klebt.

Weiter fällt auf, dass die Box des body-Elements in der Breite so viel Raum einnimmt wie verfügbar ist, obwohl der Inhalt an keiner Stelle die volle Breite erreicht. Dagegen nimmt die Box in der Höhe so viel Raum ein, wie es der Inhalt erfordert. Genau dies sind die typischen Verhaltensweisen für Block-Elemente: so breit wie möglich, so hoch wie nötig.

Die Boxen von Block-Elementen

Im zweiten Schritt versehen wir die beiden Textelemente des Beispiels ebenfalls mit Rahmen:

```
<h1 style="border:solid 4px lime">Some body</h1>
<p style="border:solid 4px violet">Some thing</p>
```

Info

Erkennbar ist, dass auch die beiden Block-Elemente h1 und p dem Gesetz folgen: so breit wie möglich, so hoch wie nötig. Ebenfalls erkennbar sind jedoch auch diverse Default-Werte, die der Browser zur Darstellung der Elemente verwendet. So gibt es Abstände zwischen den Rahmen der Überschrift und des Textabsatzes, über der Überschrift und unter dem Textabsatz, die nirgendwo in CSS definiert wurden. Es handelt sich hierbei um Default-Abstände, die der Browser bei der Darstellung eines HTML-Dokuments verwendet.

> Ebenso erkennbar ist, dass der Text der h1-Überschrift

zwar links, nicht aber oben und unten die Ränder der Box der Überschrift berührt. Auch das liegt an einer Default-Einstellung des Browsers: Er benutzt einen intern eingestellten Innenabstand zwischen Elementgrenze und Elementinhalt. Besonders bei Elementen wie Tabellenzellen ist dies sinnvoll. Würde der Browser dort keinen Innenabstand darstellen, so würden die Zelleninhalte direkt am Gitternetz der Tabelle kleben.

Die Boxen von Inline-Elementen

Betrachten wir nun noch, wie sich typische Inline-Elemente im Boxmodell verhalten. Dazu zeichnen wir ein Wort in der Überschrift des Beispiels separat mit dem em-Element aus und verpassen diesem mit CSS einen orangefarbenen Rahmen:

```
<h1 style="border:solid 4px lime">Some <em style=" border:solid 4px
orange">body</em></h1>
```

Info

Im Browser wird die zusätzliche Box angezeigt:

Für Inline-Elemente gilt wie gut erkennbar die Grundregel: so breit wie nötig und so hoch wie nötig.

Boxbestandteile

Es war bereits die Rede von Abständen zu Elternelementen, von Abständen zwischen Elementen und von Innenabständen. Zum Boxmodell gehört auch die Beschreibung, wie diese Bestandteile zusammenwirken. Eine Grafik veranschaulicht dies:

Die innerste Schicht einer Element-Box ist der **Elementinhalt**. Das kann – je nach Element und Gegebenheiten – reiner Text sein oder ein bzw. mehrere andere Elemente oder eine Mischung aus reinem Text und anderen Elementen.

Die nächste Schicht ist die des **Innenabstands**. Der Innenabstand ist der Abstand zwischen dem Elementinhalt und dem **Rahmen** des Elements. Dieser bildet die nächste Schicht. Hat ein Element keinen definierten Rahmen, d.h., beträgt die Rahmendicke 0, dann ist die Außengrenze des Rahmens gleich der Außengrenze des Innenabstands.

Die äußerste Schicht bildet der Rand des Elements,

d.h. der Abstand zu umgebenden Elementen oder den Elternelementen. Hat ein Element keinen definierten Abstand, d.h., beträgt der Randabstand zu den umgebenden Elementen 0, dann ist die Außengrenze des Rands gleich der Außengrenze des Rahmens und, falls dieser ebenfalls 0 beträgt, gleich der Außengrenze des Innenabstands.

Wenn Sie beispielsweise für ein Element mithilfe von CSS eine Breite von 500 Pixel angeben und außerdem einen Innenabstand des Elementinhalts von 10 Pixel, einen Rahmen von 5 Pixel und einen Außenabstand (Rand) von 30 Pixel bestimmen, so sollte das sichtbare Element insgesamt eine Breite von 500 + (2 * 10) + (2 * 5) Pixel, also 530 Pixel einnehmen. Der Rand geht in die Rechnung nicht mit ein, da er nicht die sichtbare Breite des Elements beeinflusst, sondern nur eine „Lücke“ zu umgebenden

Elementen darstellt.

An einem Quelltextbeispiel und dessen Auswirkung im Browser wollen wir den Zusammenhang verdeutlichen:

```
<div style="width:500px; padding:10px; border:solid black 5px; background-
color:rgb(255,255,128)">
<div style="font-size:32px; background-color:rgb(128,128,255);
color:white">Elementinhalt</div>
</div>
<div style="width:530px; background-color:rgb(255,192,255); font-size:16px;
font-weight:bold">530px</div>
<div style="width:500px; padding:10px; border:solid black 5px; background-
color:rgb(255,255,128); margin:30px">
<div style="font-size:2px; background-color:rgb(128,128,255);
color:white">Elementinhalt</div>
</div>
```

Info

Im Beispiel wird mit Bereichen, also `div`-Elementen gearbeitet. Diese haben den Vorteil, dass sie vom Browser für Außenabstand (Rand), Rahmen und Innenabstand die Default-Werte 0 erhalten und sich damit anzeigeneutral verhalten. Das Beispiel definiert zunächst einen `div`-Bereich mit einer Breite von 500 Pixel (width:500px), einem Innenabstand von 10 Pixel (`padding:10px`) und einem schwarzen durchgezogenen Rahmen von 5 Pixel (`border:solid black 5px`). Innerhalb davon wird ein weiterer `div`-Bereich notiert. Dieser stellt den Elementinhalt des zuvor definierten Bereichs dar. Beide `div`-Bereiche erhalten unterschiedliche Hintergrundfarben (`background-color`), damit sie sich optisch gut unterscheiden lassen.

> Als Nächstes folgt ein Kontrollbereich.

Bei diesem Bereich bleiben alle Werte für Innenabstand, Rahmen und Rand auf dem Default-Wert 0. Der Kontrollbereich bekommt jedoch eine Breite von 530 Pixel (`width:530px`) sowie eine sichtbare Hintergrundfarbe zugewiesen. Damit dient er zur Verifizierung der Berechnung, dass sich beim davor notierten Bereich eine sichtbare Elementbreite von 530 errechnet.

Zuletzt wird noch mal das gleiche `div`-Konstrukt wie zu Beginn definiert, diesmal jedoch zusätzlich mit 30 Pixel Rand an allen vier Seiten (`margin:30px`). Der folgende Screenshot zeigt die Wirkung im Browser.

Die Abbildung zeigt, dass unsere Rechnung korrekt war: Obwohl als Breite für den ersten `div`-Bereich 500

Pixel gewählt wurden, ergibt sich eine Gesamtbreite von 530 Pixel, da die Angaben zu Innenabstand und Rahmen in die Berechnung mit einfließen. Der Kontrollbereich mit dem Textinhalt „530px“, der eine Eigenlänge von 530 Pixel hat, bestätigt dies. Der untere `div`-Bereich schließlich, der eine Kopie des obersten mit zusätzlich definiertem Rand von 30 Pixel darstellt, ist um 30 Pixel zum Inhalt oberhalb und 30 Pixel von links entfernt. Der Rand ist einfach eine Lücke zwischen den Elementen.

> Das Boxmodell, wie es in dieser Form spezifiziert ist, hat Vor- und Nachteile.

Vorteile hat es insofern, als eine angegebene Breite nicht mit zusätzlichen Angaben zu Innenabstand und Rahmen kollidiert. D.h. die angegebene Breite steht dem Elementinhalt auf jeden Fall zur Verfügung und die zusätzlichen Angaben führen einfach zu einer faktischen Verbreiterung des Elements. Zum Ärgernis wird dieses Verhalten jedoch, wenn mit einer Angabe wie `width:100%` gearbeitet wird, um z.B. das Browserfenster exakt in der Breite auszufüllen. Soll ein solches Element dann noch einen Innenabstand und/oder Rahmen erhalten, ergibt sich eine tatsächliche Breite, die größer ist als 100%. Der Browser zeigt Quer-Scrollbalken an und die Breite des Inhalts ist etwas breiter als das Anzeigefenster, was nicht sehr elegant ist. Nur mithilfe von Hacks (Tricks) lässt sich dieses Problem umgehen.

Zuletzt soll noch eine Ausnahme beim Boxmodell angesprochen werden: das `body`-Element. In Abbildung unter „Die Boxen der Basiselemente“ haben wir die Ausdehnung des `body`-Elements visualisiert. Dabei hat sich gezeigt, dass das `body`-Element so viel Höhe einnimmt, wie es sein Inhalt erfordert, und dass die Browser Default-Abstände zwischen Fensterrand und `body`-Element einfügen. Dennoch wird, wenn man in CSS eine Hintergrundfarbe (`background-color`) für das `body`-Element definiert, das gesamte Browserfenster eingefärbt. Diese Ausnahme stellt ein Entgegenkommen der Theorie gegenüber der Praxis dar. Denn es musste einfach eine Handhabung her, um eine Seite mit einem Hintergrund zu versehen.

Rahmen und Innenabstände

Um einen einheitlichen Rahmen für alle vier Seiten eines Elements zu definieren, können Sie mit der CSS-Eigenschaft `border` arbeiten. Es ist aber auch möglich, Rahmen nur für einzelne Seiten zu definieren oder unterschiedlich aussehende Rahmen für einzelne Seiten. Dazu stehen die Eigenschaften `border-left` (Rahmen links), `border-right` (Rahmen rechts), `border-top` (Rahmen oben) und `border-bottom` (Rahmen unten) zur Verfügung.

Alle fünf Eigenschaften stellen Zusammenfassungen von einzelnen untergeordneten Eigenschaften zu Dicke, Farbe und Typ des Rahmens dar. Deshalb sind bei der Wertzuweisung bis zu drei Angaben möglich, die durch Leerzeichen zu trennen sind.

```
<div style="border:1px solid red">
...
</div>
<div style="border-top:double rgb(192,192,255) 6px; border-bottom:solid
rgb(255,192,192)">
```

```
...
</div>
```

Info

Im Beispiel werden zwei `div`-Bereiche definiert. Der erste Bereich erhält einen einfach durchgezogenen Rundumrahmen von 1 Pixel Breite (`border:1px solid red`). Beim zweiten Bereich bleiben die Elementseiten links und rechts rahmenlos. Für oben wird eine doppelte hellblaue Linie mit einer Gesamtbreite von 6 Pixel bestimmt (`border-top:double rgb(192,192,255) 6px`) und für unten eine einfache rosafarbene Linie ohne definierte Breite (`border-bottom:solid rgb(255,192,192)`).

Dem Beispiel ist erstens zu entnehmen, dass die Reihenfolge der Angaben bei der Wertzuweisung egal ist – `border:1px solid red` ist also genauso erlaubt wie `border:red solid 1px`. Zweitens geht aus ihm hervor, dass nicht alle drei Angaben zwingend sind. Fehlt eine Angabe, so verwendet der Browser einen Default-Wert.

> Ein Rahmen wird vom Betrachter eigentlich nur als solcher empfunden,

wenn er rund um das gesamte Element gezogen wird. Werden dagegen nur einzelne Rahmenangaben für links, rechts, oben oder unten definiert, entsteht optisch eher der Eindruck von Linien. Genau das ist aber ein interessantes Stilmittel zur Auflockerung.

Damit Rahmen bzw. Linien richtig zur Geltung kommen, sind in der Praxis meist zusätzliche CSS-Angaben erforderlich. Vor allem der Innenabstand ist von Bedeutung, um zwischen Elementinhalt und Rahmen einen als angenehm empfundenen Raum zu schaffen. Anhand eines Beispiels wollen wir zeigen, wie das Zusammenspiel von CSS-Angaben zu Rahmenteilen, Innenabständen, Elementabständen und Einrückungen zu ansprechenden Ergebnissen beim Textlayout führen kann.

Der HTML-Quelltext zur Abbildung lautet:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>Rahmen</title>
</head>
<body style="margin-left:30px; margin-right:30px;">
<h1 style="border-bottom:double green 6px; padding-bottom:12px;
color:green">Blindtexte</h1>
```

```
<p>Ein Blindtext dient keinem anderen Zweck als dem, Beispieltext zu erzeugen.
Nachfolgend einige Blindtexte</p>
<h2 style="border-top:solid teal 2px; border-left:solid teal 2px; padding-
left:10px; padding-top:4px; color:teal; margin-top:32px;">Mord am Wort</h2>
<p style="margin-left:12px;">Um mich herum ein Feuerwerk vermeintlicher
Wichtigkeiten.... </p>
<h2 style="border-top:solid teal 2px; border-left:solid teal 2px; padding-
left:10px; padding-top:4px; color:teal; margin-top:32px;">Produkt-Aussage</h2>
<p style="margin-left:12px;">Ja, wir finden auch, dass man über die Copy noch
mal reden sollte. Das hier kann es jedenfalls nicht sein. Das klingt ja wie auf
dem Totenbett getextet... </p>
</body>
</html>
```

Info

Die Überschriften erhalten in diesem Beispiel Schmucklinien zur optischen Auflockerung. Die h1-Überschrift erhält mit `border-bottom` unterhalb eine grüne und 6 Pixel breite Doppellinie. Die beiden h2-Überschriften erhalten jeweils oben und links (`border-top und border-left`) eine einfache, 2 Pixel breite Linie im Farbton „teal“.

> Damit die Überschriftentexte nicht zu nah an den Linien kleben,

werden jeweils Innenabstände notiert. Die h1-Überschrift erhält mit `padding-bottom` einen Innenabstand unten von 12 Pixel. Da diese Überschrift mit `border-bottom` eine Linie unten erhält, entsteht ein Abstand von 12 Pixel zwischen den tiefsten Unterlängen des Überschriftentextes und der Linie. Die h2-Überschriften erhalten analog zu den Definitionen für `border-left` und `border-top` geeignete Innenabstände mit `padding-left` (Innenabstand links) und `padding-top` (Innenabstand oben).

Die Überschriften bekommen im Beispiel ferner die gleiche Textfarbe (`color`) zugewiesen wie ihre Rahmen. Das unterstreicht optisch den Zusammenhang zwischen Überschriften und Linien.

Eine weitere Maßnahme zur optischen Auflockerung ist die Einrückung der Absätze mit den Blindtexten. Diese werden mit `margin-left:12px` so eingerückt, dass der Textanfang genau an der gleichen Linksposition beginnt wie der Textanfang der zugehörigen h2-Überschriften. Bei den h2-Überschriften errechnet sich die Linksposition des Textanfangs aus den Werten für `padding-left` + `border-left`, also 10 + 2 = 12 Pixel.

Damit die gesamte Seite noch ansprechender wirkt, wurden dem body-Element etwas großzügigere linke und rechte Ränder (jeweils 25 Pixel mit `margin-left` und `margin-right`) spendiert.

Die Wertzuweisungen an die Eigenschaften `border`, `border-left`, `border-top`, `border-bottom` und `border-right` setzen sich wie bereits erwähnt aus den drei Untereigenschaften für Dicke, Typ und Farbe des Rahmens zusammen. Diese Untereigenschaften sind auch mit separaten CSS-Eigenschaften ansprechbar: `border-width` (Rahmendicke), `border-style` (Rahmentyp) und `border-color` (Rahmenfarbe). In Kombination mit den vier Seiten ergeben sich dadurch auch Eigenschaften wie `border-left-width`, `border-top-style` oder `border-bottom-color`.

> Beim Rahmentyp bietet CSS eine Reihe von vordefinierten Typen an.

So bewirkt die Wertzuweisung `solid` einen durchgezogenen, einfachen Rahmen und `double` einen doppelten Rahmen. Weitere Typen lassen sich durch die Werte `dotted` (gepunktet) und `dashed` (gestrichelt) sowie durch Angaben zu 3D-Effekten wie `groove`, `ridge`, `inset` und `outset` erzwingen. Für den Effekt spielt die Rahmendicke natürlich eine entscheidende Rolle, und vor allem bei den 3D-Effektrahmen auch die Rahmenfarbe.

Hintergrundgestaltung

Jedes im Browser-Fenster sichtbare HTML-Element kann mit Hilfe von CSS eine eigene Hintergrundgestaltung erhalten – sowohl Block- als auch Inline-Elemente. Möglich sind dabei eine Hintergrundfarbe oder eine Hintergrundgrafik. Bei Hintergrundgrafiken kann es sich um ein so genanntes **Wallpaper** handeln, also ein durch Wiederholung entstehendes Muster, oder auch um eine wunschgemäß positionierte Einzelgrafik.

Die CSS-Eigenschaft `background-color` zur Definition einer Hintergrundfarbe haben wir bereits kennen gelernt. Erlaubt ist eine Farbangabe wie in Farbangaben in CSS beschrieben. Achten Sie bei Verwendung von Hintergrundfarben stets auf geeignete, d.h. gut kontrastierende Vordergrundfarben beim Text.

> Farben machen Webseiten attraktiv.

Aber man kann auch viel verkehrt machen im Umgang damit. Und noch mehr verkehrt machen kann man im Umgang mit Hintergrundgrafiken. In den Frühzeiten der grafischen Browser kamen die Wallpapers für Webseiten in Mode. Grauenhaft grobe Muster erwarteten den Besucher aller-ortens, auf denen der Text kaum noch lesbar war. Heute stößt man zwar nur noch selten auf solche Seiten, doch gerade Einsteiger lassen sich leicht dazu verleiten, grafisch zu dick aufzutragen.

Unser erstes Beispiel zum Einsatz von Hintergrundgrafiken zeigt, wie ein Wallpaper sinnvoll einsetzbar ist. Ein gern genutzter und ordentlich aussehender Effekt auf Webseiten ist das optische Hervorheben einer Navigationsleiste, indem diese über dem Seiteninhalt zu schweben scheint, da sie einen Schatten auf die Seite wirft.

Um den gewünschten Effekt zu erzielen, muss die Grafik den Style-Eigenschaften des `body`-Elements

zugewiesen werden, und zwar so, dass sie senkrecht untereinander wiederholt wird, nicht aber waagerecht:

```
<body style="background-image:url(wallpaper.gif); background-repeat:repeat-y;">
```

Info

Durch `background-image` wird eine Hintergrundgrafik für das betroffene Element eingebunden. Die Wertzuweisung hat das Schema url (URI), wobei URI eine beliebige absolute Internetadresse einer Grafik oder eine lokal referenzierte Grafik sein kann. Im obigen Beispiel liegt die Grafikdatei, also **wallpaper.gif**, im gleichen Verzeichnis wie die HTML-Datei, in der sie eingebunden wird. Die Grafik hat übrigens eine Größe von nur 1,7 Kbyte Größe, ist also selbst bei schwachen Internetverbindungen schnell übertragen.

> Die Art, wie die Hintergrundgrafik wiederholt werden soll,

wird durch die CSS-Eigenschaft `background-repeat` festgelegt. Der Wert `repeat-y` bewirkt, dass die Grafik nur „eine Spalte lang“ wiederholt wird, waagerecht dagegen nicht. Falls Sie eine Grafik nur oben „eine Zeile lang“ wiederholen möchten, geben Sie `repeat-x` an. Mit `no-repeat` wird die Hintergrundgrafik nur einmalig angezeigt – ein Beispiel dazu folgt weiter unten. Wenn Sie gar nichts zu `background-repeat` angeben, wird die Voreinstellung `repeat` verwendet. Dabei wird die Hintergrundgrafik sowohl waagerecht als auch senkrecht so oft wiederholt, wie sie in die Ausdehnung des Elements hineinpasst (Tapetenmustereffekt).

Im Browser wird durch unser Beispiel folgender Effekt erzielt:

Der entstandene optische Bereich lässt sich hervorragend für eine Navigationsleiste verwenden. Der entsprechende HTML-Bereich dafür sollte jedoch genau positioniert werden.

Ein zweites Beispiel zum Einsatz von Hintergrundgrafiken zeigt, wie Sie den Seiteninhalt bewusst über die Hintergrundgrafik legen können. Die Hintergrundgrafik ist im Beispiel ein Bild der Sonne als Feuerball. An allen Rändern ist die Grafik schwarz. Dadurch eignet sie sich zur Kombination mit einem schwarzen Seitenhintergrund.

Diese Hintergrundgrafik binden wir wieder ins <body>-Tag ein:

```
<body style="background-image:url(sonne.jpg); background-repeat:no-repeat;
background-color:black">
```

Info

Die Hintergrundgrafik wird mit `background-image` eingebunden. Dass sie nur einmal angezeigt wird, wird durch `background-repeat:no-repeat` sichergestellt. Den Seitenhintergrund setzen Sie durch `background-color:black` auf Schwarz. Auf diesem Hintergrund platzieren wir nun eine passende, effektvolle Überschrift mit folgender Formatierung:

```
<h1 style="margin-top:100px; margin-left:130px; margin-right:40px; font-
size:36px; color:white; border-top:rgb(239,82,0) dotted 4px; border-
bottom:rgb(239,82,0) dotted 4px">Die Sonne - Licht und Leben</h1>
```

Info

Die Überschrift wird mithilfe von `margin-top` (Rand oben) und `margin-left` (Rand links) so positioniert, dass sie über der Sonnenmitte beginnt. Weiterhin erhält die Überschrift eine weiße Textfarbe (`color`), eine auffällige Schriftgröße von 36 Pixel (`font-size`) und zwei 4 Pixel dicke, gepunktete Rahmenlinien oberhalb und unterhalb (`border-top` bzw. `border-bottom`).

> Wenn eine Hintergrundgrafik nur einmalig angezeigt wird,

wie im Beispiel, so bedeutet dies: links oben an Position 0,0, gemessen von den Elementgrenzen. Mitunter ist es jedoch sinnvoll, die genaue Position zu beeinflussen. Dazu dient die CSS-Eigenschaft `background-position`. Durch eine Angabe wie `background-position:30px 20px` erzwingen Sie eine Verschiebung um 30 Pixel zur linken Elementgrenze und 20 Pixel zur oberen Elementgrenze. Ferner stehen die relativen Wertzuweisungen `top`, `left`, `bottom`, `right` und `center` zur Verfügung. Je zwei sinnvolle Kombinationen davon ergänzen sich. So bewirkt die Angabe `background-position:top center` eine an der Elementbreite gemessen zentrierte und gemessen an der Elementhöhe obenbündige Ausrichtung der Hintergrundgrafik. Die Angabe `background-position:bottom right` richtet die Hintergrundgrafik rechts unten an den Elementgrenzen aus und `background-position:center center` positioniert die Hintergrundgrafik sowohl horizontal als auch vertikal mittig in der Elementbox. Im letzteren Fall würde übrigens auch eine einfache Angabe von `center` genügen. Wenn `background-position` nur ein Wert zugeordnet wird, gilt dieser sowohl für „von links“ als auch für „von oben“.

Ein drittes Beispiel soll einen weiteren, in der Praxis häufig genutzten Effekt demonstrieren: ein Logo oder Schriftzug, der beim Scrollen von längeren Inhalten fix an seiner Stelle stehen bleibt. Früher wurden häufig Frames verwendet, nur um diesen Effekt zu erreichen. Glücklicherweise ist dies heute nicht mehr nötig.

Ein Komplett-Listing zeigt die weitere Vorgehensweise:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>Hintergrundbild</title>
</head>
<body style="background-image:url(logo.jpg); background-repeat:no-repeat;
background-attachment:fixed; background-position:10px 10px; background-
color:rgb(251,240,204);">
<div id="Seiteninhalt" style="margin-left:105px; margin-right:25px; margin-
top:25px;">
<h1 style="font-size:48px; font-weight:normal; color:rgb(128,0,0); font-
family:'Brush Script MT',cursive">Herbstkollektion 2005</h1>
</div>
</body>
</html>
```

Info

Wie auch in den Beispielen zuvor wird die Hintergrundgrafik mit `background-image` ins body-Element eingebunden. Der Effekt, dass die Grafik beim Scrollen in längeren Seiteninhalten fix an ihrer Position stehen bleibt, wird durch die zusätzliche Angabe `background-attachment:fixed` erreicht. Das Normalverhalten, also Scrollen mit dem Seiteninhalt, kann durch `background-attachment:scroll` erzwungen werden.

In unserem Beispiel haben wir die Grafik durch

`background-position:10px 10px` etwas vom linken oberen Rand abgesetzt. Ferner definierten wir mit

`background-color` eine Hintergrundfarbe für die Seite, die exakt der hellsten Farbe am unteren Rand der Grafik entspricht. So entsteht ein fließender Übergang, der gar nicht mehr erkennen lässt, wo genau die Grafik ihre untere Grenze hat.

Der Hintergrund eines HTML-Elements lässt sich also über Angaben zu den Einzeleigenschaften `background-color`, `background-image`, `background-position`, `background-repeat` und `background-attachment` steuern. All diese Angaben können jedoch auch in der Sammeleigenschaft `background` notiert werden.

```
<div style="background:url("background.png") gray 50% no-repeat fixed">...</div>
```

Info

In diesem Beispiel wird für einen `div`-Bereich eine dunkelgraue (`gray`) Hintergrundfarbe festgelegt. Als Hintergrundgrafik soll die Datei **background.png** angezeigt werden. Die Position der Hintergrundfarbe wird mit 50% angegeben. Das bedeutet: sowohl horizontal als auch vertikal zentriert, gemessen an den Ausmaßen der Elementbox. Die Hintergrundgrafik soll nur einmalig angezeigt werden (`no-repeat`) und sie soll nicht mitscrollen (`fixed`). Da solche Sammelangaben jedoch leicht unübersichtlich wirken, sind die Einzelangaben zumindest in Hinblick auf die Lesbarkeit empfehlenswerter.

CSS-Eigenschaften für Listen und Tabellen

Listen und Tabellen haben einige spezifische Eigenschaften. So gehört zu den Variablen einer Aufzählungsliste das Bullet-Symbol oder die Art der Einrückung und bei nummerierten Listen die Art der Nummerierung. Bei Tabellen sind es spezielle Eigenschaften für Rahmen und Gitternetz oder die Art der Anzeige von leeren Zellen. Darüber hinaus haben aber auch anderweitig anwendbare Eigenschaften in Zusammenhang etwa mit Tabellen besondere Bedeutung, etwa Eigenschaften für Abstände oder horizontale und vertikale Ausrichtung.

Eigenschaften von Listen

Um das Aufzählungszeichen (Bullet-Symbol) einer Aufzählungsliste (`ul`-Element) festzulegen, stellt CSS die Eigenschaft `list-style-type` zur Verfügung. Bei nummerierten Listen (`ol`-Element) legt die gleiche Eigenschaft die Art der Nummerierung fest.

Beispiel für eine Aufzählungsliste:

```
<ul style="list-style-type:disc">
```

Beispiel für eine nummerierte Liste:

```
<ol style="list-style-type:upper-roman">
```

Info

Für Aufzählungslisten sind die Wertzuweisungen `disc` (ausgefüllter Kreis ?), `circle` (nur umrandeter Kreis ?), `square` (ausgefülltes Quadrat ¦) und `none` (gar kein Aufzählungszeichen) erlaubt.

Die Nummerierungsart bei nummerierten Listen können Sie durch folgende Werte beeinflussen: `lower-roman` (i, ii, iii, iv usw.), `upper-roman` (I, II, III, IV usw.), `lower-alpha` (a, b, c, d usw.), `upper-alpha` (A, B, C, D usw.), `lower-latin` (a, b, c, d usw., jedoch auch mit Zeichen wie Umlauten, Accent-Buchstaben), `upper-latin` (A, B, C, D usw., inklusive Zeichen wie Umlauten, Accent-Buchstaben). Auch eine Alphanummerierung für andere Schriftkulturen kann definiert werden, durch `lower-greek` (Kleinbuchstaben des griechischen Alphabets), georgian, armenian, hebrew, hiragana, katakana, hiragana-iroha, katakana-iroha.

Grafische Browser stellen Aufzählungslisten und nummerierte Listen normalerweise so dar,

dass das Aufzählungs- oder Nummerierungszeichen ausgerückt gegenüber mehrzeiligen Listenpunktinhalten dargestellt wird. Dieses Verhalten lässt sich durch Angabe von `list-style-position:inside` so ändern, dass die zweite und weitere Zeilen eines Listenpunkts unterhalb des Aufzählungspunkts oder der Nummerierung erscheinen.

Keine Lösung bietet CSS leider für das Default-Verhalten der Browser, Listen selbst eingerückt gegenüber dem normalen Fließtext darzustellen. Je nach Browser lässt sich das Verhalten zwar durch Angaben zu `margin-left` und `padding-left` beeinflussen, aber eine saubere Allgemeinlösung gibt es nicht. Das Gleiche gilt für den Abstand zwischen Aufzählungszeichen bzw. Nummerierung und Listenpunktinhalt.

Wenn Ihnen dagegen die magere Auswahl der bei `list-style-type` angebotenen Bullet-Symbole für Aufzählungslisten nicht zusagt, können Sie auch beliebige Grafiken als Bullet-Symbol verwenden. Dazu dient die Eigenschaft `list-style-image`.

```
<ul style="list-style-image:url(blaues_dreieck.gif)">
```

Info

Ähnlich wie bei `background-image` wird die gewünschte Grafik über das Schema `url` (URI) zugewiesen. URI kann eine beliebige Internetadresse oder eine lokal referenzierte Adresse sein. Im Beispiel befindet sich die Datei **blaues_dreieck.gif** im gleichen Verzeichnis. Die Grafik sollte eine passende Größe haben – die typische Icon-Größe von 16 × 16 Pixel ist eine praxisnahe Orientierungsgröße.

Rahmentyp bei Tabellen

Sichtbare Rahmen und Gitternetze von Tabellen in HTML werden von den meisten grafischen Browsern, wenn nichts anderes angegeben ist, so angezeigt, dass jede Zelle eine Umrahmung erhält. Dazu kommt der Außenrahmen. Webbrowser bilden diesen Sachverhalt so ab, dass Rahmen und Gitternetz als Doppellinien erscheinen oder in 3D-Form mit Licht- und Schattenseite. Beide Varianten wirken meist klobig. Durch die Angabe `border-collapse:collapse` im einleitenden `<table>`-Tag können Sie dafür sorgen, dass die Rahmen der Einzelzellen und die zwischen Zellen und Außenrahmen zusammenfallen.

Weitere Feinheiten zur Rahmengestaltung müssen Sie allerdings über die Eigenschaften `border`, `border-top`, `border-left`, `border-right` und `border-bottom` festlegen, und zwar sowohl für die Tabelle als auch für alle einzelnen Tabellenzellen. Da das Notieren von sich wiederholenden `style`-Attributen in etlichen Tabellenzellen jedoch nicht sehr zweckmäßig und in CSS viel eleganter über zentrale Formate lösbar ist, werden wir diese Möglichkeit im entsprechenden Zusammenhang behandeln.

Breite von Tabellen und Spalten

Durch Angaben zur CSS-Eigenschaft `width` können Sie sowohl die Breite der gesamten Tabelle als auch einzelne Spaltenbreiten vorgeben.

```
<table border="1" style="width:500px;">
<colgroup>
<col style="width:80px">
<col>
</colgroup>
<tr>
<th>Platz</th>
<th>Name</th>
</tr>
<tr>
<td>1.</td>
<td>Michael Schumacher</td>
</tr>
</table>
```

Info

Eine Zuweisung von `width` im einleitenden `<table>`-Tag gibt eine gewünschte Breite für die gesamte Tabelle vor. Breiten für einzelne Spalten können Sie wahlweise in einem `<col>`-Tag oder im ersten `<th>`– oder `<td>`-Tag der Spalte vorgeben. Auch hierfür ist die CSS-Eigenschaft `width` geeignet. Alternativ sind auch Definitionen mit `min-width` oder `max-width` möglich.

Diese Vorgaben sind jedoch nur Wunschvorgaben, die eingehalten werden, sofern es der Inhalt erlaubt. Ist der Inhalt so geartet, dass Spalten oder die gesamte Tabelle breiter werden als vorgegeben, geht die Kontrolle über die Breite verloren. CSS bietet jedoch die Möglichkeit das Tabellenlayout zu fixieren.

```
<table border="1" style="width:100%;
table-layout:fixed;
border-collapse:collapse">
<tr>
<td style="width:15%; overflow:hidden; font-size:18px">
1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20</td>
<td style="width:30%; overflow:hidden; font-size:18px">
1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20</td>
<td style="width:55%; overflow:hidden; font-size:18px">
1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20</td>
</tr>
</table>
```

Info

In diesem Beispiel wird eine Gesamtbreite der Tabelle von 100% festgelegt. Steht die Tabelle direkt innerhalb von `<body> ... </body>`, bezieht sich diese Angabe auf die gesamte Dokumentbreite, also das Anzeigefenster abzüglich kleiner Default-Abstände vom Fensterrand. Auch für die einzelnen Spalten der Tabelle ist im Beispiel je eine Breite festgelegt: Die erste Spalte soll 15% Breite einnehmen, die zweite 30% und die dritte 55%.

> Der Inhalt der Zellen ist jedoch problematisch.

Da er keine Whitespace-Zeichen enthält, findet der Browser keine Stelle zum Umbrechen. Er würde die einzelnen Spalten und damit die gesamte Tabelle so weit ausdehnen, dass alle Inhalte angezeigt werden können. Durch die Angabe `table-layout:fixed` im einleitenden `<table>`-Tag wird dies jedoch verhindert. Die Angabe bewirkt, dass der Browser die vorgegebenen Breiten auf jeden Fall einhält. Was jedoch mit nicht passenden Spalteninhalten geschehen soll, müssen Sie ebenfalls angeben. Geben Sie nichts an, so hält der Browser die Breitenangaben ein, schiebt aber einen überbreiten Spalteninhalt einfach in die nächste Spalte, was zu hässlichen Überlappungen von Spalteninhalten kommt. In unserem Beispiel haben wir die Variante gewählt, durch Angabe von `overflow:hidden` in den einzelnen Tabellenzellen dafür zu sorgen, dass übergroße Inhalte abgeschnitten werden.

Möglich wäre auch die Angabe `overflow:scroll`, die den Browser dazu veranlassen sollte, in Zellen mit übergroßem Inhalt Scrollleisten anzuzeigen. Doch leider funktioniert das gegenwärtig in kaum einem Browser.

Höhe und Ausrichtung von Tabelleninhalten

Inhalte von Tabellenzellen können unterschiedlich umfangreich sein und dadurch nicht immer die gesamte Breite der Spalte oder die gesamte Höhe der Zeile einnehmen. Die Spaltenbreite orientiert sich, sofern keine Breite vorgegeben wird, an dem breitesten Inhalt der Spalte und an den breitesten Inhalten anderer Spalten. Die Höhe orientiert sich an der Zelle mit dem vertikal umfangreichsten Inhalt innerhalb der Tabellenzeile.

> Ebenso wie Breiten lassen sich natürlich auch Höhen explizit festlegen.

Dazu steht die Eigenschaft `height` zur Verfügung. Auf das `table`-Element bezogen, gibt sie eine gewünschte Höhe für die gesamte Tabelle vor, und auf ein `th`– oder `td`-Element bezogen, gibt sie die gewünschte Höhe der betreffenden Tabellenzeile vor.

Anhand des Quelltextes einer vollständigen HTML-Datei zeigen wir nun einen in der Praxis häufig verlangten Effekt: Eine Tabelle soll den gesamten Raum des Anzeigefensters einnehmen und nur eine einzige Zelle enthalten. Deren Inhalt soll sowohl zentriert als auch vertikal mittig ausgerichtet werden, so dass der Inhalt genau in der Mitte des Anzeigefensters platziert wird:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html style="height:100%">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>Tabellen</title>
</head>
<body style="height:100%; margin:0; padding:0">
<table style="width:100%; height:100%">
<tr>
<td style="text-align:center; vertical-align:middle;">
<div style="font-size:32px; border-bottom:solid black 1px; margin-left:20%;
margin-right:20%; padding-bottom:5px">
Einsam in der Mitte</div>
<div style="font-size:20px; padding-top:5px">
Ein Film von Emanuel Piesepampel</div>
</td>
</tr>
</table>
</body>
</html>
```

Info

Das einleitende `<table>`-Tag bekommt die CSS-Definition `height:100%` zugewiesen. Das allein reicht jedoch nicht aus, da der Tabelleninhalt recht klein ist und das Elternelement, also das `body`-Element, nicht ausfüllt. Da sich aber die 100% auf die Höhe des Elternelements beziehen, bleibt der gewünschte Effekt aus. Deshalb wird dem `body`-Element ebenfalls die Definition `height:100%` zugewiesen. Doch auch das genügt noch nicht, da sich das `body`-Element, was die Höhe betrifft, nicht an den Ausdehnungen des Anzeigefensters orientiert. Deshalb wird auch noch dessen Elternelement, dem `html`-Element, explizit die Angabe `height:100%` zugewiesen. Erst dadurch wird die Tabelle tatsächlich auf 100% Höhe ausgedehnt.

Das nächste Problem entsteht jedoch dadurch, dass die Tabelle nun zwar auf 100% Höhe des Anzeigefensters ausgedehnt wird, der Browser den Inhalt des `body`-Elements jedoch per Default mit etwas Abstand zum Fensterrand versieht. Die Tabelle reicht dadurch in der Höhe ein klein wenig weiter als das Anzeigefenster, so dass ein vertikales Scrollen erforderlich wird, um den gesamten Tabelleninhalt zu sehen. Dieses Problem umgehen wir im Beispiel, indem wir dem `body`-Element `margin:0` und `padding:0` zuweisen. Diese Angaben entfernen alle Default-Abstände zum Fensterrand.

> Da die Tabelle nur eine einzige Zelle besitzt, wird diese auf Grund der Breiten- und Höhenangaben

sowohl in der Breite als auch in der Höhe über das gesamte Anzeigefenster ausgedehnt. Um ihren Inhalt genau in die Mitte des Anzeigefensters zu bringen, richten wir ihn sowohl horizontal zentriert als auch vertikal mittig aus.

Die horizontale Ausrichtung von Tabellenzelleninhalten ist genau so wie das Ausrichten von anderen Block-Elementen mithilfe von `text-align` möglich. Tabelleninhalte können explizit links (`left`), rechts (`right`), zentriert (`center`) oder als Blocksatz (`justify`) ausgerichtet werden.

Für die vertikale Ausrichtung stellt CSS die Eigenschaft `vertical-align` zur Verfügung. Gerade bei Tabellenzellen kommt es häufig vor, dass diese vertikal obenbündig (`top`), mittig (`middle`) oder untenbündig (`bottom`) ausgerichtet werden sollen. Die `vertical-align`-Eigenschaft erlaubt daneben auch noch weitere Angaben, nämlich `baseline` (Ausrichtung an der Schriftgrundlinie des Elternelements), `sub` (Ausrichtung an der Position für tiefgestellten Text), `super` (Ausrichtung an der Position für hochgestellten Text), `text-top` (Ausrichtung an der Schriftoberkante des Elternelements) und `text-bottom` (Ausrichtung an der Schriftunterkante des Elternelements). Für Tabelleninhalte sind jedoch `top`, `middle` und `bottom` in der Regel ausreichend.

Durch die Angaben `text-align:center` und `vertical-align:middle` wird der Inhalt unserer einzigen, das Dokumentfenster voll ausfüllenden Tabellenzelle also genau in der Fenstermitte ausgerichtet. Der Zelleninhalt besteht aus zwei `div`-Bereichen, die durch entsprechende Formatierungen das gewünschte Schriftbild erreichen.

Abstände innerhalb von Zellen und Abstände zwischen Zellen

In vielen Fällen ist es optisch schöner, einen gewissen Abstand zwischen den Gitternetzlinien einer Tabelle und den Zellinhalten zu haben. Dazu wenden Sie einfach die für Innenabstand üblichen CSS-Eigenschaften `padding` (alle vier Seiten), `padding-left` (links), `padding-top`, `padding-right` (rechts) und `padding-bottom` (unten) auf `th`– bzw. `td`-Elemente an.

Ein trauriges Kapitel ist leider der Abstand zwischen Zellen. Dafür steht zwar die CSS-Eigenschaft `border-spacing` zur Verfügung, die genau das mit einer einzigen Angabe im `<table>`-Tag erledigt. Doch leider interpretiert der MS Internet Explorer diese Eigenschaft nicht. Da bleibt nichts anderes, als auf das ältere, aber nicht zum Strict-Standard gehörende HTML-Attribut `cellspacing` auszuweichen. In diesem Fall müssen Sie allerdings, wenn Sie konsequent sein wollen, den HTML-Dokumenttyp auf die Transitional-Variante ändern. Nachfolgend das Beispiel einer Tabelle, das auch im Internet Explorer 6.0 funktioniert:

```
<table cellspacing="10" style="background-color:teal; border-spacing:10px">
<thead style="background-color:white;">
<tr>
<th>Äpfel</th>
<th>Birnen</th>
</tr>
</thead>
<tfoot style="background-color:aqua">
<tr>
<td>beliebter</td>
<td>weniger beliebt</td>
</tr>
</tfoot>
<tbody style="background-color:aqua">
<tr>
<td>12 Einheiten Vitamin C</td>
<td>5 Einheiten Vitamin C</td>
</tr>
<tr>
<td>regt die Verdauung an</td>
<td>wirkt entwässernd</td>
</tr>
</tbody>
</table>
```

Info

Im Beispiel wird ein 10 Pixel großer Abstand zwischen den Tabellenzellen definiert. CSS-gerecht wird dazu

im `style`-Attribut des `<table>`-Tags die Angabe `border-spacing:10px` notiert. Um rückwärtskompatibel zum MS Internet Explorer zu bleiben, wird im einleitenden `<table>`-Tag außerdem das Attribut `cellspacing="10"` notiert, welches den gleichen Effekt hat. Die Angabe zum Zellabstand funktioniert logischerweise nicht, wenn gleichzeitig `border-collapse:collapse` angegeben wird, da dies ja die Zellrahmen zusammenfallen lässt, was nur bei einem Zellabstand von 0 möglich ist.

Wenn die Tabelle einen sichtbaren Rahmen hat, ist der definierte Zellenabstand im Gitternetz gut erkennbar. Im obigen Beispiel entschieden wir uns jedoch für eine andere Visualisierung. Die Tabelle hat keine sichtbaren Rahmen und Gitternetzlinien, doch dafür wurden Hintergrundfarben für die gesamte Tabelle und für einzelne Zellen definiert – mit Hilfe der bereits bekannten CSS-Eigenschaft `background-color`. Auf das `table`-Element angewendet, wird eine Hintergrundfarbe für die Tabelle bestimmt. Diese wird im Beispiel aber nur gut sichtbar, weil ein großzügiger Zellabstand definiert wurde. Für einzelne Tabellenzellen, welche die gleiche Hintergrundfarbe erhalten sollen, genügt es, `background-color` auf das zugehörige `tr`-Element oder noch allgemeiner auf das `thead`-, `tfoot`– oder `tbody`-Element anzuwenden.

Die Formatierung von texttragenden Elementen wie Überschriften, Textabsätzen, Listenpunkten oder Tabellenzellen gehört zu den vordringlichsten Wünschen der meisten Autoren, die ein HTML-Dokument ohne CSS-Angaben im Browser betrachten. Die Times-Schriftart, welche die meisten Browser als Default-Schriftart eingestellt haben, wirkt auf viele HTML-Autoren nicht sehr ästhetisch. Auch die Browser-Defaults für Schriftgrößenverhältnisse zwischen Überschriften und Text werden oftmals als unbefriedigend empfunden. Und last but not least wirken schwarzweiße Textwüsten am Bildschirm viel ermüdender als etwa in einem Buch. Farben sollen also her.

CSS erfüllt fast alle Formatierwünsche. Allerdings stößt gerade die Schriftformatierung in der Praxis an sehr enge Grenzen. Der Grund ist, dass Webseiten auf den unterschiedlichsten Rechnertypen, Benutzeroberflächen und Anzeigegeräten dargestellt werden können. Wenn Sie beispielsweise Ihre Webseiten unter MS Windows entwickeln, können Sie nicht erwarten, dass Linux/KDE-Anwender Schriftarten wie Arial, die unter Windows zum Standard gehören, installiert haben.

> Umgekehrt sind dort andere Schriftarten viel verbreiteter.

Kaum weniger problematisch sind auch die Schriftgrößen. Im Internet herrscht ein regelrechter Glaubenskrieg zwischen aggressiven Vertretern, die einem Webdesigner am liebsten jede Einflussnahme auf Schriftgrößen verbieten wollen, und ignoranten Design-Freaks, die leider meistens auch unter einem schlimmen Schriftminiaturisierungswahn leiden.

Als Grundregel für die Schriftformatierung kann gelten: Außer bei Ausnahmefällen (z.B. individuell formatierte, auffällige Schriftzüge) sollten Sie mit starken Abweichungen gegenüber den Browser-Defaults stets behutsam sein.

Schriftart

Schriftarten können Sie mit `font-family` definieren. Es besteht jedoch keine Garantie, dass die angegebene Schriftart bei allen Anwendern angezeigt werden kann, da dies sehr stark von der jeweiligen Hard- und Softwareumgebung abhängt. Damit sich die Trefferquote verbessert, ist es bei der `font-family`-Eigenschaft nicht nur möglich, sondern auch dringend zu empfehlen, stets mehrere Schriftarten anzugeben.

```
<p style="font-family:Tahoma, Arial, Helvetica, sans-serif">
Text
</p>
```

Info

Im Beispiel werden drei Schriftarten angegeben. Alle Schriftarten werden durch Kommata getrennt. Die Reihenfolge entspricht der Priorisierung. Das bedeutet im Beispiel: Wenn die Schriftart Tahoma verfügbar ist, soll diese verwendet werden. Ist sie nicht verfügbar, soll Arial verwendet werden. Ist Arial ebenfalls nicht verfügbar, soll Helvetica verwendet werden. Ist auch Helvetica nicht verfügbar, soll irgendeine verfügbare serifenlose Schriftart verwendet werden.

> Die angegebene Kombination im Beispiel geht von folgender Überlegung aus:

Eigentlich gewünscht (aus welchen Gründen auch immer) ist die Schriftart Tahoma. Diese ist jedoch eine typische MS-Windows-Schriftart, die auch üblicherweise nur zusammen mit MS Office installiert wird. Wegen der hohen Verbreitung von MS Office und MS Windows ist natürlich anzunehmen, dass Tahoma bei vielen Anwendern angezeigt werden kann. Die zweite Wahl, Arial, stützt sich auf die Überlegung, dass die überwiegende Anzahl der Anwender MS Windows einsetzt, wo Arial seit jeher eine der Grundschriftarten ist. Ebenso wie Tahoma ist Arial eine serifenlose Schrift, auch wenn sie ansonsten typografisch durchaus von Tahoma abweicht. Die dritte Wahl, Helvetica, bedient sich des Wissens, dass auf vielen anderen Benutzeroberflächen als MS Windows die Schriftart Helvetica als eine der meist verfügbaren Grundschriftarten installiert ist. Sollte das alles nichts helfen, so wird der Browser durch die Angabe der generischen Schriftart sans-serif angewiesen, irgendeine verfügbare serifenlose Schrift zu verwenden. Voraussetzung dafür ist natürlich, dass die zugrunde liegende Benutzeroberfläche dem Browser eine Möglichkeit der Kategorisierung von Schriftarten anbietet.

Die letzte Angabe bei einer Wertzuweisung an `font-family` sollte stets eine so genannte **generische Schriftart** sein. CSS stellt folgende Angaben für generische Schriftarten zur Verfügung:

Wert	Bedeutung
`serif`	Schriftart mit Serifen

Wert	Bedeutung
`sans-serif`	Schriftart ohne Serifen
`cursive`	Schriftart für Schreibschrift
`fantasy`	Schriftart für eine ungewöhnliche Schrift
`monospace`	Nichtproportionale Schriftart (dicktengleiche Zeichen)

Info

Verwenden Sie auf ein und derselben Webseite nicht zu viele unterschiedliche Schriftarten. Das gilt als grober typografischer Fehler und wird auch tatsächlich von den meisten Anwendern als eher unprofessionell empfunden. Als Faustregel gilt: nicht mehr als zwei unterschiedliche Schriftarten.

> Die angemessene Wahl der Schriftart hängt auch von den Textinhalten ab.

Bei einem lateinischen Text wirkt eine Serifenschrift wie die obere Book Antiqua edler und inhaltstypischer. Bei einer technischen Dokumentation dagegen sind serifenlose Schriften üblicher. Doch auch dabei gibt es Unterschiede. Schriftarten wie die Century Gothic wirken schick, können aber bei längeren Texten am Bildschirm schwerer lesbar sein als etwa Arial.

Schriftgröße

Die Schriftgröße können Sie mit der CSS-Eigenschaft `font-size` bestimmen.

```
<body>
<h1 style="font-size:36px"><span style="font-
size:1.5em">Große</span>Überschrift</h1>
<h1 style="font-size:1.5em">Kleine Überschrift</h1>
<h1>Unformatierte Überschrift</h1>
</body>
```

Info

Das Beispiel verdeutlicht die Zusammenhänge bei der Schriftgrößendefinition an Hand von drei h1-

Überschriften. In der ersten h1-Überschrift wird durch die Definition `font-size:36px` eine gemessen am Ausgabegerät absolute Größe festgelegt. Das innere `span`-Element erhält dagegen die relative Größenangabe `font-size:1.5em`. Für die Maßeinheit `em` gilt: Wird sie auf andere Größenangaben als die Schriftart angewendet, ist stets 1em die Größe der Schriftart des aktuellen Elements. Wird sie dagegen auf die Schriftgröße selbst (CSS-Eigenschaft `font-size`) angewendet, so entspricht 1em der Schriftart des Elternelements. In unserem Beispiel wird em auf die Schriftgröße selbst angewendet. Zur Berechnung der anzuzeigenden Schriftgröße bei der Angabe `font-size:1.5em` greift der Browser daher auf die Schriftgröße des Elternelements zurück, also auf das h1-Element. Diese beträgt 36px, also wird das Wort „Große“ mit dem Faktor 1.5 angezeigt, d.h. in einer Schriftgröße von 54 Pixel.

> Möglicherweise verwirrend ist die zweite h1-Überschrift,

wenn man die Wirkung im Browser betrachtet. Obwohl sie die Angabe `font-size:1.5em` enthält, erscheint sie im Browser kleiner als die dritte h1-Überschrift, der gar keine CSS-Formatierung zugewiesen ist. Der Grund ist auch hier das zuvor beschriebene Verhalten bei der Bemaßung mit `em`. Zur Berechnung der tatsächlich anzuzeigenden Schriftgröße wird die Schriftgröße des Elternelements verwendet. Das gemeinsame Elternelement der h1-Überschriften ist das `body`-Element, also der Dateikörper selbst. Dessen Grundschriftgröße entspricht natürlich der von normalem Fließtext, und nicht der von Überschriften erster Ordnung. Angenommen, für Fließtext ist im Browser eine Grundschriftgröße von 15 Pixel eingestellt, so führt die Angabe 1.5em zu einer berechneten Schriftgröße von 22,5 Pixel. Halbe Pixel können natürlich aus technisch nachvollziehbaren Gründen nicht angezeigt werden – der Browser wird also 22 oder 23 Pixel Größe verwenden.

Die in der Drucktechnik üblichen Schriftgrößenbemaßungen **Punkt** (pt) und **Pica** (pc) sind für Webseiten, die vornehmlich auf Bildschirmen angezeigt werden und nicht zum Ausdrucken bestimmt sind, weniger geeignet. Der Grund ist, dass diese Angaben bei der Bildschirmanzeige intern auf Pixel umgerechnet werden müssen. Der Umrechnungsfaktor ist jedoch systemabhängig. Unter MS Windows beispielsweise entspricht 1 Punkt intern 1,33 Pixel, weil Windows mit 96 dpi arbeitet, während 1 Punkt auf einem Macintosh- oder Linux-Rechner genau in 1 Pixel umgerechnet wird, weil diese Systeme mit 72 dpi umrechnen. In der Praxis bedeutet dies, dass eine mit `font-size:10pt` definierte Schrift auf einem Windows-Rechner 13 Pixel groß erscheint, während die gleiche Schrift etwa auf einem Macintosh-Rechner nur 10 Pixel groß dargestellt wird. Dieser Unterschied ist durchaus erheblich – viele Anwender empfinden 13 Pixel Darstellungsgröße als angenehm, aber 10 Pixel als zu klein.

Schriftfarbe

Die Vordergrundfarbe von Text definieren Sie in CSS mit der Eigenschaft `color`.

```
<body style="background-color:rgb(204,255,204)">
<h1 style="color:rgb(0,80,0)">Rettet unser Grün!</h1>
</body>
```

Info

Zur Bestimmung der Schriftfarbe notieren Sie eine Farbangabe wie in Farbangaben in CCS beschrieben. Beim Definieren von Schriftfarben müssen Sie vor allem auf die Kombination mit der Hintergrundfarbe achten. Ein häufig begangener Fehler besteht darin, dass nirgends eine Hintergrundfarbe definiert wird. Die Annahme, dass die dunkelblaue Schrift auf dem weißen Default-Browser-Fensterhintergrund ordentlich aussieht, trügt nämlich: Ein Anwender könnte auch Schwarz als seinen Default-Fensterhintergrund eingestellt haben. In diesem Fall (Dunkelblau auf Schwarz) würde Ihre Seite gar keinen guten Eindruck machen.

> Das obige Beispiel vermeidet diesen Fehler,

indem im `body`-Element mit `background-color` eine hellgrüne Hintergrundfarbe definiert wird. Die Farbdefinition in der h1-Überschrift (dunkles Grün) kann sich damit auf eine passende Hintergrundfarbe verlassen.

Vorder- und Hintergrundfarbe sollten einen guten Kontrast haben. Zu schwache Kontraste bereiten Mitmenschen mit Sehschwächen schnell Leseschwierigkeiten. Es gibt verschiedene Arten von Farbkontrasten, z.B. Komplementärfarben wie Gelb und Blau, oder Hell-Dunkel-Kontraste mit der gleichen Grundfarbe, wie im obigen Beispiel Dunkelgrün und Hellgrün. Nicht wenige Menschen leiden an einer Rot-Grün-Sehschwäche. Mischungen von Rot und Grün können dabei leicht zu Problemen führen. Auch das sollte bei der Farbwahl berücksichtigt werden.

Weitere Angaben zur Schriftformatierung

Weitere Formatierungsmöglichkeiten zur Beeinflussung des Schriftbilds betreffen das Schriftgewicht, den Schriftstil, den Wort- und Zeichenabstand, die Zeilenhöhe sowie verschiedene Effekte.

Das **Schriftgewicht** können Sie mit `font-weight` festlegen. Die Angabe `font-weight:bold` erzwingt eine Fettschrift, während `font-weight:normal` ein normales Schriftgewicht bewirkt. Bei der Default-Darstellung von HTML-Elementen verwenden moderne grafische Browser teilweise Fettschrift, teilweise Normalschrift. Für Überschriften bis zur vierten Ebene werden meistens Fettschriften verwendet. Durch eine Angabe wie `<h1 style="font-weight:normal"> ... </h1>` können Sie also beispielsweise die Default-Darstellung aushebeln. Umgekehrt können Sie mit einer Angabe wie `<p style="font-weight:bold"> ... </p>` das Schriftgewicht eines Elements, dessen Default-Darstellung normalerweise nicht fett ist, Fettschrift erzwingen.

> Der **Schriftstil** bestimmt die Neigung der Schrift

und kann durch `font-style` festgelegt werden. Wie beim Schriftgewicht wird eine abweichende Schriftneigung von vielen Browsern bei bestimmten Elementen, z.B. beim em-Element, zur Darstellung verwendet. Durch `font-style:italic` oder `font-style:oblique` erzwingen Sie eine Kursivschrift und durch `font-style:normal` eine normale Darstellung.

Manche Schriftarten benötigen Korrekturen bei den Defaults für Zeichen- und Wortabstände und bei der Zeilenhöhe. Erst dadurch ergibt sich ein abgerundetes und angenehmes Schriftbild.

Den **Zeichenabstand** bestimmen Sie durch `letter-spacing` und den Abstand zwischen Wörtern mit `word-spacing`. Die Zeilenhöhe eines Block-Elements kann mit `line-height` verändert werden. Bei fest definierter Zeilenhöhe muss die Schriftgröße aller davon betroffenen Textinhalte dazu passen, andernfalls könnte Text bei der Darstellung abgeschnitten werden. Das gilt besonders, wenn ein Block-Element Inline-Elemente enthält, die wiederum größere Schriftgrößen erzwingen. Auch im Text platzierte Grafiken können bei definierter Zeilenhöhe abgeschnitten werden. Das Experimentieren mit Zeichenabstand, Wortabstand und Zeilenhöhe empfiehlt sich vor allem bei schwer lesbaren Schriften, z.B. bei engen Schriften mit schmalen Zeichen.

Eine weitere Möglichkeit besteht darin, Text unterstrichen oder durchgestrichen darzustellen. Dazu dient die CSS-Eigenschaft `text-decoration`.

```
<p>Unser Rasierer - jetzt nur noch <span style="text-decoration:line-
through">49,95 €</span><span style="text-decoration:underline">39,95 €</span>
</p>
```

Info

Mit `text-decoration:underline` können Sie Text unterstreichen und mit `text-decoration:line-through` streichen Sie ihn durch.

Eine interessante Möglichkeit der Formatierung bietet schließlich auch noch die Angabe `font-variant:small-caps`. Dadurch wird Text in Kapitälchen dargestellt, d.h., Kleinbuchstaben werden als Großbuchstaben angezeigt, jedoch in der Größe von Kleinbuchstaben.

In den voranstehenden Abschnitten haben Sie die wesentlichen HTML-Elemente zur Strukturierung von Textinhalten kennen gelernt. Dabei wurden Sie mehrfach darauf vertröstet, dass eine schicke Optik mit HTML allein nicht zu bewerkstelligen ist, sondern dass dafür Cascading Style Sheets (**CSS**) erforderlich sind. Bevor wir deshalb weitere HTML-Elemente behandeln, möchten wir Sie nicht länger auf die Folter spannen. In diesem Abschnitt werden Sie mit den Formatiermöglichkeiten von **CSS** vertraut gemacht.

Mit CSS Text gestalten

Zunächst werden wir mit dem Attribut `style=` arbeiten. Dieses Attribut dürfen Sie in allen HTML-Elementen notieren, die „sichtbare Inhalte“ haben, egal ob es sich um Block- oder Inline-Elemente handelt. In der Wertzuweisung an das `style`-Attribut können Sie dann mit Hilfe CSS-eigener Syntax Formateigenschaften notieren, die dem jeweiligen Element zugeordnet werden. Beginnen wir mit dem Komplettlisting einer kleinen HTML-Datei, in der diverse Elemente mit Hilfe des `style`-Attributs formatiert werden:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="de">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>Arbeiten mit dem style-Attribut</title>
</head>
<body style="background-color:#FFFFC0; margin-left:30px; margin-right:30px;
font-family:Tahoma,Helvetica,sans-serif">
<h1 style="font-size:28px; padding-bottom:20px; border-bottom:solid blue 4px;">
Das <em style="color:red">style</em>-Attribut</h1>
<p style="font-size:15px;">
Das <em style="color:red; font-weight:bold">style</em>-Attribut eignet sich zur
Formatierung einzelner HTML-Elemente.</p>
</body>
</html>
```

Info

Die Abbildungen verdeutlichen, wie durch wenige CSS-Formatdefinitionen aus einer „nackten“ HTML-Datei eine „gestaltete“ Seite werden kann.

Zunächst hat das <body>-Tag ein `style`-Attribut erhalten. Da das `body`-Element den gesamten sichtbaren Dateikörper auszeichnet, wirken sich CSS-Definitionen für dieses Element auf die gesamte sichtbare Seite aus. Im Beispiel wird eine Hintergrundfarbe (`background-color`), ein linker und rechter Rand (`margin-left`, `margin-right`) sowie eine Angabe zur Grundschriftart (`font-family`) für die gesamte Seite notiert.

> Die Hintergrundfarbe färbt das gesamte Browser-Fenster ein.

Die Ränder links und rechts werden bei allen Inhalten innerhalb des Dateikörpers eingehalten und ebenso die Grundschriftart – solange in einem einzelnen Element nichts definiert wird, was diese Definitionen überschreibt. Damit wird bereits eine Grundeigenschaft von CSS deutlich: nämlich das Prinzip der Vererbung von Eigenschaften. Wir werden darauf später noch genauer eingehen.

Innerhalb des `body`-Elements folgt im Beispiel eine Seitenüberschrift 1. Ordnung. Diese erbt die Schriftart des `body`-Elements, bekommt aber eine individuelle Schriftgröße (`font-size`), einen Rahmen unterhalb (`border-bottom`) sowie einen Innenabstand unten (`padding-bottom`) zugewiesen. Der Innenabstand sorgt dafür, dass die blaue Linie, die in Wirklichkeit der Rahmen unten ist, nicht zu nah am Überschriftentext klebt.

Innerhalb der `h1`-Überschrift ist das Wort „`style`“ durch ein `em`-Element hervorgehoben. Dieses erbt wiederum alle Formateigenschaften der Überschrift und bekommt aber außerdem eine rote Schriftfarbe (`color`) zugewiesen. Dass das Wort dann in den meisten Browsern außerdem kursiv dargestellt wird, liegt an den Default-Einstellungen, mit denen Browser HTML-Elemente darstellen. Diese gehören ebenfalls zum „Erbe“. CSS-Formatierungen wirken immer nur additiv oder überschreibend. Um die Kursivdarstellung etwa beim `em`-Element wegzubekommen, könnte man die CSS-Formatierung `font-style:normal` notieren. Dadurch würde die Browser-Default-Einstellung „kursiv“ überschrieben.

Auch das `p`– und das `em`-Element im Beispiel erhalten individuelle CSS-Formatierungen durch ein `style`-Attribut im Start-Tag.

CSS-Eigenschaften für Abstände und Ausrichtung

Während die Schriftformatierung alle sichtbaren Elemente betrifft, sind die nachfolgend vorgestellten CSS-Eigenschaften vornehmlich für Block-Elemente gedacht. Zwischen Überschriften und Text sind oft andere als die vom Browser gewählten Default-Abstände oben und unten wünschenswert. Absätze sollen möglicherweise auch mal eingerückt werden und das Standard-Feature der Textausrichtung, also linksbündig, zentriert, rechtsbündig und Blocksatz, wird ebenfalls häufig verlangt.

Abstände zwischen Elementen

Die CSS-Basiseigenschaft zum Definieren von Abständen lautet `margin` (zu Deutsch: Rand). Im Fließtext sind jedoch vor allem die Abstände zwischen aufeinanderfolgenden Block-Elementen interessant, also etwa die Abstände zwischen Überschrift und Textabsätzen oder zwischen zwei Textabsätzen. Deshalb erlaubt es CSS zusätzlich, zwischen den Einzeleigenschaften `margin-top` (Abstand oben), `margin-bottom` (Abstand unten), `margin-left` (Abstand links) und `margin-right` (Abstand rechts) zu unterscheiden.

Gerade bei Abständen zwischen Blockelementen im Fließtext ist jedoch eine Besonderheit zu beachten.

```
<p style="margin-bottom: 8px">Ein Textabsatz</p>
<p style="margin-top: 5px">Noch ein Textabsatz</p>
```

Info

Vom CSS-Boxmodell her betrachtet, müssten sich die beiden Randangaben einfach addieren, so dass zwischen den beiden Absätzen ein Abstand von 13 Pixel entsteht. Dies ist jedoch nicht der Fall. Im normalen Fließtext kollabieren zwei Angaben zu einer. Maßgeblich ist dabei einfach der höhere Wert. Da dieser im obigen Beispiel 8 Pixel beträgt, bleibt ein tatsächlicher Abstand von 8 Pixel zwischen den beiden p-Elementen.

> Das Kollabierungsprinzip gilt jedoch nur für Elemente,

die im Textfluss vertikal aufeinander folgen. Für Elemente, die nebeneinander angeordnet sind und die Angaben zu `margin-left` bzw. `margin-right` enthalten, gilt das Prinzip nicht.

Ausrichtung von Inhalten

Block-Elemente, so haben wir bei der Betrachtung des Boxmodells gesehen, nehmen so viel Breite ein wie verfügbar ist. Der Elementinhalt eines solchen Elements kann innerhalb dieser Breite links, zentriert oder rechts ausgerichtet werden. Text kann außerdem als Blocksatz ausgerichtet werden:

Die linksbündige Ausrichtung ist bei den meisten Elementen das eingestellte Default-Verhalten. Eine Ausnahme ist beispielsweise das `th`-Element für Tabellenkopfzellen, die von den meisten Browsern per Default zentriert ausgerichtet werden.

Mit `text-align:left` erzwingen Sie in jedem Fall eine linksbündige Ausrichtung, mit `text-align:right` eine rechtsbündige Ausrichtung, mit `text-align:center` eine horizontal mittige Zentrierung des Inhalts und mit `text-align:justify` einen Blocksatz.

> Da die Browser für HTML keine automatische Silbentrennung anbieten,

kann ein erzwungener Blocksatz besonders bei Elementen mit begrenzter Breite zu unschönen Wortzwischenraumlücken führen, wie die Abbildung zeigt. In dem abgebildeten Beispiel wurde definiert:

```
<p style="width: 200px; text-align: justify">Lorem ipsum ... </p>
```

Info

Durch die Angabe `width:200px` wird die Breite des p-Elements auf 200 Pixel begrenzt. Ähnliche Breitenverhältnisse ergeben sich aber auch häufig in Tabellenspalten.

Die CSS-Eigenschaft `text-align` bezieht sich stets nur auf den Inhalt eines Blockelements, nicht auf das Element selbst. Falls Sie das Element selbst ausrichten möchten, müssen Sie mit `margin-left` und `margin-right` arbeiten.

```
<p style="width: 200px; text-align: right">Lorem ipsum ...</p>
<p style="width: 200px; text-align: right; margin-right: 0px; margin-left:
auto">Lorem ipsum ...</p>
```

Info

In dem Beispiel sind zwei Textabsätze notiert. Beide haben eine reduzierte Breite von 200 Pixel und der Inhalt beider Elemente wird rechtsbündig ausgerichtet. Das erste p-Element wird jedoch selbst linksbündig innerhalb der verfügbaren Gesamtbreite ausgerichtet, während das zweite p-Element rechtsbündig ausgerichtet wird. Die folgende Abbildung demonstriert das Ergebnis:

Die durch `text-align:right` bewirkte rechtsbündige Ausrichtung des Elementinhalts

hält sich in jedem Fall an die mit `width` erzwungene Elementbreite. Die Ausrichtung des Elements selbst wird wie im zweiten p-Element zu sehen durch Angaben zu `margin-left` und `margin-right`, also zu den horizontalen Elementrändern, gesteuert. Die rechtsbündige Elementausrichtung wird durch `margin-right:0px` angewiesen. Doch erst im Zusammenspiel mit der Angabe `margin-left:auto` funktioniert die Ausrichtung tatsächlich wie gewünscht.

Einrückungen

Soll beispielsweise ein einzelner Textabsatz links eingerückt werden, können Sie dies ganz einfach mit `margin-left` erreichen. Weisen Sie einfach den gewünschten linken Rand zu, und das Element wird entsprechend eingerückt dargestellt.

Aus belletristischen Büchern ist auch die Einrückung der ersten Textzeile eines Absatzes bekannt. Für diesen Effekt steht die CSS-Eigenschaft `text-indent` zur Verfügung. Durch eine Angabe wie `text-indent:5mm` rücken Sie die erste Textzeile 5 Millimeter ein. Mit negativen Werten bewirken Sie übrigens eine Ausrückung der ersten Zeile gegenüber den weiteren Zeilen, also z.B. `text-indent:-5mm`.

CSS unterstützt das RGB-Farbmodell. Über dieses Modell sind ca. 16,7 Millionen unterschiedliche Farben definierbar. RGB steht für **Red Green Blue** (zu Deutsch: Rot Grün Blau). Im RGB-Farbmodell wird jede Farbe aus den Anteilen der Grundfarben Rot, Grün und Blau definiert. Die drei Werte aus diesen Grundfarben werden additiv behandelt, wodurch Mischfarben entstehen. Jeder der drei Werte kann zwischen 0 und 255 betragen. Dadurch entstehen 256 × 256 × 256, also ca. 16,7 Millionen mögliche Farben. Wie viele unterschiedliche Farben bei einem Anwender tatsächlich anzeigbar sind, hängt natürlich von dessen Hard- und Software ab, aber letztlich auch vom Sehvermögen.

Der Wert 0 bedeutet: keinen Anteil an der entsprechenden Grundfarbe. Der Wert 255 bedeutet: maximalen Anteil an der entsprechenden Grundfarbe. Angegeben werden die Werte in der Reihenfolge für die Farben Rot, Grün und Blau.

> Wenn die Werte aller drei Grundfarben gleich sind, bedeutet dies eine Grauabstufung.

Die Wertekombination 0, 0, 0 (Rot, Grün, Blau) erzeugt die Farbe Schwarz, die Kombination 255, 255, 255 die Farbe Weiß, eine Kombination wie 160, 160, 160 ein mittleres Grau. Durch Überwiegen einzelner Werte entstehen Farbtöne und deren Mischung. Die Wertkombination 0, 0, 255 bewirkt beispielsweise ein maximal kräftiges und reines Blau, und 255, 0, 255 ein maximal kräftiges Magenta, da in diesem Fall Rot und Blau gemischt werden.

Zum Auswählen von Mischfarben sollten Sie ein Farbwähl-Tool benutzen. Grafikprogramme verfügen in der Regel über die Möglichkeit, RGB-Farben auszuwählen, aber auch HTML-Editoren.

Bei der interaktiven Farbauswahl in solchen Dialogen können Sie entweder RGB-Werte direkt eingeben oder durch Eingabe anderer Faktoren wie Farbton, Sättigung und Helligkeit berechnen lassen. Auch durch Maussteuerung anhand eines Farbtabletts lässt sich die gewünschte Farbe intuitiv ermitteln.

RGB-Farbangaben in CSS

In CSS können Sie RGB-Farbangaben folgendermaßen notieren. (Beispiel):

```
<h2 style="color:rgb(0,0,255)">Blaue Überschrift</h2>
```

Info

Die Wertzuweisung hat also das Schema `rgb` (`Rotwert`, `Grünwert`, `Blauwert`). Anstelle der Zahlen 0 bis 255 können Sie auch Prozentwerte von 0 bis 100% angeben. Das nachfolgende Beispiel hat den gleichen Effekt wie das obige:

```
<h2 style="color:rgb(0%,0%,100%)">Blaue Überschrift</h2>
```

Hexadezimale RGB-Farbangaben

Obwohl die Notationsform `rgb` (`Rotwert`, `Grünwert`, `Blauwert`) viel praktischer ist, findet man in der heutigen Webpraxis noch immer wesentlich häufiger eine eher kryptische Variante von Farbangaben. Der Grund dafür ist, dass Farben auch in verschiedenen, heute allerdings nicht mehr empfehlenswerten HTML-Attributen angegeben werden können und HTML im Gegensatz zu CSS ausschließlich die hexadezimale Notationsform von Farbangaben unterstützt.

Diese Notationsform hat das Schema #XXXXXX, also ein Gatterzeichen, gefolgt von sechs Zeichen, ohne Leerzeichen dazwischen. Die sechs hier mit X bezeichneten Zeichen stellen Hexadezimalziffern dar. Hexadezimale Ziffern sind die dezimalen Ziffern `0` bis 9 sowie die Buchstaben A, B, C, D, E und F. Eine Hexadezimalziffer kann also insgesamt 16 Werte annehmen (deshalb: Hexadezimalsystem).

Die sechs Ziffern der hexadezimalen Notationsform für Farben sind folgendermaßen zu lesen:

- Ziffer 1 und 2 bestimmen den Rotanteil
- Ziffer 3 und 4 bestimmen den Grünanteil
- Ziffern 5 und 6 bestimmen den Blauanteil

Da mit jeder Hexadezimalziffer 16 Zahlenwerte darstellbar sind, lassen sich mithilfe von zwei solchen Ziffern 16 × 16 = 256 Zahlenwerte darstellen – also genau der Wertebereich eines Farbwerts im RGB-Modell. Es genügen demnach zwei Hexadezimalziffern pro Farbanteil.

Der Wert 255 entspricht beispielsweise dem Hexadezimalwert FF und der Wert 192 lautet hexadezimal `C0` (C entspricht dezimal 12, C0 bedeutet also 12 × 16 + 0 = 192). Die Notation #FFFFC0 bedeutet also das Gleiche wie `rgb`(255, 255, 192) – ein helles, mattes Gelb.

> Selbst wenn Sie die Hexadezimalnotation nicht mehr benötigen,

weil CSS die praktischere RGB-Schreibweise anbietet, so sollten Sie diese ältere Notationsform dennoch kennen und verstehen, da einfach noch so viele Webseiten damit arbeiten. Hin und wieder stößt man auch auf eine abgekürzte Hexadezimalnotation mit nur drei Ziffern, also etwa #FFF oder #99C. Die verkürzte Schreibweise bedeutet intern eine Verdopplung der Ziffernfolge. #FFF bedeutet in Wirklichkeit #FFFFFF, #99C bedeutet #9999CC.

Farbnamen

Noch einfacher als die RGB-Schreibweise für Farbdefinitionen sind Farbnamen. CSS selbst unterstützt folgende Farbnamen.

Farbname	Hexadezimal	RGB
black	#000000	rgb(0, 0, 0)
maroon	#800000	rgb(128, 0, 0)
green	#008000	rgb(0, 128, 0)
olive	#808000	rgb(0, 128, 128)
navy	#000080	rgb(0, 0, 128)
purple	#800080	rgb(128, 0, 128)
teal	#008080	rgb(0, 128, 128)
gray	#808080	rgb(128, 128, 128)
silver	#C0C0C0	rgb(192, 192, 192)
red	#FF0000	rgb(255, 0, 0)
lime	#00FF00	rgb(0, 255, 0)
yellow	#FFFF00	rgb(255, 255, 0)
blue	#0000FF	rgb(0, 0, 255)
fuchsia	#FF00FF	rgb(255, 0, 255)
aqua	#00FFFF	rgb(0, 255, 255)

Farbname	Hexadezimal	RGB
white	#FFFFFF	rgb(255, 255, 255)
orange	#FFA500	rgb(255, 165, 0)

Info

Insider werden erkennen, dass es sich bei den ersten 16 der 17 aufgelisteten Farben um die klassischen VGA-Farben handelt. Die 17. Farbe, `orange`, wurde vom W3-Konsortium zusätzlich für CSS definiert, weil Orange im klassischen VGA-Schema nicht vorkommt.

```
<h2 style="color:blue">Blaue Überschrift</h2>
```

Info

Neben diesen Farbnamen erlaubt CSS auch die Angaben umgebungsspezifischer Farben. Diese Farben bezeichnen Farbwerte, die Sie als Designer gar nicht kennen, da es sich um Farben handelt, die bei jeden Anwender anders sind und die er sich selber in seiner Benutzeroberfläche eingestellt hat. Wenn Sie ausschließlich mit diesen Farbnamen arbeiten, geben Sie zwar die farbliche Kontrolle ab, doch Sie kommen Ihren Anwendern optimal entgegen – die Annahme vorausgesetzt, dass alle Anwender von sich aus für sie optimale und angenehme Farben in ihrer Benutzeroberfläche einstellen.

```
<h2 style="background-color:ActiveBorder; color:ActiveCaption">Überschrift in Systemfarben</h2>
```

Info

Im Beispiel wird für die Überschrift als Hintergrundfarbe (`background-color`) die beim Anwender eingestellte Hintergrundfarbe für aktive Fenstertitel definiert und als Textfarbe (`color`) die zugehörige Vordergrundfarbe des aktiven Fenstertitels.

Die nachfolgende Tabelle listet erlaubte Umgebungsfarbnamen auf.

Farbname	Bedeutung
`ActiveBorder`	Farbe der aktiven Fenstertitelzeile
`ActiveCaption`	Farbe des Titels in der aktiven Fenstertitelzeile
`AppWorkspace`	Farbe des Hintergrunds der aktiven Anwendung

Farbname	Bedeutung
`Background`	Farbe des Desktop-Hintergrunds
`ButtonFace`	Farbe von Buttons in Dialogfenstern
`ButtonHighlight`	Helle Farbe für 3D-Schatten von Buttons
`ButtonShadow`	Helle Farbe für 3D-Schatten von Buttons
`ButtonText`	Farbe von Button-Beschriftungen
`CaptionText`	Farbe von Überschriften in Dialogfenstern
`GrayText`	Farbe von deaktiviertem Text Dialogfenstern
`Highlight`	Farbe von selektierten Einträgen in Auswahllisten
`HighlightText`	Farbe von selektiertem Text
`InactiveBorder`	Farbe einer inaktiven Fenstertitelzeile
`InactiveCaption`	Hintergrundfarbe des Titels der inaktiven Fenstertitelzeile
`InactiveCaptionText`	Farbe des Titels der inaktiven Fenstertitelzeile
`InfoBackground`	Hintergrundfarbe für Tooltips und Hints
`InfoText`	Textfarbe für Tooltips und Hints
`Menu`	Farbe für Menüleisten

Farbname	Bedeutung
MenuText	Farbe für Menüeinträge
Scrollbar	Farbe der Scrollleisten in Fenstern
ThreeDDarkShadow	Dunkle Farbe bei Schatten von 3D-Elementen
ThreeDFace	Farbe von 3D-Elementen
ThreeDHighlight	Farbe von gerade angeklickten 3D-Elementen
ThreeDLightShadow	Helle Farbe bei Schatten von 3D-Elementen
ThreeDShadow	Dunkle Farbe bei Schatten von 3D-Elementen
Window	Hintergrundfarbe von Dokumentfenstern
WindowFrame	Farbe von Fensterrahmen
WindowText	Farbe von normalem Text in Dokumentfenstern

Websichere Farben

Bei älteren Computern, aber auch bei Mini-Computern wie Handhelds, Tablet-PCs usw. ist das Spektrum gleichzeitig darstellbarer Farben oft reduziert, häufig auf nur 256 Farben. Eine Palette legt fest, welche Farben das sind. Jedes System kann dabei seine eigene Default-Farbpalette einsetzen. Stimmt eine in CSS definierte Farbe nicht mit einer in der Palette verfügbaren Farbe überein, muss das System irgendeine Lösung finden, um den Konflikt zu lösen. Es könnte z.B. die nächstähnliche Farbe aus seiner Palette auswählen. Oder es könnte versuchen, durch so genanntes Dithering die gewünschte Farbe zu erzeugen. Dabei werden Pixelmuster aus zwei oder mehreren Farben gebildet, um die gewünschte Farbe einigermaßen echt darzustellen. Das Dithering ist jedoch je nach Pixelauflösung und Muster durchaus sichtbar und kann z.B. als Textfarbe angewendet die Lesbarkeit beeinträchtigen.

Aus diesem Grund hat sich schon früh eine Farbpalette durchgesetzt,

die praktisch alle Webbrowser in allen geeigneten Hard- und Softwareumgebungen anzeigen können. Die in dieser Palette definierten Farben gelten als **websichere** Farben. Die Tabelle listet alle 216 websicheren Farben auf.

Die Farben dieser Palette bestehen aus RGB-Farben mit den Dezimalwerten 0, 51, 102, 153, 204 oder 255 (hexadezimal: 00, 33, 66, 99, CC, FF). Alle daraus resultierbaren Farben sind erlaubt, also z.B. `rgb(51, 102, 255)` oder `#990000`. Da es sich um sechs mögliche Farbwerte handelt und drei Farbwerte jede Farbe bestimmen, sind dadurch insgesamt 6 × 6 × 6 = 216 Farben möglich. Die typische Farbtiefe von 256 Farben wird damit nicht ganz ausgereizt – es verbleiben 40 Farben, die z.B. für die Benutzeroberfläche des Browsers benötigt werden.

Wenn Sie Hintergrundbilder mit CSS einbinden, haben Sie nicht nur die Möglichkeit, die Grafik zu wählen, sondern das Verhalten des Browsers dahingehend zu beeinflussen, ob er das Hintergrundbild kacheln, als „Wasserzeichen“ einbinden oder in welchem Abstand zum Außenrand er das Bild positionieren soll. Verlassen Sie sich aber nicht darauf, dass alle Browser diese Angaben anstandslos interpretieren und korrekt darstellen.

Als Eigenschaft zum Referenzieren einer Hintergrundgrafik müssen Sie `background-image` einsetzen. Als Wert verwenden Sie wieder das Konstrukt url(datei- und pfadname), z.B.

```
body {
background-image:url(marble.jpg);
}
```

Im Hintergrund des HTML-Dokuments würde der Browser nun die Datei marble.jpg darstellen. Natürlich ist diese Eigenschaft nicht nur auf das `body`-Element beschränkt, Sie können es auch auf `table`, `p`, `h1` usw. anwenden.

```
h1 {
background-image:url(marble.jpg);
}
table {
background-image:url(marble.jpg);
}
p {
background-image:url(marble.jpg);
}
```

Je nach Größe der Grafik wird sie so lange wiederholt, bis der gesamte Bereich ausgefüllt ist. Dieses

Verhalten nennt sich Kacheln. Standardmäßig kachelt der Browser eine Grafik sowohl horizontal als auch vertikal. Und genau das lässt sich mit der Eigenschaft `background-repeat` beeinflussen. Dafür stellt CSS vier verschiedene Werte zur Verfügung:

- `repeat` Die Grafik wird horizontal und vertikal wiederholt.
- `repeat-x` Die Grafik wird nur horizontal (waagerecht) wiederholt.
- `repeat-y` Die Grafik wird nur vertikal (senkrecht) wiederholt.
- `no-repeat` Die Grafik wird nicht wiederholt, sondern nur einmal dargestellt.

```
body {
background-image:url(marble.jpg);
background-repeat:no-repeat;
}
table {
background-image:url(marble.jpg);
background-repeat:repeat-x;
}
p {
background-image:url(marble.jpg);
background-repeat:repeat-y;
}
```

Bei Inhalten, die über die verfügbare Darstellungsfläche hinausgehen, wird beim Verschieben des Ausschnitts (engl. scrolling) der Hintergrund mit verschoben. Auch dieses Verhalten kann mit der Eigenschaft `background-attachment` in zwei Stufen verändert werden.

- `scroll` Der Hintergrund wird mit dem Inhalt verschoben.
- `fixed` Der Hintergrund wird fixiert, und nur der Inhalt wird verschoben.

```
body {
background-image:url(marble.jpg);
background-attachment:fixed;
}
```

Bei diesem Beispiel würde also die Hintergrundgrafik an der gleichen Stelle im Fenster bleiben, während die darüber liegenden Elemente verschoben werden würden.

Mit der Eigenschaft `background-position` können Sie die Position des Hintergrundbilds bestimmen, und zwar sowohl horizontal auf der x-Achse als auch vertikal auf der y-Achse.

```
background-position:x y;
```

Als Werte für x und y können Sie numerische Angaben (außer Prozent) und Angaben mittels Schlüsselwort machen. Bei numerischen Angaben wird der Wert für x durch den Abstand vom linken Fensterrand und der Wert für y durch den Abstand zum oberen Fensterrand bestimmt. Als Schlüsselwörter können Sie die folgenden verwenden.

Für x:

- `left` Die Grafik wird horizontal linksbündig ausgerichtet.
- `center` Die Grafik wird horizontal zentriert ausgerichtet.
- `right` Die Grafik wird horizontal rechtsbündig ausgerichtet.

Und für y:

- `top` Die Grafik wird vertikal obenbündig ausgerichtet.
- `middle` Die Grafik wird vertikal mittig ausgerichtet.
- `bottom` Die Grafik wird vertikal untenbündig ausgerichtet.

Einige Beispiele:

```
background-position:10px 10px;
background-position:1cm 5cm;
background-position:center middle;
background-position:right bottom;
```

Das erste Beispiel positioniert die Hintergrundgrafik jeweils 10 Pixel von oben und links verschoben. Das zweite Beispiel verschiebt die Grafik um 1 cm nach rechts und 5 cm nach unten. Bei dem dritten Beispiel wird die Grafik horizontal und vertikal in der Mitte platziert. Das letzte Beispiel positioniert die Hintergrundgrafik dann unten rechts.

Alle eben vorgestellten Eigenschaften zur Verwendung von Hintergrundbildern können auch mit einer einzigen Eigenschaft zusammengefasst werden. Die dafür zu verfassen wendende Eigenschaft lautet `background`, die Reihenfolge der Angaben ist diesmal egal.

```
body {
background:url(marble.jpg) fixed repeat-x 10px 10px;
}
```

Nachfolgend ein Beispiel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Hintergrundgrafik -->
<html>
<head>
<title>Hintergrundbilder</title>
<style type="text/css">
body {
background-image:url(marble.jpg);
background-attachment:fixed;
}
h1 {
background-image:url(bricks-red.gif);
background-repeat:no-repeat;
background-position:right bottom;
}
p {
background-image:url(bricks-green.gif);
background-repeat:repeat-y;
}
</style>
</head>
<body>
<h1>Überschrift mit Hintergrundgrafik, die nicht wiederholt wird. Überschrift
mit Hintergrundgrafik, die nicht wiederholt wird.</h1>
<p>Textabsatz mit Hintergrundgrafik, die
<br>
nur vertikal wiederholt
<br>
wird. Textabsatz mit Hintergrundgrafik, die
<br>
nur vertikal wiederholt
<br>
wird.Textabsatz mit Hintergrundgrafik, die
<br>
nur vertikal wiederholt
<br>
wird.</p>
</body>
</html>
```

Als das World Wide Web Consortium (W3C) die Positionierung mit **CSS** einführte, dachten einige Designer verständlicherweise, sie könnten Webseiten erstellen, die wie gedruckte Dokumente aussähen (die mit Programmen wie PageMaker, InDesign oder Quark XPress erstellt werden). Tatsächlich reichen schon wenige CSS-Anweisungen aus, um ein **HTML**-Element präzise auf einer Webseite zu positionieren, z.B. 100 Pixel von der Oberkante und 200 Pixel von linken Seite des Ansichtsbereichs entfernt. Die pixe1genaue

Platzierung mit CSS-P (wie es damals genannt wurde) schien zu versprechen, dass man ein Design einfach dadurch erstellen könnte, dass man Fotos und Überschriften an der gewünschten Stelle positioniert.

Leider konnte CSS-P Designern die gewünschte präzise Kontrolle nie ganz bieten. Es gab immer Unterschiede in der Darstellung positionierter Elemente durch die verschiedenen Browser. Hinzu kommt, dass das Web einfach nicht funktioniert wie eine gedruckte Farbbroschüre, eine Zeitschrift oder ein Buch. Webseiten sind in der Darstellung viel flexibler als gedruckte Medien. Ist eine Zeitschrift einmal gedruckt, können die Leser die Schriftgröße nicht mehr anpassen. Die einzige Möglichkeit, das Aussehen noch zu verändern, besteht darin, Kaffee darüber auszukippen.

Die Besucher einer **Website** haben dagegen die Möglichkeit, mit Ihrer Präsentation herumzuspielen. Sie können beispielsweise die Schriftgröße des Browsers verändern, wodurch der Text möglicherweise nicht mehr in die genau platzierten und bemessenen Layoutelemente passt. Dennoch gibt es nicht nur schlechte Nachrichten: Solange Sie nicht versuchen, die genaue Höhe, Breite und Position aller Designelemente zu bestimmen, kann Ihnen die Positionierung von Elementen mit CSS eine große Hilfe sein. Sie können die entsprechenden Eigenschaften beispielsweise einsetzen, um Fotos mit Überschriften zu versehen, mehrspaltige Layouts zu erstellen, ein Logo an beliebiger Stelle auf der Seite zu platzieren und vieles mehr.

Wie CSS-Eigenschaften für die Positionierung funktionieren

Mithilfe der CSS-Eigenschaft `position` können Sie bestimmen, wie und wo ein Webbrowser bestimmte Seitenelemente anzeigt. So können Sie `position` beispielsweise verwenden, um eine Seitenleiste an beliebiger Stelle im Dokument zu platzieren oder um ein Navigationsmenü selbst dann im oberen Teil des Browserfensters anzuzeigen, wenn der Benutzer den Inhalt der Seite scrollt. In CSS gibt es vier Formen der Positionierung:

- **Absolute Positionierung**. Bei der absoluten Positionierung geben Sie an, wie weit ein Element vom Rand des Ansichtsbereichs entfernt sein soll. Dies geschieht, wie bei den anderen Arten der Positionierung auch, anhand der Eigenschaften `left`, `right`, `top` und `bottom`. Sie können die Entfernung mit allen in CSS gültigen Maßeinheiten angeben (Pixel, em-Einheiten, Prozentwerte usw.). Sie können ein Element z.B. 20 Pixel von der Oberkante (`top`) und 200 Pixel von der linken Seite (`left`) des Ansichtsbereichs platzieren, wie in Abbildung 1, Mitte, gezeigt.

Absolut positionierte Elemente sind komplett vom Textfluss der Seite ausgenommen, die normalerweise vom **HTML-Quellcode** vorgegeben wird. Das heißt, die anderen Dinge auf der Seite wissen nicht einmal, dass die absolut positionierten Elemente überhaupt existieren. Standardmäßig sitzen positionierte Elemente scheinbar auf dem übrigen Inhalt. Wenn Sie nicht aufpassen, werden die anderen Elemente von den absolut positionierten Teilen der Seite überlagert.

Info

Verwenden Sie in derselben CSS-Regel die Eigenschaft `float` ausschließlich zusammen mit der statischen oder relativen Positionierung (im Folgenden erklärt). `Floats` und `absolute` oder feste Positionierung können nicht zusammen für dasselbe Element benutzt werden.

- **Relative Positionierung**. Ein relativ positioniertes Element wird in Bezug zu seiner aktuellen Position im Textfluss der Seite verschoben. Befindet sich ein Element in der Seitenmitte (vertikal und horizontal), sorgen die Deklarationen `top: 20px;` und `left: 200px;` dafür, dass es von seiner aktuellen Position aus gesehen in Richtung unterer Seitenrand und nach rechts verschoben wird.

Im Gegensatz zur absoluten Positionierung bleibt der Platz im Textfluss für ein relativ positioniertes Element reserviert. Durch die Verschiebung entsteht an der Stelle, an der das Element eigentlich gestanden hätte, eine Lücke im Text. Das wird im unteren Bild in Abbildung 1 deutlich. Der dunkle Streifen zeigt an, wo die relativ positionierte Box eigentlich gestanden hätte, bevor sie verschoben wurde. Der Hauptvorteil der relativen Positionierung liegt allerdings nicht darin, ein Element zu verschieben. Vielmehr geht es darum, einen Referenzpunkt für darin enthaltene absolut positionierte Elemente zu haben.

- **Feste Positionierung**. Ein fest positioniertes Element wird an einer bestimmten Stelle im Browserfenster fixiert. Die Wirkung ist die gleiche wie der Wert `fixed` für die Eigenschaft `background-attachment`. Wenn ein Besucher den Inhalt der Seite herunterscrollt, bleiben fest positionierte Elemente (Absätze. überschriften usw.) weiterhin sichtbar, während die übrigen Dinge (Text, Fotos usw.) nach oben aus dem Browserfenster verschwinden.

Mithilfe von fest positionierten Elementen lässt sich hervorragend eine ständig sichtbare Navigationsleiste oder der Effekt von **HTML-Frames** realisieren, bei denen ein bestimmter Teil der Seite (`Frame`) gescrollt wird.

- **Statische Positionierung** bedeutet einfach nur, dass der Inhalt im normalen Textfluss des HTML-Codes dargestellt wird {Abbildung 1 oben}. Aber wann sollte man dann einem Element diese Art der Positionierung ausdrücklich zuweisen? Schließlich ist dies der Standardwert. Die kurze Antwort: vermutlich nie.

Info

Internet Explorer 6 und seine Vorgänger verstehen die Deklaration `position: fixed;` nicht, d.h., sie ignorieren diese Einstellung einfach. Da der IE 6 immer noch sehr häufig verwendet wird, kann der Einsatz einer festen Positionierung unter Umständen zu Problemen führen. Eine mögliche Lösung ist die Verwendung des IE 7-JavaScripts.

Um die Positionierung eines Elements zu ändern, nehmen Sie die CSS-Eigenschaft `position`. Als Wert wird eines der vier Schlüsselwörter `static`, `absolute`, `relative` und `fixed` verwendet. Um ein Element absolut zu positionieren, können Sie beispielsweise die folgende Deklaration nutzen:

```
position: absolute;
```

Standardmäßig wird für alle Elemente die statische Positionierung verwendet. Diese Form müssen Sie also nur angeben, wenn Sie die Einstellung für ein absolut, relativ oder fest positioniertes Element wieder zurücksetzen wollen. Hinzu kommt, dass statisch positionierte Elemente die im Folgenden diskutierten Werte für die Positionierung nicht beachten.

Die Definition der Positionierungsform ist also nur die halbe Miete. Um ein Element tatsächlich an einer bestimmten Stelle auf der Seite zu platzieren, müssen Sie sich auch mit den dazugehörigen speziellen Eigenschaften auskennen.

Die Positionierung steuern

Für jede Seite des Ansichtsbereichs {des sogenannte Viewports} im Browserfenster gibt es eine entsprechende CSS-Eigenschaft: `top` (oben), `bottom` (unten), `left` (links) und `right` {rechts}. Mithilfe dieser Eigenschaften geben Sie an, wie groß der Abstand zwischen der Seite des Ansichtsbereichs und dem Element sein soll. Hierbei ist es nicht nötig, für alle vier Seiten einen Abstand anzugeben. Normalerweise reichen zwei Werte aus, um ein Element auf der Seite zu platzieren. Wenn Sie wollen, können Sie ein Element 10 em-Einheiten von der linken Seite und 20 em-Einheiten von der Oberseite des Ansichtsbereichs platzieren.

Info

Im Internet Explorer kann es passieren, dass mit den Eigenschaften `bottom` oder `right` positionierte Elemente falsch platziert werden.

Die Entfernung vom Rand des Ansichtsbereichs bis zum Element kann in einer beliebigen in CSS gültigen Maßeinheit, beispielsweise Pixel, em-Einheiten, Prozentwerte usw., vorgenommen werden. Durch die Verwendung negativer Werte wie `left; -10px;` ist es möglich, ein Element teilweise oder ganz außerhalb der Seite (oder des umgebenden Elements) zu positionieren. Auf diese Weise lassen sich einige interessante visuelle Effekte erzielen, wie später in diesem Kapitel noch geschildert wird.

Nach der Angabe der Art der Positionierung mit `position` werden üblicherweise zwei Angaben zur Position selbst gemacht (mit `top`, `bottom`, `left` oder `right`). Soll ein Element schmaler sein als die verfügbare Breite (z.B. um eine schmale Seitenleiste zu erstellen), können Sie das auch hier mit der Eigenschaft `width` angeben. Um die Kopfzeile einer Seite beispielsweise in der linken oberen Ecke der Seite zu verankern, können Sie die folgende Regel verwenden:

```
#kopfteil {
position: absolute;
left: 100px;
top: 50px;
width: 760px;
}
```

Das Resultat dieser Stildefinition sehen Sie in Abbildung 2 oben.

Hier noch ein Beispiel: Sie können ein Element so positionieren, dass es immer einen bestimmten Abstand zur rechten Seite des Ansichtsbereichs hat. Hierfür benutzen Sie die Eigenschaft `right`. Bei der Verwendung dieser Eigenschaft misst der Browser die Entfernung von der rechten Seite des Ansichtsbereichs bis zur rechten Seite des Elements (Abbildung 2 Mitte). Um die Kopfzeile 100 Pixel von der rechten Seite entfernt zu positionieren, können Sie fast die gleiche Regel verwenden wie oben – nur wird hier `left` durch `right` ersetzt:

```
#kopfteil {
position: absolute;
right: 100px;
top: 50px;
width: 760px;
}
```

Da die Position anhand der rechten Seite des Ansichtsbereichs berechnet wird, wird die Kopfzeile bei einer Größenänderung des Browserfensters automatisch neu ausgerichtet, wie in Abbildung 2 zu sehen ist. Obwohl sich die Kopfzeile bewegt, bleibt ihre Entfernung zwischen der rechten Seite des Elements und der rechten Seite des Ansichtsbereichs immer gleich.

Info

Auch bei im Einsatz von Prozentwerten für `top` und `bottom` wird die tatsächliche Entfernung basierend auf der Breite des Browsertensters (oder eines positionierten Eltern-Elements, wie im folgenden Abschnitt gezeigt) berechnet.

Technisch gesehen, können Sie Werte sowohl für `left` als auch für `right` angeben und dann den Browser die Höhe des Elements berechnen lassen. Angenommen, der Haupttext soll 100 Pixel von der Oberkante und jeweils 100 Pixel von den Seiten des Ansichtsbereichs positioniert werden. Hierfür können Sie den Block `absolut` positionieren und die Werte für `top`, `left` und `right` auf je 100 Pixel festlegen. Damit sind die Seiten der Box jeweils 100 Pixel von den Seiten des Ansichtsbereichs entfernt (Abbildung 3 oben). Die genaue Breite der Box hängt in diesem Fall davon ab, wie breit das Browserfenster und damit der Ansichtsbereich ist. Die Entfernung zu den Seiten bleibt dabei jedoch gleich.

Und auch hier hat Internet Explorer Probleme mit der Darstellung (Abbildung 3 unten). Die linke Position wird korrekt dargestellt, während der rechte Wert einfach ignoriert wird. Solange der IE 6 noch benutzt wird, sollten Sie daher `left` und `right` nicht gleichzeitig benutzen und stattdessen einen Wert für `width` definieren, um die Breite eines `absolut` positionierten Elements zu bestimmen.

Die Eigenschaften `width` und `height`, die Sie bereits aus Kapitel 7 kennen, funktionieren auch bei positionierten Elementen. Um eine 50 x 50 Pixel große graue Box in der rechten oberen Ecke des Ansichtsbereichs zu positionieren, können Sie die folgende Regel verwenden:

```
.box {
position: absolute;
right: 0;
top: 0;
width: 50px;
height: 50px;
background-color: #333;
}
```

Sofern Sie nicht mit einer Grafik arbeiten, die eine feste Höhe hat, können Sie nicht sicher sein, wie hoch ein Element auf der Seite tatsächlich dargestellt wird. Wenn Sie die Höhe einer Seitenleiste ausdrücklich auf 200 Pixel festlegen (`height: 200px;`), kann es passieren, dass das Element „überläuft“. Und selbst wenn der Platz eigentlich ausreicht, ist es möglich, dass der Benutzer die Schriftgröße im Browser erhöht, wodurch der Text eventuell doch über die Grenzen des Elements hinausragt. Ist der Inhalt größer als die angegebene Höhe oder Breite, können recht seltsame Dinge passieren.

Einstein und die absolute Positionierung

Bisher haben wir in diesem Kapitel nur über die Positionierung eines Elements an einer bestimmten Stelle im Browserfenster gesprochen. Allerdings funktioniert die absolute Positionierung nicht immer auf diese Weise. Tatsächlich werden absolut positionierte Element relativ zu den Begrenzungen des nächsten positionierten Vorfahren-Elements platziert. Angenommen, Sie haben bereits ein absolut positioniertes Element (z.B. ein `<div>`, das 100 Pixel von der Oberkante des Ansichtsbereichs entfernt ist). Befindet sich darin ein weiteres absolut positioniertes Element, wird dieses relativ zu den Seiten des äußeren `<div>` angeordnet.

Im oberen Bild von Abbildung 4 wurde die hellgraue Box jeweils 5 em-Einheiten von der oberen und der linken Kante des Ansichtsbereichs positioniert.

Innerhalb der Box befindet sich außerdem ein `<div>`-Tag. Wenn Sie dieses Tag mit absoluter Positionierung versehen, wird es relativ zu seinem absolut positionierten Vorfahren-Element angeordnet. Die Deklaration `bottom: 0;` verankert das Element also nicht an der Unterkante des Browserfensters, sondern an der Unterkante seines Eltern-Elements. Das Gleiche gilt für den Wert der Eigenschaft `right` (Abbildung 4 unten).

Die Platzierung eines absolut positionierten Elements hängt immer auch davon ab, ob und wie die es umgebenden Elemente positioniert wurden. Hier eine Kurzfassung der Regeln:

- Ein absolut positioniertes Tag wird relativ zum Browserfenster angeordnet, wenn sich das Tag nicht in einem anderen positionierten Element (`absolut`, `relativ` oder `fest`) befindet.
- Ein absolut positioniertes Tag wird relativ zu den Seiten eines umgebenden Elements angeordnet, wenn es sich innerhalb eines anderen Elements befindet, für das eine absolute, relative oder feste Positionierung definiert wurde.

Wann (und wo) relative Positionierung sinnvoll ist

Die Platzierung eines Elements relativ zu einem anderen Tag hat einen Vorteil: Verändert das Tag seine Position in der Darstellung, bewegt sich das relativ platzierte Element mit. Angenommen, Sie wollen ein Bild, das innerhalb eines <h1>-Tags platziert wurde, rechts neben der Überschrift darstellen. Eine absolute Positionierung an einer ganz bestimmten Stelle im Ansichtsbereich ist hier keine gute Idee, denn eine Größenänderung des Browserfensters kann dafür sorgen, dass sich die Überschrift verschiebt. Das Bild bleibt dagegen an der ursprünglichen Stelle. Daher ist es ratsam, das Bild in diesem Fall relativ zum <h1>-Tag zu positionieren. Wenn sich die Überschrift verschiebt, wird das Bild mit verschoben (wie in den beiden unteren Bildern von Abbildung 5 zu sehen).

Info

Normalerweise ist es sinnvoller, die Eigenschaft `background-image` zu verwenden, wenn Sie ein Bild mit einer Überschrift verknüpfen wollen. Ist die Grafik allerdings höher als die Schrift im <h1>-Tag oder soll die Grafik außerhalb der Grenzen der Überschrift erscheinen (Abbildung 5, drittes Bild von oben), ist die hier beschriebene relative Positionierung das Werkzeug der Wahl.

Sie könnten hier die Deklaration `position: relative;` verwenden, aber das hat gewisse Nachteile. Wenn Sie ein Element mit dieser Einstellung anhand der Eigenschaften `left` und `top` positionieren, wird es basierend auf seiner Ursprungsposition im Textfluss verschoben. Dabei bleibt an der ursprünglichen Position allerdings eine unschöne Lücke (Abbildung 1 unten). Das ist in der Regel nicht erwünscht.

Ein besserer Weg besteht darin, einen neuen Positionierungskontext für die verschachtelten Tags zu erzeugen. So ist das am Anfang dieses Beispiels erwähnte <h1>-Tag ein Vorfahren-Element von <img>. Indem Sie den Wert der Eigenschaft position für das <h1>-Tag auf relative setzen, können Sie die enthaltenen Elemente absolut positionieren. Hierbei wird die Position jetzt nicht mehr im Verhältnis zum Ansichtsbereich, sondern zu den Seiten des <h1>-Tags berechnet. Der CSS-Code sieht folgendermaßen aus:

```
h1 {
```

```
position: relative;
}
h1 img {
position: absolute;
top: 0;
right: 0;
}
```

Dadurch dass Sie den Eigenschaften `top` und `right` den Wert 0 geben, wird das Bild nicht in die rechte obere Ecke des Ansichtsbereichs, sondern der Überschrift verschoben.

In CSS hat der Begriff „relativ“ offenbar eine andere Bedeutung als in der wirklichen Welt. Wenn Sie das <img>-Tag relativ zum <h1>-Tag positionieren wollen, würden Sie intuitiv die **Deklaration** `position: relative;` für das Bild verwenden. Tatsächlich erhält das Bild aber die absolute Positionierung, während das Tag, das als Referenz dienen soll (die Überschrift), als relative definiert wird. Als Eselsbrücke können Sie sich den Wert relative mit „relativ zu mir“ übersetzen.

Info

Da Sie die relative Positionierung oftmals einsetzen, um einfach nur einen neuen Positionierungskontext für enthaltene Tags zu schaffen, brauchen Sie hier keine Werte für die Eigenschaften `top`, `bottom`, `fett` oder `right` anzugeben. Die Deklaration `position: relative;` reicht für das <h1>-Tag in diesem Fall aus.

Elemente stapeln

Wie Sie in Abbildung 6 sehen können, scheint sich das absolut positionierte Element näher am Betrachter zu befinden. Es schwebt quasi „über“ der Website. Zusätzlich zur x-Achse (von links nach rechts) und der y-Achse (von oben nach unten) gibt es in CSS auch noch eine z-Achse (von vorne nach hinten), auf der die Elemente hintereinander angeordnet werden. Diese Reihenfolge steuern Sie mit der Eigenschaft `z-index`. Wenn Sie das Konzept der Ebenen aus Photoshop, Fireworks oder Adobe InDesign kennen, wissen Sie was hier gemeint ist. Die Eigenschaft `z-index` steuert die Reihenfolge, in der die Elemente auf der Seite angeordnet werden.

Wenn Sie sich die Webseite als Stück Papier vorstellen, entspricht ein absolut positioniertes Element einem Notizaufkleber. Wenn Sie zu viele Aufkleber verwenden, besteht natürlich die Gefahr, dass der eigentliche Inhalt der Seite irgendwann nicht mehr sichtbar ist.

Normalerweise folgt die Reihenfolge auf der z-Achse der Reihenfolge der Elemente im HTML-Quellcode. Auf einer Seite mit zwei absolut positionierten -Tags wird das , das im **HTML-Code** als Zweites

kommt, über dem ersten dargestellt. Mit der Eigenschaft z-index können Sie diese Reihenfolge aUerdings auch ändern. Als Wert übernimmt die Eigenschaft einen numerischen Wert. z.B.:

```
z-index: 3;
```

Je höher dieser Wert, desto weiter vorne („näher am Benutzer“) wird das entsprechende Element dargestellt. Wenn Sie beispielsweise drei absolut positionierte Bilder haben, die sich teilweise überlappen, wird das Bild mit dem höchsten z-Index vor den anderen dargestellt (s. Abbildung 6 oben). Eine Änderung am z-Index ändert auch die Reihenfolge, in der die Elemente hintereinander angeordnet werden (Abbildung 6 Mitte).

Info

Hierbei müssen die Werte für z-index nicht direkt aufeinanderfolgen. Die Werte 10, 20, 30 funktionieren genauso gut wie 1, 2, 3. Wenn Sie zwischen den Werten „Platz lassen“, gibt Ihnen das Spielraum für Elemente, die Sie vielleicht später noch hinzufügen wollen. Wenn Sie sicherstellen möchten, dass ein bestimmtes Element immer vor allen anderen erscheint, geben Sie ihm einfachen einen sehr hohen z-index-Wert, z.B. 1000.

Teile einer Seite verstecken

Häufig wird bei absolut positionierten Elementen auch die CSS-Eigenschaft `visibility` verwendet. Hiermit können Sie Teile einer Seite verstecken (oder wieder anzeigen). Das kann beispielsweise praktisch sein, um eine Beschriftung anzuzeigen, wenn der Besucher den Mauszeiger über ein Bild bewegt. Hierfür definieren Sie die Beschriftung beim Laden der Seite als „unsichtbar“. Erst wenn sich die Maus über das Element bewegt, wird die Beschriftung angezeigt (`visibility: visible`). In Abbildung 7 sehen Sie ein Beispiel.

Der Wert `hidden` für die Eigenschaft `visibility` hat eine gewisse Ähnlichkeit mit dem Wert `none` für die Eigenschaft `display`. Allerdings gibt es auch einen wesentlichen Unterschied. Bei der Verwendung von `display: none;` verschwindet das Element komplett aus der Darstellung der Seite. Benutzen Sie dagegen die Deklaration `visibility: hidden;`, wird zwar verhindert, dass der Browser das Element darstellt, der Platz, den das Element eigentlich einnehmen würde, bleibt jedoch frei. Bei absolut positionierten Elementen, die vom Textfluss der Seite ausgenommen sind, funktionieren beide Deklarationen gleich.

Meistens wird JavaScript verwendet, um ein Element zu verstecken und wieder zum Vorschein zu bringen. Anstelle der Programmiersprache können Sie aber auch einfach die Eigenschaft `visibility` verwenden (oder auch die Eigenschaft display), um das Element zu verstecken. Mithilfe der Pseudoklasse `:hover` können Sie es wieder hervorzaubern.

Info

Unter der Adresse psacake.com wird eine einfache CSS-basierte Methode beschrieben, mit der Sie zusätzliche Informationen anzeigen können, wenn der Mauszeiger sich über einem bestimmten Element befindet. Daneben gibt es natürlich auch viele **JavaScript**-Lösungen, zum Beispiel finden Sie bei SolarDreamStudios eine einfache Anleitung (qrayg.com).

Sinnvolle Strategien für die Positionierung

Wie bereits am Anfang dieses Kapitels erklärt, kann es zu Schwierigkeiten kommen, wenn Sie versuchen, alle Elemente einer Seite zu positionieren. Keinesfalls kann man alle Kombinationen von Browsern und Benutzereinstellungen vorhersehen. Daher sollten Sie die Positionierung nur verwenden, um eine kleine Anzahl von Elementen exakt zu platzieren.

In diesem Abschnitt lernen Sie die Details über die absolute Positionierung, mit der Sie kleine, aber visuell wichtige Details in Ihr Seitendesign einbauen können. Wir zeigen Ihnen, wie bestimmte Layoute1emente absolut positioniert werden können und wie Sie wichtige Teile der Seite an ihrem Platz fixieren können, während der übrige Inhalt gescrollt wird.

Positionierung innerhalb eines Elements

Eine der effektivsten Anwendungen der Positionierung besteht darin, kleine Dinge relativ zu anderen Elementen zu platzieren. Mit absoluter Positionierung können Sie eine ähnliche Ausrichtung erreichen wie mit Floats. In Abbildung 8, Bild 1, wirkt das Datum in der Überschrift übertrieben deutlich. Mit CSS können Sie es neu formatieren und in die rechte untere Ecke verschieben.

Um für das Datum einen eigenen Stil zu definieren, müssen Sie es mit -Tags umgeben. Dies ist die erste Wahl, wenn es darum geht, einen Textteil unabhängig vom Rest des Absatzes mit eigenen Formatierungsanweisungen zu versehen.

```
<h1><span class="datum">10. März 2009</span>CosmoFarmer an Google verkauft</h1>
```

Jetzt müssen wir nur noch die nötigen **CSS-Regeln** erstellen. Um das Datum verschieben zu können, muss zuerst das umgebende <h1>-Tag relativ positioniert werden. Danach wird das zu verschiebende Element, in diesem Fall das Datum, absolut positioniert. Hier der nötige CSS-Code für die untere Zeile in Abbildung 8, Bild 1:

```
h1 {
position: relative;
width: 100%;
border-bottom: 1px dashed #999999;
}
h1 span.datum {
position: absolute;
```

```
bottom: 0;
right: 0;
font-size: .5em;
background-color: #E9E9E9;
color: black;
padding: 2px 7px 0 7px;
}
```

Einige der oben verwendeten Eigenschaften, z.B. `border-bottom`, sind nur für das Aussehen zuständig. Durch die Deklaration `position: relative;` für die Überschrift haben Sie einen neuen Positionierungskontext für das -Tag (mit dem Datum) geschaffen. Dieses können Sie nun in der rechten unteren Ecke platzieren, indem Sie den Eigenschaften `bottom` und `right` den Wert 0 zuweisen.

Info

Internet Explorer 6 und seine Vorgänger haben Probleme mit den Eigenschaften `bottom` und `right`. Das Problem wird durch die Deklaration `width: 100%;` im Stil für das <h1>-Tag gelöst.

Ein Element außerhalb seiner Box darstellen

Mithilfe der Positionierung können Sie ein Element auch so platzieren, dass es aus seinem umgebenden Element herausragt. In Abbildung 8. Bild 2, ist im oberen Bild eine Überschrift mit einer Grafik zu sehen. Das <img>-Tag wurde innerhalb des <h1>-Tags platziert, sodass es als Teil der Überschrift erscheint. Mit absoluter Positionierung und negativen Werten für die Eigenschaften `top` und `left` können Sie das Bild auf die linke Seite der Überschrift und über dessen obere und linke Kante hinausschieben. Hier der CSS-Code für das Beispiel:

```
h1 {
position: relative;
margin-top: 35px;
padding-left: 55px;
border-bottom: 1px dashed #999999;
}
h1 img {
position: absolute;
top: -30px;
left: -30px;
}
```

Wir verwenden hierbei das gleiche Konzept wie im vorigen Beispiel, allerdings mit ein paar Erweiterungen.

So benutzen wir hier negative Werte für die Eigenschaften `top` und `left`. Dadurch erscheint die Grafik 30 Pixel oberhalb und links von der Überschrift und 30 Pixel links von ihrer linken Kante. Bei der Verwendung von negativen Werten sollten Sie vorsichtig sein. Es kann leicht passieren, dass ein Element teilweise oder komplett aus dem sichtbaren Bereich des Browserfensters verschoben wird. Das können Sie verhindern, indem Sie dem <body>-Tag oder dem umgebenden relativ positionierten Element (hier das <h1>-Tag) ausreichende Außen- oder Innenabstände zuweisen. Hierdurch schaffen Sie genügend Platz für das überstehende Bild. Damit das Bild in unserem Beispiel den Inhalt oberhalb der Überschrift nicht überlagert, muss ausreichend oberer Außenabstand vorhanden sein. Der linke Innenabstand von 55 Pixeln verschiebt den Text so weit nach rechts, das dieser ebenfalls nicht mehr vom absolut positionierten Bild überlagert wird.

Wie im vorigen Beispiel macht hier der Internet Explorer erneut Schwierigkeiten. Diesmal reicht auch die Deklaration `width: 100%;` nicht aus. Durch den Innenabstand wird das <h1> im Internet Explorer tatsächlich breiter als 100 Prozent der Seite dargestellt. Eine mögliche Lösung ist die Verwendung der proprietären (Microsoft-eigenen) Eigenschaft `zoom`. Erweitern Sie die <h1>-Regel in diesem Fall einfach um die Deklaration `zoom: 1;`:

```
h1 {
position: relative;
margin-top: 3Spx;
padding-left: 55px;
border-bottom: 1px dashed #999999;
zoom: 1;
}
```

Info

Auf andere Browser hat `zoom` keine Auswirkungen, allerdings verhindert diese Eigenschaft, dass Ihr **CSS-Code** korrekt validiert. Mithilfe von IEs „Conditional Comments“ können Sie die Eigenschaft verstecken. Eine noch bessere lösung besteht allerdings darin, den Code für den IE in einem externen Stylesheet zu speichern, das per Conditional Comments eingebunden wird.

CSS-Positionierung für das Seitenlayout verwenden

Wie bereits gesagt, kann es schnell frustrierend werden, wenn Sie auch das letzte Element pixelgenau positionieren wollen. Mithilfe von absoluter Positionierung lassen sich viele Standardlayouts erstellen. In diesem Abschnitt lernen Sie, wie ein dreispaltiges flüssiges Layout mithilfe absoluter Positionierung erstellt werden kann. Die Seite besteht aus einem Kopfteil, je einer linken und einer rechten Seitenleiste, einem Bereich für den Hauptinhalt und einer Fußzeile für Copyright-Hinweise.

Info

In diesem Abschnitt zeigen wir Ihnen einen allgemeinen Ansatz für die absolute Positionierung den Sie für fast jedes Layout verwenden können.

Bei der Arbeit mit absoluter Positionierung sollten Sie die folgende Faustregel immer beherzigen: Versuchen Sie nicht, alles zu positionieren. Für ein gutes Layout müssen normalerweise nur ein paar Seitenelemente absolut positioniert werden.

Hier eine einfache Technik, mit der Sie herausbekommen, welche Elemente positioniert werden müssen. Angenommen, Sie müssen ein dreispaltiges Design wie das in Abbildung 9 rechts, erstellen. Stellen Sie zuerst fest, wie die einzelnen Teile der Seite ohne CSS-Positionierung dem normalen Textfluss des HTML-Codes folgen {Abbildung 9 links}. Dann fragen Sie sich für jedes Layoutelement der Seite: „Würde sich dieses Element auch ohne Positionierung an der richtigen Stelle befinden?“

Hier ein Überblick über die Elemente der Seitenelemente im linken Bild von Abbildung 9:

- **Der Kopfteil**. Der Kopfteil (1) befindet sich wunschgemäß am Anfang der Seite, es wird also keine absolute Positionierung gebraucht. Mithilfe von Innen- und Außenabständen können Sie den Inhalt trotzdem ein wenig herumschieben (etwa um oberhalb oder links davon etwas Leerraum zu schaffen).
- **Der `<div>`-Container** für den Hauptinhalt der Seite. Dieses `<div>`-Tag ist ein spezielles Element, das alle anderen Elemente der Seite enthält (2). Da es den Hauptinhalt der Seite umgibt, sollten Sie sich fragen, ob der Hauptinhalt unterhalb des Kopfteils dargestellt werden soll. Da ist der Fall, also wird auch hier keine absolute Positionierung benötigt.

Info

Im `<div>`-Container für den Hauptinhalt werden Seitenelemente, z.B. Seitenleisten, positioniert.

- **Der Hauptinhalt**. Der Hauptteil dieser Seite befindet sich ebenfalls direkt unterhalb des Kopfteils (3). Dieser muss auf der linken und rechten Seite eingerückt werden, damit genügend Platz für die Seitenleisten bleibt. Auch hierfür wird allerdings keine absolute Positionierung gebraucht.
- **Die linke Seitenleiste**. In Abbildung 9 links, erscheint erst gegen Ende der Quelleode, also unterhalb des Hauptinhalts (4). Dort sollte die Seitenleiste ganz bestimmt stehen. Sie benötigen hier absolute Positionierung, um den Abschnitt an die richtige Stelle zu rücken.
- **Die rechte Seitenleiste**. Trotz ihres Namens erscheint diese Seitenleiste (5) ohne weitere Hilfe fast am Ende der Seite, noch unterhalb der linken Seitenleisten. Auch hier brauchen Sie absolute Positionierung, um das Element an der richtigen Stelle zu platzieren: unterhalb des Kopfteils und rechts neben dem Hauptteil der Seite.
- **Die Fußzeile**. Im linken Bild erscheint die Fußzeile am Ende der Seite (6), also genau da, wo sie hingehört. Es wird also keine spezielle Positionierung gebraucht.

Info

Normalerweise ist es keine gute Idee, eine Fußzeile mithilfe von absoluter Positionierung an der unteren

Kante des Ansichtsbereichs zu fixieren. Ist der Inhalt der Seite länger als die Höhe des Browserfensters, wird auch die Fußzeile mit nach oben gesaollt. Eine bessere Lösung ist, hier die feste Positionierung zu verwenden.

Da Sie jetzt wissen, wann die CSS-Positionierung im Design nötig ist, finden Sie im Folgenden einen Überblick über die Erstellung eines dreispaltigen Layouts:

- 1. Umgeben Sie jeden logischen Abschnitt der Seite mit eigenen `<div>`-Tags.

Mit diesen Tags können Sie den inhalt der Seite in verschiedene Layout-Container fur den Kopfteil, Seitenleisten usw. unterteilen. Wie bei den im vorigen Kapitel besprochenen Floats versehen Sie auch hier die Container mit IDs (z.B. `<div id= "kopfteil">`), damit Sie für jeden Teil der Seite eigene **CSS-Stile** definieren können.

Die linke Grafik in Abbildung 9 zeigt, in welcher Reihenfolge die verschiedenen `<div>`-Tags im HTML-Code erscheinen. Ein Vorteil der absoluten Positionierung besteht darin, dass Sie sich (im Gegensatz zur Verwendung von Floats} keine Sorgen um die Reihenfolge der `<div>`-Tags im HTML-Code zu machen brauchen. Der Code für absolut positionierte Elemente kann an beliebiger Stelle in der Datei stehen. Er muss nur zwischen den `<body>` ... `</body>`-Tags stehen. Tatsächlich sind es die Positionierungseigenschaften, die festlegen, wo ein absolut positioniertes Element tatsächlich auf dem Bildschirm angezeigt wird, und nicht die Reihenfolge im HTML-Code.

- 2. Umgeben Sie sämtlichen HTML-Code für Hauptinhalt, Seitenleisten und Fußzeile mit einem weiteren `<div>`-Tag.

Das `<div>`-Tag (Abbildung 9, Bild 2) fasst diese Teile in einem gemeinsamen Container zusammen. Geben Sie ihm eine ID, damit Sie ihm eigene Stile zuweisen können (z.B. `<div id="inhaltscontainer">`). Dieses `<div>`-Tag stellt den Kontext für die Positionierung der Seitenleisten bereit, wie Sie im nächsten Schritt sehen werden.

- 3. Geben Sie dem `<div>`-Container eine relative Position.

Verwenden Sie die Eigenschaft position mit dem Wert `relative`, um die folgende Regel zu erstellen:

```
#inhaltscontainer {
position: relative;
}
```

Wenn Sie einem relativ positionierten Element keine Werte für die Eigenschaften `top`, `left`, `bottom` oder `right` zuweisen, behält es seine ursprüngliche Position bei, in diesem Fall unterhalb des Kopfteils. Allerdings verändert die relative Positionierung des `<div>`-Containers die Positionierung der enthaltenen Elemente. Wenn Sie im nächsten Schritt die Seitenleisten mittels absoluter Positionierung an die richtige Stelle schieben, beziehen sich die Werte für `top` usw. nicht mehr auf den Ansichtsbereich, sondern auf den `<div>`-Container. Sollten Sie Änderungen am Kopfteil der Seite vornehmen, wird der Container inklusive

seines Inhalts nach unten verschoben. Ohne den Container hätten Sie in diesem Fall die Werte für `top` der Seitenleisten anpassen müssen, um die Änderungen zu berücksichtigen.

- 4. Positionieren Sie die Seitenleisten `absolut`.

Da sich die übrigen Inhalte der Seite bereits durch ihre Position im Quellcode an der richtigen Stelle befinden, brauchen Sie nur noch die Seitenleisten zu positionieren. Weil die Seitenleisten relativ zum in Schritt 3 angelegten `<div>`-Container positioniert werden, brauchen Sie nur die Eigenschaften `top` und `left` für die linke bzw. `top` und `right` für die rechte Seitenleiste auf den Wert 0 zu setzen.

```
#linkeSeitenleiste {
position: absolute;
left: 0;
top: 0;
width: 175px;
}
#rechteSeitenleiste {
position: absolute;
right: 0;
top: 0;
width: 180px;
}
```

Info

Sie können die Werte für `top`, `right` und `left` ganz Ihrem persönlichen Geschmack anpassen. Um die Seitenleiste einzurücken, brauchen Sie nur die Werte der Eigenschaften zu erhöhen (z.B. auf 10px oder 0.9em oder einen anderen Wert, der Ihnen richtig erscheint).

- 5. Geben Sie für jede Seitenleiste eine Breite (in einer der für CSS gültigen Maßeinheiten) an.

Die Breitenangaben beschränken den von den Seitenleisten beanspruchten Platz. Ohne diese Angabe nehmen die Seitenleisten so viel Platz wie möglich in Anspruch, wodurch kein Raum mehr für den Hauptinhalt bleibt.

- 6. Passen Sie die Außenabstände für den Hauptinhalt an.

Da die Seitenleisten `absolut` positioniert wurden, sind sie vom normalen Textfluss der Seite ausgenommen. Der Hauptinhalt weiß nicht einmal, dass sie existieren, und nimmt einfach die gesamte Breite des Ansichtsbereichs ein. Da sich der Hauptinhalt innerhalb der Seite bereits an der richtigen Stelle befindet (unterhalb des Kopfteils), brauchen Sie diesen nicht neu zu positionieren. Es reicht aus, an den Seiten etwas Platz zu schaffen, damit es keine Überschneidungen mit den Seitenleisten gibt.

Das erreichen Sie, indem Sie für das `<div>`-Tag mit dem Hauptinhalt den linken und rechten Außenabstand erhöhen. Der Wert für `margin-left` bzw. `margin-right` sollte mindestens der Breite der

jeweiligen Seitenleiste entsprechen. Etwas breiter ist sogar noch besser:

```
#hauptinhalt {
margin-left: 185px;
margin-right: 190px;
}
```

In diesem Code sind die Außenabstände etwas größer als die Breiten der Seitenleisten. Normalerweise sollten Sie etwas mehr Platz lassen als nötig, um die einzelnen Elemente der Seite visuell besser voneinander zu trennen.

So schön dieses Layout auch ist, es besitzt eine Achillesferse. Wenn Sie die Seitenleisten absolut positionieren, kann es passieren, dass sie über die Fußzeile oder andere nachfolgende **HTML-Elemente** hinausragen und diese verdecken (s. Abbildung 10 oben rechts und unten). Im Gegensatz zu Float-basierten Layouts kann hier nicht die Eigenschaft `float` oder etwas Ähnliches verwendet werden. Am besten, Sie umgehen das Problem. wie im folgenden Schritt erklärt.

- 7. Verwenden Sie gegebenenfalls Außenabstände für die Fußzeile, damit sie nicht verdeckt wird.

Alternativ dazu können Sie auch dafür sorgen, dass die absolut positionierten Spalten auf keinen Fall höher sind als der Hauptinhalt. Wenn der Hauptinhalt hoch genug ist, schiebt er die Fußzeile automatisch unter die Spalten, und das Problem wird vermieden.

Info

Eine JavaScript-basierte Lösung für dieses Problem finden Sie unter der Adresse shauninman.com.

Diese Grundtechnik lässt sich auf verschiedenste Weisen variieren. Für ein zweispaltiges Layout reicht es, die rechte Seitenleiste (rechteSeitenleiste) aus dem HTML-Code zu entfernen und den rechten Außenabstand (`margin-right`) für das `<div>`-Tag des Hauptinhalts wieder auf null zu setzen. Oder Sie entfernen die linke Seitenleiste und erhalten wiederum ein zweispaltiges Layout, bei dem die schmalere Spalte auf der rechte Seite angezeigt wird. Das Grundlayout lässt sich übrigens auch mit festen Breiten realisieren. Hierfür reicht es aus, für den Kopfteil und den `<div>`-Container einen entsprechenden Wert zu definieren, wie hier:

```
#kopfteil, #inhaltscontainer {
width: 760px;
}
```

Frames mit CSS und fester Positionierung

Da die meisten Webseiten länger sind als der Ansichtsbereich hoch ist, sollen vielleicht einige Seitenelemente prinzipiell sichtbar sein, z.B. eine Navigationsleiste, ein Suchfenster oder das Logo Ihrer Site. Wollte man das ohne CSS-Positionierung erreichen, waren HTML-Frames die einzige Lösung. Frames haben jedoch große Nachteile. Für jeden Frame musste eine separate **HTML-Datei** angelegt werden. Um eine einzige Webseite zu erstellen, waren also prinzipiell mehrere Dokumente im Spiel (das Frameset). Dabei war nicht nur die Erstellung der Framesets zeitaufwendig, sie erschwerten es den Suchmaschinen außerdem, die Seite zu indexieren. Besuchern, die aufgrund von Sehproblemen ein Bildschirmlesegerät verwenden oder die die Seite ausdrucken wollen, machen Frames besonders große Schwierigkeiten.

Dennoch kann das Konzept der Frames weiterhin nützlich sein. Daher gibt es in CSS einen Wert für die Positionierung, mit der die Darstellungsweise von Frames mit deutlich weniger Aufwand imitiert werden kann. Eine mit dem Wert `fixed` erstellte Seite sehen Sie in Abbildung 11.

Info

Wie bereits gesagt, funktioniert die feste Positionierung erst ab Internet Explorer 7. Mit etwas zusätzlichem CSS-Code können Sie aber immerhin dafür sorgen, dass die Seite auch im IE 6 gut aussieht (auch wenn die „festen“ Elemente dabei mit dem übrigen Inhalt gescrollt werden.) Und da seit IE 7 die feste Positionierung unterstützt wird, ist es nur noch eine Frage der Zeit, bis alle Besucher Ihrer Sete ein ähnliches Ergebnis zu sehen bekommen.

Feste und absolute Positionierung funktionieren auf ähnliche Weise. Bei beiden verwenden Sie die Eigenschaften `left`, `top`, `right` oder `bottom`, um ein Element auf der Seite zu platzieren. Auch bei der festen Positionierung werden positionierte Elemente vom normalen Textfluss der Seite ausgenommen. Sie werden quasi „über“ den anderen Teilen der Seite dargestellt, ohne diese weiter zu beachten.

In den folgenden Schritten wird erklärt, wie Sie eine Seite wie die in Abbildung 11 erstellen können. Kopf- und Fußteil sowie die Seitenleisten wurden fest positioniert:

- 1. Unterteilen Sie die Seite mithilfe von -Tags mit ID-Attributen in logische Abschnitte.

Für dieses Beispiel sollten vier -Tags ausreichen, um die verschiedenen Bereiche zu markieren: kopfteil, seitenleiste, hauptteil und Jusszeile (siehe Abbildung 12). Die Reihenfolge der Elemente im HTML-Code spielt hier keine Rolle. Wie bei der absoluten Positionierung können Sie auch fest positionierte Elemente an beliebiger Stelle im Browserfenster platzieren.

Info

Es gibt eine Ausnahme: Damit die Seite auch für die Benutzer des Internet Explorer 6 einigermaßen aussieht, sollte die Fußzeile unterhalb des HTML-Codes für den Hauptteil stehen, wie in Schritt 5 gezeigt.

- 2. Füllen Sie die `<div>`-Tags mit dem gewünschten Inhalt.

Normalerweise verwenden Sie die fest positionierten `<div>`-Tags für Dinge, auf die Benutzer jederzeit Zugriff haben sollen. In diesem Beispiel enthalten Kopfteil und Fußzeile sowie die Seitenleiste das Logo der Site, die Navigation und Copyright-Hinweise. Der Hauptinhalt steht im verbleibenden `<div>`-Tag.

Sie sollten darauf achten. ein fest positioniertes `<div>`-Tag nicht mit zu viel Inhalt zu füllen. Ist eine fest positionierte Seitenleiste länger als das Browserfenster, wird sie nur teilweise angezeigt. Und weil fest positionierte Elemente keine Rollbalken besitzen (nicht einmal, wenn Sie es mit der Deklaration `overflow: scroll;` probieren), kann der fehlende Inhalt nur dargestellt werden, wenn der Benutzer einen größeren Monitor kauft.

- 3. Legen Sie die nötigen Stilregeln für die fest positionierten Elemente an.

Die Werte der Eigenschaften `left`, `right`, `top` und `bottom` beziehen sich auf das Browserfenster. Sie brauchen also nur zu überlegen, an welcher Stelle im Ansichtsbereich die Elemente stehen sollen, und die entsprechenden Werte einzugeben. Vergessen Sie nicht, auch eine Breite zu bestimmen.

Info

Im Gegensatz zur absoluten Positionierung bezieht sich die feste Positionierung immer auf das Browserfenster (oder genauer, den darin sichtbaren Bereich). Das gilt selbst dann, wenn sich ein fest positioniertes Element in einem Tag befindet, das `relativ` positioniert wurde.

Hier die Regeln, um die Elemente 1, 3 und 4 aus Abbildung 12 so aussehen zu lassen wie in der Abbildung:

```
#kopfteil {
position: fixed;
left: 0;
top: 0;
width: 100%;
}
#seitenleiste {
position: fixed;
left: 0;
top: 110px;
width: 175px;
}
#fusszeile {
```

```
position: fixed;
bottom: 0;
left: 0;
width: 100%;
```

- 4. Erstellen Sie die Stildefinition für den scrollbaren Inhaltsbereich.

Da fest positionierte Elemente vom normalen Textfluss der Seite ausgenommen sind, haben die übrigen **HTML-Tags** keine Ahnung, wo sich die positionierten Elemente befinden. Im Ergebnis scheint sich das `<div>`-Tag mit dem Hauptinhalt der Seite „hinter“ den fixierten Elementen zu befinden. Die wichtigste Aufgabe dieser Regel besteht deshalb darin, mithilfe von Außenabständen diese Bereiche frei zu halten und überschneidungen zu verhindern.

```
#hauptteil {
margin-left: 190px;
margin-top: 110px;
}
```

- 5. Passen Sie das Layout für Internet Explorer 6 und seine Vorgänger an.

Der IE 6 versteht keine feste Positionierung. Fest positionierte Elemente werden behandelt, als hätten Sie den Wert `static`. Das heißt, die einzelnen Bestandteile werden im IE 6 gemäß ihrer Reihenfolge im HTML-Code der Seite dargestellt. Vielleicht wird dadurch einfach nur ein sehr großer Leerraum zwischen dem Kopfteil und der Seitenleiste angezeigt, oder, noch schlimmer, die Navigationsleiste und der Kopfteil werden unterhalb des Hauptinhalts dargestellt.

Der Trick besteht darin, den IE 6 anzuweisen, die fest positionierten Elemente absolut zu positionieren. Hierdurch werden sie vom normalen Textfluss der Seite ausgenommen und an der richtigen Stelle im Browserfenster platziert.

```
* html #kopfteil {
position: absolute;
}
* html #seitenleiste {
position: absolute;
}
```

Info

Die hier gezeigten Regeln verwenden den Star-**HTML-Hack**, um die Definitionen vor anderen Browsern als IE 6 und seinen Vorgängern zu verstecken. Alternativ können Sie auch mit „Conditional Comments“ arbeiten.

Wie Sie vielleicht bemerkt haben, gibt es keinen eigenen #fusszeile-Stil. Wenn Sie die Fußzeile ebenfalls absolut positionieren, wird sie beim Scrollen über dem Hauptinhalt dargestellt und mitgescrollt.

In diesem Fall ist es am besten, wenn die Fußzeile einfach am unteren Ende der Seite angezeigt wird und erst dann zu sehen ist, wenn der Inhalt nach unten gescrollt wurde. (Hier haben wir auch den Grund dafür, dass die Fußzeile nach dem HTML-Code für den Hauptinhalt stehen sollte. Auf diese Weise erscheint sie im IE 6 zumindest unterhalb des Hauptinhalts.)

Diese Technik ist kein vollwertiger Ersatz für die feste Positionierung. Immerhin werden Kopfteil und Seitenleiste beim Laden der Seite korrekt platziert. Allerdings werden diese Teile, wie auch die Fußzeile, mit dem übrigen Inhalt gescrollt. Die Seite funktioniert im IE 6 also wie jede normale Webseite mit absoluter Positionierung. Ab IE 7, in Firefox, Safari und Opera geht das sogar noch besser.

Wenn Sie in CSS numerische Werte für Breiten, Höhen, Abstände usw. definieren, müssen Sie eine Maßeinheit angeben.

Maß	Bedeutung	Typ
`pt`	point (zu Deutsch: Punkt). Typografische Maßeinheit. 1pt entspricht ca. 1.39in.	absolut
`pc`	pica. Typografische Maßeinheit. 1pc entspricht 12pt.	absolut
`in`	inch. Allgemeines Längenmaß im angelsächsischen Raum.1in entspricht 2.54cm.	absolut
`mm`	millimeter. 1mm entspricht 0.1cm.	absolut
`cm`	centimeter. Allgemeines Längenmaß.	absolut
`px`	pixel. Abhängig von der Pixeldichte des Ausgabegeräts, relativ also von Ausgabegerät zu Ausgabegerät, absolut dagegen auf ein und dasselbe Ausgabegerät bezogen.	relativ
`em`	element. Schriftgröße des relevanten Elements.	relativ
`%`	percent. Prozentangabe.	relativ

Info

Generell wird zwischen absoluten Maßangaben und relativen Maßangaben unterschieden. So ist

beispielsweise `cm` (Zentimeter) eine absolute Angabe, während `em` (element-eigene Schriftgröße) eine relative Angabe darstellt. Hardliner empfehlen immer wieder, man solle ausschließlich relative Angaben verwenden, da man als Webdesigner niemals wissen kann, welche Art von Ausgabegerät ein Anwender verwendet oder wie groß sein Browser-Fenster ist. Für die Praxis empfiehlt sich kein völliger Verzicht auf absolute Angaben, jedoch ein behutsamer Umgang damit. Für reine Bildschirmpräsentationen wird in der Praxis am häufigsten mit der `px`-Bemaßung gearbeitet, also mit Pixelangaben, sowie mit Prozentangaben. Angaben wie `pt`, `pc`, `in`, `mm` und `cm` eignen sich dagegen eher für Print-Ausgaben von CSS-formatierten HTML-Dokumenten. Die Angaben `em` und `ex` schließlich sind besonders zum relativen Abstimmen von Schriftgrößen geeignet.

Regeln für numerische Werte

Folgende Regeln müssen Sie bei numerischen Werten in CSS beachten:

- Das Dezimalzeichen wird in der englisch/amerikanischen Form notiert, d.h. als Punkt, nicht das Komma. Tausenderpunkte und Tausenderkommas sind nicht zulässig.
- Manche CSS-Eigenschaften erlauben auch die Angabe negativer numerischer Werte. Solche Werte werden wie üblich durch ein voranstehendes Minuszeichen gekennzeichnet.
- Bei Zuweisung des Werts 0 für Längen, Breiten, Abstände und Größen ist keine Maßangabe zwingend erforderlich, bei allen anderen Werten dagegen schon.

Ein paar gültige Beispiele:

- `width:300px` definiert eine Breite von 300 Pixel.
- `margin-top:0.7cm` definiert einen Abstand oben von 0,7 Zentimetern.
- `font-size:0.95em` definiert eine Schriftgröße von 95%, gemessen an der Schriftgröße des Elternelements.
- `border-width:0.06in` definiert eine Rahmendicke von ca. 0,15 Zentimetern.
- `max-width:90%` legt fest, das ein Element maximal 90% Breite seiner verfügbaren Breite einnehmen darf.

Mit den bisher vorgestellten Möglichkeiten von CSS beeinflussen wir nicht den normalen Textfluss eines HTML-Dokuments, im CSS-Fachjargon als **normal flow** bezeichnet. In diesem Abschnitt werden wir nun unter anderem CSS-Eigenschaften kennen lernen, die den normalen Textfluss außer Kraft setzen oder manipulieren. Dadurch ergeben sich Möglichkeiten, die vor allem beim Design von Seitenlayouts mit CSS zum Tragen kommen.

Kontrolle von Breite und Höhe

Block-Elemente im normalen Textfluss nehmen so viel Breite ein wie möglich und so viel Höhe wie nötig. Dieses Default-Verhalten lässt sich auf mehrfache Weise manipulieren:

- Breite und Höhe können vorgegeben werden.

- Spielraum kann durch Angabe von Mindestbreite, Maximalbreite, Mindesthöhe und Maximalhöhe definiert werden.
- Für Inhalte, die größer sind als die Vorgaben zu Breite und/oder Höhe, lässt sich eine Konfliktlösung festlegen (z.B. Inhalte einfach abschneiden oder Inhalte scrollen).
- In Verbindung mit anderen Angaben lassen sich Elemente mit definierter Breite ausrichten. Breite und Höhe eines Block-Elements können durch die CSS-Eigenschaften `width` (Breite) und `height` (Höhe) festgelegt werden.

```
<p style="width:300px; height:150px; background-color:aqua"></p>
```

Info

Das normale Elementverhalten wird dadurch aufgehoben. Selbst wenn das Element keinerlei Inhalt hat, so wie im Beispiel, wird eine Box mit den für Breite und Höhe angegebenen Ausmaßen dargestellt:

Die angegebene Hintergrundfarbe macht die Box des p-Elements sichtbar.

> Füllt man eine solche Box mit Inhalt,

werden die Angaben zu `width` und `height` jedoch nur so lange eingehalten, wie der Inhalt hineinpasst. Ist der Inhalt umfangreicher, wird die Box ausgedehnt. Reiner Text beispielsweise passt in der Breite durchaus in die Box, aber wenn er in der Länge mehr Raum benötigt, als Höhe angegeben ist, wird die Box in der Höhe einfach ausgedehnt und angepasst. Das Gleiche passiert mit der Breite, falls der Inhalt nicht so umgebrochen werden kann, dass er die Breite einhält. Das ist beispielsweise der Fall, wenn der Inhalt aus einer Grafik besteht, deren Breite größer ist, oder wenn der Inhalt eine Zeichenfolge ohne Whitespace-Zeichen enthält, die mehr Breite einnimmt.

Dies ist jedoch ebenfalls nur ein Default-Verhalten, das Sie ändern können. Dazu dient die Eigenschaft `overflow`.

Durch Angabe von `overflow:hidden` erreichen Sie, dass die Boxgrößen von `width` und `height` eingehalten werden und dass übergroßer Inhalt einfach abgeschnitten wird.

Für manche Fälle interessant ist jedoch auch das Angebot an den Anwender, übergroße Inhalte zu scrollen. Dazu dient die Angabe `overflow:scroll`.

> Die Angaben zu Breite und Höhe können auch prozentual sein.

Interpretiert werden sie dann jedoch nur, wenn es ein Elternelement gibt, dessen absolut festgelegte oder errechnete Breite den Maßstab für die Prozentangabe bildet. Wenn Sie also beispielsweise einem p-Element im normalen Textfluss `width:70%` zuweisen, dann gilt die Breite des `body`-Elements, innerhalb dessen das p-Element vorkommt, als Bezug. Kommt das p-Element dagegen innerhalb eines `div`-Bereichs vor, dessen Breite mit `width:400px` festgelegt wurde, dann beziehen sich die 70% auf diese Breite.

Sie können auch Werte für `min-width` bzw. `max-width` und `min-height` bzw. `max-height` notieren. Wenn die angegebene oder verfügbare Breite größer ist als `min-width`, hat diese Eigenschaft gar keine Wirkung – das Element wird dann so breit wie angegeben oder möglich. Eine Wirkung hat die Angabe nur dann, wenn die angegebene oder verfügbare Breite kleiner ist. Analog verhält es sich bei `min-height`. Leider werden die Mindest- und Maximalangaben zu Breite und Höhe nicht von allen Browsern interpretiert. Von daher ist bei ihrer Verwendung Vorsicht angebracht.

Absolute, relative und fixe Positionierung

Durch Angaben zur Positionierung von Elementen heben Sie den normalen Textfluss wie folgt auf:

- Relative Positionierung positioniert eine Elementbox relativ zu der Position, die sie im normalen Textfluss hat.
- Absolute Positionierung positioniert eine Elementbox entweder relativ zu einem ebenfalls positionierten Elternelement, oder, falls es kein positioniertes Elternelement gibt, absolut im Dokument.
- Fixe Positionierung positioniert eine Elementbox in jedem Fall absolut im Dokument, und zwar so, dass das Element beim Scrollen nicht mitscrollt.

Zunächst ein Beispiel zur relativen Positionierung:

```
<p style="background-color:aqua">ABSATZ</p>
<p style="background-color:aqua">ABSATZ</p>
<p style="background-color:aqua; position:relative; left:100px;
top:8px">ABSATZ</p>
<p style="background-color:aqua">ABSATZ</p>
<p style="background-color:aqua">ABSATZ</p>
```

Info

Zur besseren Visualisierung des Effekts, den die relative Positionierung bewirkt, werden fünf Textabsätze definiert. Allen wird eine Hintergrundfarbe zugewiesen, damit ihre Elementbox sichtbar ist. Der mittlere Absatz wird mit `position:relative relativ` positioniert. Wo genau er positioniert werden soll, wird im Beispiel durch die Eigenschaften `left` (links) und `top` (oben) festgelegt.

Die Angaben zu `left` und `top` bewirken in Verbindung mit der relativen Positionierung, dass die

Elementbox im Beispiel 100 Pixel weiter rechts und 8 Pixel weiter unten beginnt als sie es normalerweise tun würde.

Im Ergebnis fällt auf, dass der Browser die Verschiebung des Elements

nach unten bzw. rechts wie definiert ausführt. Die Elementbreite wird jedoch nicht angepasst. Das p-Element nimmt weiterhin die gesamte verfügbare Breite ein, die es im normalen Textfluss einnehmen würde. Das bedeutet im Klartext, dass der Anwender möglicherweise quer scrollen muss, um alle Inhalte des Elements zu sehen. Wenn Ihnen dieses Verhalten unerwünscht erscheint, dann könnten Sie es im Beispiel durch folgende Erweiterung korrigieren:

```
<p style="background-color:aqua; position:relative; left:100px; top:8px; margin-right:100px">ABSATZ</p>
```

Info

Durch `margin-right:100px` wird hier ein Gegengewicht zu `left:100px` geschaffen. Die Elementbreite verkürzt sich rechts um so viele Pixel, wie es links durch die relative Positionierung verschoben wird. Seine rechte Elementgrenze ist nun wieder bündig zu denen der übrigen Absätze.

Um die absolute Positionierung besser zu verstehen, greifen wir auf die Hintergrundgestaltung zurück. Dort wurde bereits darauf hingewiesen, dass sich die durch die Hintergrundgrafik entstandene linke Leiste hervorragend für eine Navigationsleiste eignet. Wir werden nun einige sichtbare Boxen in dieser Leiste positionieren. Auf die Navigations-Links verzichten wir an dieser Stelle noch und notieren an ihrer Stelle nur die Linkbeschriftungen.

```
<body style="background-image:url(wallpaper.gif); background-repeat:repeat-y;">
<div style="position:absolute; left:20px; top:20px; width:230px">
<div style="position:absolute; left:0px; top:0px; width:100%; height:20px; padding:2px; background-color:olive; color:white; border:white solid 1px; font-size:16px; font-weight:bold; text-align:center">HOME</div>
<div style="position:absolute; left:0px; top:30px; width:100%; height:20px; padding:2px; background-color:olive; color:white; border:white solid 1px; font-size:16px; font-weight:bold; text-align:center">Produkte</div>
<div style="position:absolute; left:0px; top:60px; width:100%; height:20px; padding:2px; background-color:olive; color:white; border:white solid 1px; font-size:16px; font-weight:bold; text-align:center">Feedback</div>
</div>
</body>
```

Info

Innerhalb des `body`-Bereichs werden insgesamt vier `div`-Bereiche definiert. Einer davon enthält die anderen drei als Elementinhalt. Alle Bereiche werden mit `position:absolute absolut` positioniert. Dabei kommt die Regel für absolutes Positionieren zum Tragen: Der Bezug für den äußeren `div`-Bereich ist das Dokument bzw. das Browserfenster. Die Angaben `left:20px` und `top:20px` bedeuten bei diesem Bereich also den Offset gegenüber der linken oberen Ecke des Anzeigefensters. Bei den drei inneren Bereichen ist es anders. Da der äußere Bereich selbst positioniert ist, stellt er für die inneren Bereiche den Bezugspunkt dar. Die Angaben `left:0px` und `top:0px` beim ersten inneren Bereich, der Box mit dem Textinhalt „HOME“, bewirken also, dass die linke obere Ecke dieses Elements beim `Offset 20,20` beginnt. Denn das ist die linke obere Ecke des positionierten Elternelements, und der `Offset 0,0` muss relativ zu diesem Offset betrachtet werden.

Die drei inneren Bereiche unterscheiden sich in ihren CSS-Definitionen nur durch den Wert bei `top` (jeweils 30 Pixel mehr pro Bereich). Dadurch werden die Bereiche sauber untereinander positioniert.

> Der äußere div-Bereich gibt übrigens auch die Breite vor.

Dort wird mit `width:230px` eine Breite festgelegt, die optimal auf das Hintergrundbild passt, das für die schattierte Fläche am linken Fensterrand verantwortlich ist. Die inneren Bereiche erhalten nur die Angabe `width:100%`, um diesen Bereich komplett auszufüllen. Man könnte meinen, dass diese Angabe überflüssig sei, weil Block-Elemente ja normalerweise die gesamte zur Verfügung stehende Breite einnehmen. Das gilt jedoch nur, wenn sich die Elemente im normalen Textfluss befinden. Positionierte Elemente haben in dieser Hinsicht keinen Bezugspunkt und werden deshalb nur so breit dargestellt, wie es ihr Inhalt erfordert – es sei denn, man definiert explizit eine Breite.

Noch einen Schritt weiter als die absolute Positionierung geht die fixe Positionierung. Dieses Feature ist so mächtig, dass es locker 90% aller Frameset-basierten Seiten im Web überflüssig macht. Denn meistens werden Frames nur genutzt, um so genannte **non scrolling regions** zu realisieren, z.B. für Firmenlogos oder Werbebanner, die beim Scrollen von längeren Inhalten im Browserfenster stehen bleiben. Leider wird die fixe Positionierung vom MS Internet Explorer 6.0 noch nicht unterstützt, weswegen die Akzeptanz bisher gering ist. Es existieren zwar Workarounds, um auch den Microsoft-Browser dazu zu bewegen, Bereiche fix zu positionieren, doch ist dies mit einiger Trickserei verbunden, was keine wirklich befriedigende Lösung ist.

Das folgende Beispiel definiert im oberen Anzeigefensterbereich eine waagerechte `non scrolling region` mit Logo:

```
<body style="background-color:rgb(251,240,204); margin:0">
<div style="position:fixed; top:0px; left:0px; width:100%; height:66px; text-
align:right; border-bottom:rgb(244,42,42) solid 2px; padding:0px; background-
color:rgb(251,240,204); z-index:2">
```

```
<img src="logo.jpg" style="width:315px; height:66px; border:none" alt="Logo">
</div>
<div style="position:absolute; top:85px; margin-left:10%; margin-right:10%; z-
index:1">
<h1 style="font-size:48px; font-weight:normal; color:rgb(128,0,0); font-
family:'Brush Script MT',cursive">Herbstkollektion 2005</h1>
</div>
</body>
```

Info

Der Quelltext dieses Beispiels führt bei längeren Dokumentinhalten dazu, dass Browser, welche die Eigenschaft/Wert-Kombination `position:fixed` korrekt interpretieren, den Bereich mit Logo und roter Linie nicht mitscrollen. Der nach oben wegscrollende Text verschwindet unterhalb des Kopfbereichs. Die Abbildung im Beispiel zeigt, dass der Mozilla-Browser, der die `fixed`-Eigenschaft interpretiert, zwar den vertikalen Scrollbalken über die gesamte Fensterhöhe anzeigt. Dennoch bleibt beim Scrollen der Logobereich fest eingeblendet. Der MS Internet Explorer zeigt den Inhalt genauso an, scrollt jedoch den Kopfbereich mit.

Die Angaben `z-index:2` und `z-index:1` sorgen übrigens dafür, dass der untere Bereich beim Scrollen unter dem oberen verschwindet, weil er die niedrigere `z-index`-Nummer hat. Wir werden später noch genauer auf die `z-index`-Eigenschaft eingehen.

Umflusskontrolle

Ein optisch ansprechendes Gestaltungsmerkmal, bekannt aus Büchern und Zeitschriften, sind Grafiken, um die der Text herumfließt, oder auch in den Text eingeschobene und von ihm umflossene Überschriften, Infokästen oder Merksprüche. CSS stellt Mittel bereit, um solche Effekte auch auf Webseiten zu zaubern.

Beginnen wir mit einer Überschrift, die in den Text eingeschoben wird:

Der zugehörige Quelltext lautet:

```
<h1 style="width:150px; float:left; padding-top:0; margin-top:0; padding-
bottom:0; margin-bottom:10px">Blindtext</h1>
<p>Lorem ipsum ...</p>
```

Info

Der Effekt kommt durch die Kombination der CSS-Eigenschaften `float` und `width` zustande. Durch `width` wird die Anzeigebreite des `h1`-Elements begrenzt. Mit `float:left` wird angewiesen, dass nachfolgende Elemente im Textfluss um das so definierte Element herumfließen. Die Angabe `float:right` wäre ebenfalls zulässig. In diesem Fall würde die Überschrift rechts außen ausgerichtet und der nachfolgende Text würde links darum fließen. Die übrigen CSS-Definitionen im Beispiel zu Abstand und Innenabstand sorgen für die Feinjustierung des Überschriftentextes.

Eigentlich ist in CSS 2.1 eine andere Angabe vorgesehen, um obigen Effekt zu erzielen, nämlich `display:run-in`. Doch leider interpretieren die Browser diese Wertzuweisung noch nicht, weshalb noch davon abzuraten ist.

Wichtig in Zusammenhang mit der `float`-Eigenschaft ist auch die Eigenschaft `clear`. Ein weiteres Beispiel soll dies demonstrieren:

```
<body style="background-color:rgb(251,240,204)">
<h1 style="font-size:48px; font-weight:normal; color:rgb(128,0,0); font-
family:'Brush Script MT',cursive">
<img src="logo.jpg" style="width:66px; height:315px; border:none; float:left;
margin-right:20px" alt="Logo">Herbst
<br>
Kollektion
<br>
2005</h1>
<p style="clear:left">In diesem Herbst dominieren frohe und kräftige Farben.</p>
</body>
```

Info

In diesem Beispiel wird eine Grafik (`img`-Element) vom nachfolgend notierten Text umflossen. Im `<img>`-Tag der Grafik ist zu diesem Zweck die Eigenschaft `float:left` notiert. Da die Grafik eine gewisse Höhe hat, würde auch noch der im Anschluss an die `h1`-Überschrift notierte Textabsatz rechts um die Grafik fließen. Im einleitenden `<p>`-Tag des Absatzes sorgt jedoch `clear:left` dafür, dass die Wirkung von `float:left` aufgehoben wird. Für den Textabsatz gilt damit wieder der normale Textfluss, was bedeutet, dass er unterhalb der Grafik platziert wird.

Durch `clear:left` heben Sie eine `float:left`-Definition auf, durch `clear:right` eine `float:right`-Definition. Falls sowohl `float:left` als auch `float:right` „aktiv“ sind, können Sie durch `clear:both` beide Umflussanweisungen aufheben.

Eine weitere wichtige CSS-Eigenschaft, die unter anderem zur Textflusskontrolle eingesetzt werden kann, ist `display`. Mithilfe dieser Eigenschaft können Sie einem Element explizit zuweisen, ob es sich als Block- oder als Inline-Element verhalten soll. Damit ist es möglich, die Block- und Inline-Defaults von HTML außer

Kraft zu setzen.

Dem sichtbaren Text dieser Abbildung liegt folgender Quelltext zu Grunde:

```
<div>
<h1 style="display:inline; margin-right:10px; vertical-align:text-
bottom">Blindtext:</h1>
Lorem ipsum ...</div>
```

Info

Das gesamte Konstrukt ist also in einen `div`-Bereich eingebettet, damit es sich in einem Block-Element befindet, was von striktem HTML gefordert wird. Aus der `h1`-Überschrift wird durch die Definition `display:inline` ein Inline-Element. Der nachfolgende Text beginnt einfach in der gleichen Zeile direkt hinter der rechten Elementgrenze der Überschrift. Damit der nachfolgende Text nicht unmittelbar an der Überschrift klebt, erhält diese im Beispiel mit `margin-right:10px` etwas Abstand rechts.

Bemerkenswert ist im Beispiel auch die Angabe `vertical-align:text-bottom`. Mit der `vertical-align`-Eigenschaft können Sie nebeneinander stehende und gegeneinander ausgerichtete Inhalte unterschiedlicher Höhe vertikal ausrichten. Würde diese Angabe im Beispiel fehlen, so würde die erste Zeile des nachfolgenden Textes zur zweiten Zeile mehr Zeilenabstand haben als die nachfolgenden Textzeilen untereinander, weil die Schriftgröße der `h1`-Überschrift den Zeilenabstand beeinflusst. Auf die Eigenschaft `vertical-align` werden wir in Zusammenhang mit der CSS-Formatierung von Tabelleninhalten noch genauer eingehen.

Arbeiten mit Layern, Sichtbarkeit und Transparenz

Dank absoluter Positionierung in Verbindung mit Angaben zu Breite und Höhe ist es möglich, Elementboxen frei anzuordnen. Selbstverständlich ist es auch möglich, zwei Elemente an der gleichen Stelle anzuordnen. Untersuchen wir, was dabei passiert:

```
<div style="position:absolute; top:20px; left:60px; width:400px; height:100px;
border:solid 2px blue">
<h1>Erster Bereich</h1>
</div>
<div style="position:absolute; top:20px; left:60px; width:400px; height:100px;
border:solid 2px red">
<h1>Zweiter Bereich</h1>
</div>
```

Info

Es werden zwei `div`-Bereiche definiert. Beide erhalten identische Angaben zur absoluten Position. Sie unterscheiden sich lediglich in der Rahmenfarbe (der erste Bereich hat einen blauen Rahmen und der zweite einen roten) sowie im Elementinhalt (beide enthalten eine `h1`-Überschrift, jedoch mit unterschiedlichem Text).

Die Anzeige des Beispiels im Browser offenbart Folgendes:

Der zweite Bereich liegt über dem ersten, was dadurch sichtbar wird, dass die Rahmenfarbe Rot ist. Denn dies ist die Rahmenfarbe des zweiten Bereichs. Der Inhalt ist dagegen transparent, was daran erkennbar ist, dass beide Überschriftentexte sichtbar sind. Es sind also folgende Regeln erkennbar:

- Bei positionierten Elementboxen, die sich überlappen, ist die natürliche Schichtreihenfolge durch die Reihenfolge der Notation im Dokument vorgegeben. Ein hinter einem anderen Element notiertes Element wird über diesem angeordnet.
- Der Inhalt von Elementboxen ist transparent.

Mithilfe geeigneter CSS-Definitionen lässt sich dieses Default-Verhalten ändern. Die Schichtreihenfolge von Elementen können Sie durch die Eigenschaft `z-index` beeinflussen. Dazu bekommen alle Elemente, die sich überlappen, eine `z-index`-Eigenschaft zugewiesen. Als Wert wird jeweils eine frei wählbare Zahl angegeben. Das Element mit der niedrigsten Zahl wird dann in der untersten Schicht angeordnet, also von allen übrigen Elementen überlappt, und das Element mit dem höchsten `z-index`-Wert überlappt alle anderen Elemente. In unserem Beispiel können wir das einfach überprüfen, indem wir dem zweiten `div`-Bereich einen höheren `z-index`-Wert zuordnen als dem ersten:

```
<div style="position:absolute; top:20px; left:60px; width:400px; height:100px;
border:solid 2px blue; z-index:2">
<h1>Erster Bereich</h1>
</div>
<div style="position:absolute; top:20px; left:60px; width:400px; height:100px;
border:solid 2px red; z-index:1">
<h1>Zweiter Bereich</h1>
</div>
```

Info

Der sichtbare Layer ist nun das `div`-Element mit dem blauen Rahmen, obwohl dieses als Erstes notiert wurde. Grund ist, dass es mit `z-index:2` einen höheren Schichtwert erhalten hat als der zweite Bereich,

dem `z-index:1` zugewiesen wurde. Die Default-Transparenz von Elementen, erkennbar an den sichtbaren Überschrifttexten beider Bereiche, lässt sich am einfachsten durch eine definierte Hintergrundfarbe ausschalten:

```
<div style="position:absolute; top:20px; left:60px; width:400px; height:100px;
border:solid 2px blue; background-color:white">
<h1>Erster Bereich</h1>
</div>
<div style="position:absolute; top:20px; left:60px; width:400px; height:100px;
border:solid 2px red; background-color:white">
<h1>Zweiter Bereich</h1>
</div>
```

Info

Beide Bereiche haben nun mit `background-color:white` eine definierte Hintergrundfarbe erhalten. Die `z-index`-Eigenschaft wurde entfernt.

Der zweite Bereich überlappt den ersten aufgrund der natürlichen Schichtreihenfolge. Da die Elementinhalte der Bereiche nun nicht mehr transparent sind, überdeckt der zweite Bereich den ersten vollständig.

Wer das Arbeiten mit Layern aus Grafikprogrammen kennt, weiß, dass besonders edle Effekte durch eine skalierbare Transparenz sich überdeckender Inhalte entstehen. CSS bietet ein entsprechendes Feature Sprachstandard 3.0 an. In einigen Browsern ist es bereits implementiert:

```
<div style="position:absolute; top:20px; left:60px; width:400px; height:100px;
border:solid 2px red; background-color:white; opacity:0.5">
<h1>Zweiter Bereich</h1>
</div>
```

Info

Das Beispiel zeigt den Quelltext des zweiten unserer beiden identisch positionierten Bereiche. Beide Bereiche sind aufgrund der definierten weißen Hintergrundfarbe nicht transparent. Durch die CSS-3.0-Eigenschaft `opacity` lässt sich jedoch eine Halbtransparenz definieren. Wie stark unterhalb liegende Inhalte durchscheinen, lässt sich durch die Wertzuweisung skalieren. Empfohlen wird hierbei, mit Werten von 0.0 (`voll transparent`) bis 1.0 (`voll opak`, also deckend) zu arbeiten.
Das Ergebnis der Zuweisung von `opacity:0.5` ist, dass der zweite Bereich den ersten nun zu 50%

durchscheinen lässt. Auch die Rahmenfarben Rot und Blau mischen sich zu Lila.

Der MS Internet Explorer 6.0 interpretiert die `opacity`-Eigenschaft noch nicht, doch es gibt einen Workaround: die Microsoft-proprietäre CSS-Erweiterung der grafischen Filter. Auf unser Beispiel angewendet, könnten wir die CSS-Definition für den Internet Explorer folgendermaßen anpassen:

```
<div style="position:absolute; top:20px; left:60px; width:400px; height:100px;
border:solid 2px red; background-color:white; opacity:0.5;
filter:alpha(opacity=50)">
<h1>Zweiter Bereich</h1>
</div>
```

Info

Die CSS-Angabe `filter:alpha(opacity=50)` entspricht `opacity:0.5` und wird aber im Gegensatz zu Letzterer vom Internet Explorer interpretiert. Das Notieren beider Angaben bewirkt also, dass Browser, welche bereits mit `opacity` zurechtkommen, diese Eigenschaft interpretieren, während der Internet Explorer die filter-Eigenschaft interpretiert.

Das Arbeiten mit skalierbarer Teiltransparenz führt besonders bei Schrift auf halb durchscheinenden Grafiken zu edlen Effekten, wie sie beispielsweise aus zahlreichen Computerspielen mit transparenten Menüs bekannt sind. Behalten Sie jedoch bei solchen Effekten stets die Lesbarkeit des Textes im Auge. Wenn die Lesbarkeit dem Design geopfert wird, kann das Design noch so schick aussehen, es ist dennoch schlecht.

Jedes **HTML-Tag** ist von einer ganzen Welt aus CSS-Eigenschaften umgeben, die das Aussehen des Tags im Webbrowser bestimmen. Einige Eigenschaften, wie Rahmen oder Hintergrundfarben, sind sofort mit bloßem Auge erkennbar. Andere scheinen dagegen unsichtbar, z.B. Innen- und Außenabstände. Sie bestimmen, wie viel Leerraum um die einzelnen **HTML-Elemente** dargestellt wird. Wenn Sie die Funktionsweise dieser Eigenschaften begriffen haben. können Sie attraktive, spaltenbasierte Layouts und dekorative Seitenleisten erstellen und selbst bestimmen, wie groß der Abstand (bzw. Leerraum oder `white space`) zu den anderen HTML-Elementen sein soll. Mit wenigen **CSS-Regeln** erscheint Ihre Seite nicht mehr so überladen, sondern leichter und professioneller.

Gemeinsam bilden die in diesem Kapitel besprochenen CSS-Eigenschaften eines der wichtigsten Konzepte von CSS: das Boxmodell.

Das Boxmodell verstehen

Wenn Sie an Absätze oder überschriften denken, stellen Sie sich dabei vermutlich Buchstaben, Wörter und Sätze vor. Beim `<img>`-Tag haben Sie Fotos, Logos und Bilder vor Augen. Ein Webbrowser behandelt diese (und auch alle anderen) Tags als kleine Kästen (eng!. boxes), die die Sätze, Bilder, überschriften usw. enthalten, wie in Abbildung 1 gezeigt.

Die den Inhalt umgebende Box besteht aus verschiedenen Teilen:

- Der Innenabstand (`padding`) ist der Leerraum zwischen dem Inhalt und dem Rahmen. Der Innenabstand sorgt beispielsweise fur einen gewissen Abstand zwischen einem Foto und seinem Rahmen.
- Der Rahmen (`border`) steht in CSS für die visuelle Begrenzung (meistens eine durchgehende Linie. Details finden Sie im weiteren Verlauf dieses Kapitels), um das HTML-Element herum. Ein Rahmen muss so definiert werden, dass er das gesamte Element umgibt oder nur an bestimmten Seiten dargestellt wird (oder auch gar nicht).
- Die Hintergrundfarbe (`background-color`) füllt den vom Rahmen begrenzten Raum innerhalb der Box (auch wenn der Rahmen nicht angezeigt wird).
- Der Außenabstand (`margin`) bezeichnet den Leerraum, der einzelne Tags voneinander trennt. Ein Beispiel für Außenabstände ist der Leerraum, den ein Browser standardmäßig zwischen dem Rand des Browserfensters und dem eigentlichen Inhalt Ihrer Seite darstellt.

Benutzen Sie diese Eigenschaften für beliebige Tags einzeln oder miteinander kombiniert: Sie können für ein Tag nur einen Außenabstand deftnieren oder auch einen Rahmen, Innen- und Außenabstände. Oder Sie verwenden einen Rahmen und Außenabstände, aber keinen Innenabstand, und so weiter. Wenn Sie diese Einstellungen nicht anpassen, werden die Standardwerte des jeweiligen Browsers verwendet, was in Ordnung sein kann – oder auch nicht. So benutzen die Browser, von ein paar Ausnahmen abgesehen, standardmäßig weder Rahmen noch Innenabstände. Einige Tags, wie Absätze oder überschriften, besitzen allerdings obere und untere Außenabstände. (In diesem Kapitel werden Sie diese Standardeinstellungen und die Methoden, sie durch eigene Werte zu ersetzen, genauer kennenlernen.)

Den Leerraum mit Innen- und Außenabständen kontrollieren

Innen- und Außenabstände legen fest, wie viel Leerraum um ein **HTML-Element** herum dargestellt wird. Mit diesen Eigenschaften können Sie die einzelnen Tags visuell voneinander trennen, um beispielsweise ein Navigationsmenü auf der linken Seite vom Hauptinhalt einer Seite zu trennen oder um zwischen dem eigentlichen Inhalt und dessen Rahmen etwas Platz zu schaffen. Mit Innenabständen können Sie die Entfernung zwischen dem Bildrand eines Fotos und dessen Rahmen einstellen, wie in Abbildung 2 gezeigt.

Innen- und Außenabstände funktionieren auf ähnliche Weise. Sofern Sie keinen Rahmen oder eine eigene Hintergrundfarbe definieren, ist es schwer zu sagen, ob der Leerraum zwischen zwei Tags durch Innen- oder Außenabstände erzeugt wird. Wird das Element dagegen von einem Rahmen umgeben oder besitzt es eine eigene Hintergrundfarbe, ist der Unterschied zwischen beiden Eigenschaften deutlich zu sehen. Innenabstände schaffen Platz zwischen Inhalt und Rahmen und sorgen so dafür, dass der Inhalt nicht in

seine Box gequetscht erscheint. Außenabstände sorgen dagegen für den Abstand zwischen einzelnen Elementen, wodurch die Seite insgesamt „luftiger“ erscheint.

Sie können die Breite der Innen- und Außenabstände für jede Seite eines Elements einzeln steuern. Die vier Eigenschaften für Außenabstände heißen `margin-top`, `margin-right`, `margin-bottom` und `margin-left`. Entsprechend sind auch die Eigenschaften für Innenabstände benannt: `padding-top`, `padding-right`, `padding-bottom` und `padding-left`. Für Innen- und Außenabstände können Sie beliebige in CSS gültige Maßeinheiten verwenden. z.B.:

```
margin-right: 20px;
padding-top: 3em;
margin-1eft: 10%;
```

Wie für textbezogene Eigenschaften. werden auch hier häufig Pixel und em-Einheiten verwendet. Ein Außenabstand von 20 Pixeln erzeugt einen 20 Pixel großen Leerraum. während ein Innenabstand von 3em innerhalb des Elements einen Leerraum erzeugt, der dreimal größer ist als die Standardschriftgröße für das betreffende Element. Zwar sind auch Prozentwerte möglich, sie sind allerdings komplizierter in der Benutzung.

Info

Soll ein Element überhaupt keine Innen- oder Außenabstände haben, verwenden Sie den Wert 0 (z.B. `margin-top: 0;` oder `padding-bottom: 0;`). Wenn der standardmäßige leerraum am Rand des Ansichtsbereichs entfernt werden soll, etwa damit ein Kopfteil, logo oder ein anderes Element bündig mit dem Fensterrand dargestellt wird, ist es gängige Praxis, Innen- und Außenabstand für das `<body>`-Tag auf null zu setzen: `margin: 0; padding 0;`. Normalerweise ist die Eigenschaft `margin` für den leerraum am Seitenrand zuständig. Nur Opera verwendet hierfür `padding`, deshalb müssen beide Eigenschaften angegeben werden.

Kurzschrift-Eigenschaften für Innen- und Außenabstände

Häufig sollen für alle vier Seiten eines Elements Innen- und Außenabstände definiert werden. Allerdings kann es mühsam sein, jedes Mal die Einzel-Eigenschaften (`margin-right`, `margin-left` usw.) anzugeben. Glücklicherweise haben die Entwickler von CSS mitgedacht und die Kurzschrift-Eigenschaften `margin` und `padding` erfunden, mit denen Sie alle vier Werte in einer gemeinsamen Deklaration definieren können. wie hier:

```
margin: 0 10px 10px 20px;
padding: 10px spx spx 10px;
```

Info

Ist der Wert einer CSS-Eigenschaft 0, brauchen Sie keine Maßeinheit anzugeben. Es reicht also, `margin:`

`0;` zu schreiben, wenn Sie `margin: 0px;` meinen.

Bei der Angabe der vier Werte muss eine bestimmte Reihenfolge eingehalten werden, und zwar: oben, rechts, unten, links. Im Englischen gibt es eine nette Eselsbrücke: Halten Sie sich an die richtige Reihenfolge, wenn Sie keinen TRouBLe bekommen wollen. Die vier Großbuchstaben stehen für die Anfangsbuchstaben der englischen Wörter `top` (oben), `right` (rechts), `bottom` (unten) und `left` (links).

Soll für alle vier Seiten der gleiche Wert verwendet werden, ist die Sache sogar noch einfacher: Geben Sie diesen Wert einfach nur einmal an. Wollen Sie beispielsweise alle Außenabstände vom <h1>-Tag entfernen, können Sie schreiben:

```
h1 {
margin: 0;
}
```

Wenn ein Element einen einheitlichen Innenabstand von 10 Pixeln bekommen soll, können Sie entsprechend dieses angeben:

```
padding: 10px;
```

Info

Soll für die Bereiche ober- und unterhalb eines Element der gleiche Wert und für den Platz links und rechts davon ein anderer Wert verwendet werden, reicht es, diese Werte jeweils einmal anzugeben. Die Deklaration margin: 0 2em; setzt den oberen und unteren Außenabstand auf null, während der linke und der rechte Außenabstand eine Breite von jeweils 2 em-Einheiten erhalten.

Wenn Außenabstände kollabieren

Wenn es um **CSS** geht, ist 2 mal 2 nicht unbedingt 4. Treffen beispielsweise der untere Außenabstand eines Elements und der obere Außenabstand eines anderen zusammen, kann es zu interessanten mathematischen Phänomenen kommen. Anstatt die beiden Außenabstände einfach zu addieren, verwendet der Webbrowser nur den größeren der beiden (Abbildung 3 oben). Folgende Situation: Unterhalb einer Liste befindet sich ein Absatz. Für die Liste wurde ein unterer Außenabstand von 30 Pixeln definiert (`margin-bottom: 30px;`). für den Absatz ein oberer Außenabstand von 20 Pixeln (`margin-top: 20px;`). Anstatt jetzt beide Werte zu einem gemeinsamen Leerraum von 50 Pixeln zu addieren, verwendet der Webbrowser nur den größeren der beiden Werte, in diesem Fall also 30 Pixel. Dieses Verhalten können Sie verhindern, indem Sie stattdessen Innenabstände (`padding`) verwenden (Abbildung 3 unten).

Noch verwirrender werden die Dinge. wenn es um verschachtelte Elemente geht. Diese Situation kann leicht zu Kopfschmerzen führen. Angenommen, Sie bauen eine Warnmeldung in Ihre Seite ein (innerhalb eines `<div>...</div>`-Tag-Paars. das die Nachricht enthält). Um die Meldung vom übrigen Inhalt (einer überschrift darüber und einem Absatz mit Text darunter) optisch zu trennen. erhält die Nachricht (das `<div>`-Element) einen oberen und unteren Außenabstand von je 20 Pixeln. So weit, so gut.

Die Warnmeldung erhält nun eine eigene überschrift. die ihrerseits darüber und darunter etwas Leerraum erhalten soll. Also setzen Sie den oberen und unteren Außenabstand der überschrift auf 10 Pixel. Wenn Sie aber jetzt glauben. dass der Leerraum innerhalb des `<div>`-Tags erscheint, liegen Sie leider falsch (s. Abbildung 4). Stattdessen erscheint der Außenabstand außerhalb des umgebenden `<div>`-Elements. Dies Verhalten ist unabhängig davon. wie hoch der definierte Außenabstand für die überschrift ist.

Info

Werden zwei Außenabstände, die sich berühren, zusammengefasst. spricht man in der **CSS-Fachsprache** von „kollabierenden Außenabständen“.

Sie können das Problem auf zwei Arten umgehen: Entweder Sie umgeben das `<div>`-Tag mit einem kleinen Innenabstand. oder Sie versehen es mit einem Rahmen. Da sich Innenabstand und Rahmen per definitionem zwischen den beiden Außenabständen befmden, berühren sich die beiden Außenabstände nicht mehr und werden daher auch nicht zusammengefasst – und die überschrift erhält etwas mehr Platz zum Atmen (Abbildung 4 rechts).

Info

Horizontale Außenabstände (links und rechts) für Elemente, die als „Float“ definiert wurden, werden nicht auf diese Weise zusammengefasst. ähnliches gilt für absolut und relativ positionierte Elemente, die ebenfalls nicht zusammengefasst werden.

Negative Außenabstände zum Entfernen von Leerraum

Die meisten Längenangaben müssen einen positiven Wert haben. Ein minus 20 Pixel großer (oder kleiner) Text soll zwar bereits mit dem Large Hadron Collider erzeugt worden sein, für Webseiten ist er aber nicht sinnvoll. Auch Innenabstände können keine negativen Werte annehmen. Aber CSS ermöglicht eine Reihe kreativer Techniken, die mit negativen Werten für Außenabstände arbeiten. Anstatt einem Element Leerraum hinzuzufügen, entfernt ein negativer Wert für Außenabständen den Leerraum. Mit dieser Methode ist es möglich, überschneidungen von Absätzen und überschriften in der Weise zu erzeugen, dass Inhalt aus seinem umgebenden Element herausragt (z.B. eine Seitenleiste oder ein anderes Layout-`<div>`) oder sogar ganz oder teilweise aus dem sichtbaren Bereich des Browserfensters verschwindet. Daneben ist es sogar möglich, sinnvolle Dinge mit negativen Außenabständen anzufangen.

Selbst wenn Sie die Außenabstände zwischen zwei überschriften auf null setzen gibt es (dank der

Eigenschaft `line-height`), einen gewissen Abstand zwischen dem Text der überschriften. Das ist normalerweise auch gut so – schließlich sind Sätze, die direkt aneinandergrenzen oder sich sogar überlappen, nicht besonders gut zu lesen. Vorsichtig eingesetzt, kann ein geringer Abstand zwischen zwei überschriften aber zu sehr interessanten Ergebnissen führen. Der zweiten überschrift in Abbildung 5 („Raise Tuna“) wurde beispielsweise ein negativer Außenabstand von -10 Pixeln zugewiesen. Hierdurch wird diese überschrift effektiv um 10 Pixel nach oben verschieben.

Info

Anstelle eines negativen Außenabstands oberhalb vom Absatz können Sie natürlich auch einen negativen Außenabstand an der Unterseite der überschrift verwenden. Beides hat den gleichen Effekt, indem der Abstand zwischen Absatz und überschrift verkleinert wird.

Inline- und Block-Elemente darstellen

CSS unterscheidet zwei grundsätzliche Arten von Elementen, **Block-Elemente** und **Inline-Elemente**, die jeweils ihre eigene Art von Box erzeugen, Bei einem Block-Element wird vor und nach dem Element automatisch ein Zeilenumbruch erzeugt. Das beste Beispiel ist das <p>-Tag. Es erzeugt eine Box, die von den Tags darüber und darunter getrennt ist. Andere Beispiele für Block-Elemente sind überschriften, `<div>`-Tags, Tabellen und Listen (`<ul>`, `<ol>`, `<dl>`).

Inline-Elemente erzeugen dagegen keine Umbrüche. Sie stehen aufderselben Zeile wie der Inhalt davor oder danach. Ein Beispiel für ein Inline-Element ist das `<strong>`-Tag. Mit diesem Tag formatierte Wörter stehen „gleichberechtigt“ neben den übrigen Wörtern – selbst neben Text, der mit anderen Inline-Elementen wie dem -Tag formatiert wurde. Es würde auch ziemlich seltsam aussehen, wenn ein einzelnes Wort in einem Absatz plötzlich auf einer eigenen Zeile stehen würde, bloß weil es mit `<strong>`-Tags umgeben ist. Weitere Beispiele für Inline-Elemente sind `<img>` für das Einbinden von Bildern. <a> (für Links) und die verschiedenen Tags zum Erzeugen von Formularfeldern.

Dabei haben Inline- und Block-Elemente mehr Gemeinsamkeiten als Unterschiede. So können Sie Schriftarten, -farben und Hintergründe definieren und für beiden Arten Rahmen defmieren. Wenn es um Innen- und Außenabstände geht, gibt es allerdings gewisse Unterschiede. So können Sie zwar den Leerraum links und rechts von einem Inline-Element mit Innen- und Außenabständen verändern, die Höhe eines Inline-Elements lässt sich dagegen mit oberen und unteren Innen- und Außenabständen nicht verändern. Für den oberen Absatz in Abbildung 6 wurde das Inline-Element mit Rahmen (`border`), einer Hintergrundfarbe (`background-color`) und einem Außenabstand von 20 Pixeln an allen Seiten versehen. Der Browser fügt aber nur links und rechts vom Inline-Element den gewünschten Leerraum ein.

Info

Für die Regel, dass Inline-Elemente beim Hinzufügen von vertikalen Innen- oder Außenabständen nicht höher werden, gibt es allerdings eine Ausnahme: das `<img>`-Tag. Für dieses Tag wird, obwohl es ein Inline-Element ist, die Höhe der Box für das Bild entsprechend den Werten für `podding` und `margin` angepasst.

Gelegentlich ist es wünschenswert, dass sich Inline-Elemente mehr wie Block-Elemente verhalten und umgekehrt. Die Elemente einer Liste sind hier ein gutes Beispiel. Normalerweise steht jeder Eintrag brav auf seiner eigenen Zeile. Manchmal sollen die Einträge aber gemeinsam auf einer Zeile dargestellt werden (z.B. in einem Navigationsmenü). Oder es soll ein einzelnes Inline-Element wie ein Block-Element funktionieren. Vielleicht soll ein bestimmtes in einen Absatz eingebettetes Bild nicht vom Text umflossen werden, sondern mit dem entsprechenden Platz darüber und darunter für sich stehen.

Genau für diese Fälle gibt es die CSS-Eigenschaft `display`. Mit ihr können Sie die sogenannte Darstellungsrolle eines Elements verändern, sodass ein Block-Element (z.B. ein Listeneintrag) wie ein Inline-Element dargestellt wird:

```
display: inline;
```

Oder Sie können Inline-Elemente (Bilder. Links usw.) wie Block-Elemente darstellen lassen:

```
display: block;
```

Info

Die Eigenschaft `display` besitzt eine Vielzahl von möglichen Optionen, die aber oft nicht vollständig von den aktuellen Browser unterstützt werden. Der Wert `none` zählt jedoch nicht dazu. Auch wenn das nicht ganz offensichtlich ist hat er viele Anwendungsmöglichkeiten. Er sorgt dafür, dass ein so markiertes Element nicht angezeigt wird. Es erscheint einfach nicht im Webbrowser. Mit ein wenig JavaScript können Sie ein so verstecktes Element auf der Stelle wieder sichtbar machen, indem Sie einfach seine Darstellungsrolle auf `inline` oder `block` setzen. Allerdings gibt es auch Methoden, per `display: none;` versteckte Elemente nur mit CSS wieder zum Vorschein zu bringen.

Rahmen

Ein **Rahmen** ist einfach eine Linie. die um ein Element herum dargestellt wird. Wie in Abbildung 1 zu sehen ist, befindet sich der Rahmen zwischen Innen- und Außenabstand. Ein Rahmen, der das gesamte Element umgibt, kann als „Bilderrahmen“ dienen oder die Begrenzungen eines Kopfteils oder eines anderen Seitenelements markieren. Hierbei können die Seiten des Rahmens einzeln definiert werden. Sie können also frei wählen, an welchen Seiten eines Elements die Rahmenlinien dargestellt werden sollen. Dadurch können Sie Rahmen auch als eigenständige Designelemente verwenden. Wenn Sie links von einem Element eine 1em dicke Begrenzungslinie definieren, kann diese auch als eckiges Aufzählungszeichen angesehen werden. Ein Rahmen unterhalb eines Absatzes kann die gleiche Funktion

übernehmen wie das <hr>-Element (horizontale Linie – horizontal rule), indem es eine visuelle Trennung zwischen verschiedenen Teilen einer Seite realisiert.

Für Rahmen gibt es drei Parameter, die bestimmen, ob und wie ein Rahmen dargestellt wird: `border-color` definiert die **Rahmenfarbe**, die wie die Schriftfarbe als hexadezimaler Wert, Schlüsselwort oder RGB-Wert angegeben werden kann; `border-width` bestimmt die Breite des Rahmens. Hiermit ist die Dicke der Linie gemeint, die um ein Element herum gezeichnet wird. Außer Prozentwerten können Sie hier alle gängigen CSS-Längeneinheiten verwenden sowie die Schlüsselwörter `thin`, `medium` und `thick`. Am häufigsten werden allerdings einfach Pixel verwendet, die außerdem am einfachsten zu verstehen sind. Negative Angaben sind hier nicht möglich.

Die Eigenschaft `border-style` schließlich legt fest, wie die Linie dargestellt wird. Sie können aus mehreren verschiedenen Darstellungsformen wählen, die allerdings von Browser zu Browser recht unterschiedlich dargestellt werden, wie in Abbildung 7 zu sehen ist. Der Rahmenstil wird über Schlüsselwörter wie `solid` und `dashed` angegeben. Die einzelnen Rahmenstile sind: `solid` (durchgehend), `dotted` gepunktet), `dashed` (gestrichelte Linie), `double` (doppelte Linie), `groove` (3-D-Rahmen, spitz, Variante 1), `ridge` (3-D-Rahmen, spitz, Variante 2), `inset` (3-D-Rahmen, erhaben), `outset` (3-D-Rahmen, eingelassen), `none` (kein Rahmen) und `hidden` (Rahmen, versteckt).

Info

Die Schlüsselwörter `none` und `hidden` funktionieren auf die gleiche Weise: Sie entfernen den Rahmen komplett. Der Wert `none` ist sinnvoll, um eine einzelne Rahmenseite komplett auszublenden. Außerdem ist `none` der Standardwert für `border-style`. Das heißt, Sie müssen immer einen Rahmenstil definieren, damit der Rahmen auch angezeigt wird. Dieser Fehler wird öfter gemacht als Sie vielleicht denken.

Info

Im Internet Explorer 6 für Windows gibt es zwischen einem 1 Pixel breiten gestrichelten Rahmen und einem 1 Pixel breiten gepunkteten Rahmen keinen Unterschied.

Die Kurzschrift-Eigenschaft für Rahmen

Wenn Sie jemals eine Liste der verschiedenen CSS-Eigenschaften für Rahmen gesehen haben, könnten Sie schnell den Eindruck gewinnen, dass Rahmen eine sehr komplexe Sache sind. Schließlich gibt es mehr als 20 verschiedene Eigenschaften allein für Rahmen, die Sie in den folgenden Abschnitten kennenlernen werden. Im Prinzip sind diese Eigenschaften aber lediglich Variationen über ein Thema, das verschiedene Möglichkeiten zur Kontrolle von Rahmenfarbe, -breite und -stil für die einzelnen Rahmenseiten (oben. rechts, unten und links) bietet. Die einfachste und am leichtesten zu verstehende Eigenschaft heißt einfach `border`, mit der alle vier Rahmenseite auf einmal definiert werden:

```
border: 4px solid #F00;
```

Die obige **Deklaration** erzeugt einen Rahmen, der das gesamte Element umgibt, in der Farbe Rot mit einer Breite von 4 Pixeln. Mit dieser Eigenschaft können Sie beispielsweise einen einfachen Rahmen um ein Bild, eine Navigationsleiste oder ein anderes Element erzeugen, das von anderen Teilen der Seite getrennt erscheinen soll.

Info

Die Reihenfolge der einzelnen Werte ist hier nicht so wichtig. `4px solid #F00;` funktioniert genauso gut wie `border: #F00 solid 4px;`. Damit Sie beim Erstellen Ihrer eigenen Regeln nicht durcheinanderkommen, sollten Sie sich allerdings möglichst an eine bestimmte Reihenfolge pro Stylesheet halten.

Rahmenseiten einzeln formatieren

Sie können die einzelnen Rahmenseiten auch separat mit **Stildefinitionen** versorgen. Analog zu den Eigenschaften für Innen- und Außenabstände werden dem Begriff `border` Schlüsselwörter für die jeweilige Rahmenseite angehängt. Die einzelnen Eigenschaften heißen `border-top`, `border-bottom`, `border-left` und `border-right`. Sie funktionieren genau wie `border`, kontrollieren aber jeweils nur eine Seite. Die folgende Deklaration erzeugt unter dem Element eine 2 Pixel dicke. rote, gestrichelte Linie:

```
border-bottom: 2px dashed red;
```

Sie können `border` auch mit einer seitenspezifischen Eigenschaft kombinieren, um den grundsätzlichen Rahmenstil für ein Element, z.B. einen Absatz, zu definieren, der dann für die Unterseite mit anderen Werten überschrieben wird. Entweder schreiben Sie hierfür vier Zeilen **CSS-Code**, wie hier:

```
border-top: 2px solid black;
border-left: 2px solid black;
border-right: 2px solid black;
border-bottom: 4px dashed #333;
```

Sie können den gleichen Effekt aber auch mit nur zwei Zeilen CSS-Code erreichen:

```
border: 2px solid black;
border-bottom: 4px dashed #333;
```

Die erste Zeile im zweiten Beispiel definiert das grundsätzliche Aussehen aller vier Rahmenseiten. Die zweite Zeile definiert danach die Darstellung der Rahmenunterseite neu. Das ist nicht nur einfacher, weil

Sie lediglich zwei Zeilen zu schreiben brauchen, sondern auch, weil sich eine solche Regel leichter anpassen lässt. Soll die Farbe der oberen, linken und rechten Rahmenseite verändert werden, brauchen Sie nur eine Zeile Code anzupassen und nicht drei:

```
border: 2px solid red;
```

Wenn Sie das grundsätdiche Aussehen aller vier Rahmenseiten mit der Kurzschrift-Eigenschaft `border` definieren und die Definitionen für eine Seite überschreiben (z.B. mit `border-left`, `border-top` usw.), ist es absolut wichtig, den Code in der richtigen Reihenfolge zu schreiben. Zuerst muss immer die allgemeine Deklaration stehen und erst danach die spezifische Definition:

```
border: 2px solid black;
border-bottom: 4px dashed #333;
```

Da die Eigenschaft `border-bottom` hier als Zweites definiert wird, kann sie die Einstellung von `border` überschreiben. Wäre die Reihenfolge umgekehrt, würde die `border-bottom`-Deklaration ihrerseits von der darunterstehenden `border`-Regel überschrieben, und alle vier Seite würden gleich dargestellt. Aufgrund der Kaskade, kann die zuletzt definierte Eigenschaft vorhergehende Einstellungen außer Kraft setzen.

Sie können diese Methode auch benutzen, um gezielt eine Rahmenseite mit dem Schlüsselwort `none` auszublenden. Um einen Rahmen für ein Element zu definieren, bei dem die rechte Rahmenseite fehlen soll, reichen ebenfalls zwei Zeilen CSS-Code aus:

```
border: 2px inset #FFCC33;
border-right: none;
```

Genau diese Möglichkeit, die einzelnen Rahmenseiten und ihre Darstellung sehr genau zu steuern, ist der Grund für die vielen einzelnen CSS-Eigenschaften für Rahmen. Die übrigen 15 Rahmeneigenschaften ermöglichen Ihnen die Definition bestimmter Farben, Rahmenstile und Breiten für die einzelnen Rahmenseiten. Die Deklaration `border: 2px double #FFCC33;` können Sie daher auch wie folgt formulieren:

```
border-width: 2px;
border-style: double;
border-color: #FFCC33;
```

Da hierfür drei Zeilen Code gebraucht werden, wollen Sie diese Methode vermutlich meistens vermeiden. Dennoch gibt es für jede Rahmenseite eigene CSS-Eigenschaften, mit denen Sie die einzelnen Parameter einstellen können. Für die rechte Rahmenseite gibt es die Einzel-Eigenschaften `border-right-width`,

`border-right-style` und `border-right-color`. Für die anderen Rahmenseiten (`left`, `top` und `bottom`) gibt es entsprechend benannte Einzel-Eigenschaften wie `border-left-width`, `border-left-style` und so weiter.

Im folgenden Beispiel soll für ein Element ein zwei Pixel breiter gestrichelter Rahmen erzeugt werden. Allerdings soll jede Seite eine unterschiedliche Farbe erhalten. (Vielleicht arbeiten Sie an einer Website für Kinder.) Mit **CSS** ist das ganz einfach:

```
border: 2px dashed green;
border-left-color: blue;
border-right-color: yellow;
border-bottom-color: red;
```

Mit diesen Deklarationen erzeugen Sie in der ersten Zeile den gewünschten Rahmen, der im Moment noch vollständig grün gefärbt ist. Mit den folgenden drei Zeilen wird dieser Wert für die einzelnen Rahmenseiten mit eigenen Definitionen überschrieben, wodurch die linke Seite blau wird, die rechte gelb und die Unterseite rot.

Auf gleiche Weise können Sie die Breite der Seiten einzeln einstellen, z.B. mit `border-right-width: 4px;`. Diese Methode hat noch einen angenehmen Nebeneffekt: Wenn Sie später entscheiden, doch lieber eine durchgehende statt einer gestrichelten Linie zu verwenden, brauchen Sie nur in der entsprechenden Zeile `dashed` in `solid` zu ändern.

Info

Normalerweise wird bei der Verwendung von Rahmen auch ein gewisser Innenabstand (`podding`) definiert, der den Abstand zwischen Inhalt und Rahmen vergrößert. Ohne einen gewissen Innenabstand befinden sich die Rahmenlinien meist zu nah am Inhalt.

Hintergrundfarben

Es ist ein Kinderspiel, für eine komplette Seite eine Hintergrundfarbe zu definieren – oder auch für eine überschrift oder ein anderes Seitenelernent. Verwenden Sie hierfür die Eigenschaft `background-color`, gefolgt von einem Farbwert. Um Ihre Seite mit einem schockierend grünen Hintergrund zu versehen, reicht beispielsweise folgende Regel:

```
body {
background-color: #6DDA3F;
}
```

Alternativ dazu können Sie auch einen klassenbasierten Stil definieren, z.B. `.review`, der die Eigenschaft `background-color` verwendet. Diese Klasse können Sie dann dem <body>-Tag (und anderen

Elementen) zuweisen, wie hiermit: `<body class="review">`.

Info

Sie können als Hintergrund für die HTML-Elemente (inklusive `<body>`) auch Bilder verwenden. Deren Platzierung und Wiederholung lässt sich auf viele Arten steuern, wie Sie im nächsten Kapitel sehen werden.

Mit Hintergrundfarben lässt sich eine Reihe verschiedener visueller Effekte erzielen. Sie können den Eindruck von überschriften verstärken, indem Sie für den Hintergrund eine dunkle Farbe und für die Schrift selbst eine helle Farbe definieren. Hintergrundfarben stellen außerdem eine hervorragende Möglichkeit dar, verschiedene Elemente einer Seite, wie Navigationsleiste, Logo und Seitenleiste visuell voneinander zu trennen.

Info

Bei der Verwendung von **Hintergrundfarben** und **Rahmen** sollten Sie Folgendes beachten: Bei gestrichelten und gepunkteten Rahmenstilen (5. Abbildung 7) scheint die Hintergrundfarbe zwischen den Punkten oder Strichen hindurch. Der Webbrowser stellt die Hintergrundfarbe also bis zur Außenkante des Rahmens dar.

Höhe und Breite festlegen

Zwei weitere CSS-Eigenschaften sind ebenfalls Teil des Boxmodells. Mit ihnen können Sie die Ausmaße der **HTML-Elemente** einer Seite (Tabellen, Spalten, Logo, Seitenleiste usw.) kontrollieren. Die Eigenschaften `height` und `width` definieren für den Inhaltsbereich eines Elements eine bestimmte Höhe bzw. Breite. Diese Eigenschaften kommen vor allem bei der Erstellung der verschiedenen CSS-basierten Layouts zum Einsatz. Aber auch für einfache Designaufgaben, wie die Definition der Breite einer Tabelle, das Anlegen einer simplen Seitenleiste oder die Erstellung einer Galerie aus Thumbnail-Bildern, können die Eigenschaften `height` und `width` hilfreich sein.

Die Erweiterung einer Stildefinition um diese Eigenschaften ist sehr leicht. Geben Sie einfach den Namen der Eigenschaft ein und weisen Sie ihr nach dem Doppelpunkt einen Wert in einer der CSS-gültigen Maßeinheiten zu, zum Beispiel:

```
width: 300px;
width: 30%;
height: 20em;
```

Die Verwendung von Pixeln ist auch hier recht einfach zu verstehen. Hiermit können Sie exakte Breiten- und Höhenangaben machen, die sich nicht verändern. Eine em-Einheit entspricht der Textgröße für ein bestimmtes Element. Beträgt die Textgröße beispielsweise 24 Pixel und Sie definieren eine Breite von 2em, hat das Element nun eine tatsächliche Breite von 2 x 24, also 48 Pixeln. Wenn Sie für das Element

keine eigene Textgröße definieren, basiert der berechnete Wert für eine em-Einheit auf der vererbten Textgröße.

Für die Eigenschaft `width` basieren Prozentwerte auf der Breite des umgebenden Elements. Legen Sie die Breite einer überschrift mit 75 Prozent fest und das <body>-Tag ist das einzige Vorfahren-Element, hat die überschrift 75 Prozent der Breite des Ansichtsbereichs. Ändert der Benutzer die Größe des Browserfensters, wird die Breite der überschrift automatisch angepasst. Befindet sich die überschrift aber z.B. innerhalb eines -Tag-Paars (vielleicht um eine Layoutspalte zu erzeugen) mit 200 Pixeln Breite, wird die überschrift prinzipiell 150 Pixel breit sein, egal wie groß das Browserfenster ist. Prozentwerte für die Eigenschaft `height` funktionieren so ähnlich, basieren aber auf der Höhe des umgebenden Elements.

Die tatsächliche Höhe und Breite einer Box berechnen

Auch wenn die Eigenschaften `width` und `height` sehr leicht verständlich erscheinen, haben sie dennoch ein paar Eigenarten, die viele Leute verwirren. So gibt es beispielsweise einen Unterschied zwischen dem Wert, den Sie als Breite für ein Element festlegen, und der Breite, die der Webbrowser letztendlich verwendet. Also noch einmal: Die Eigenschaften `width` und `height` definieren die Ausmaße des Inhaltsbereichs eines Elements. Dies ist der Bereich, in dem sich Text, Bilder oder andere Dinge befinden. Umgeben wird dieser Bereich von Innenabstand, Rahmen und Außenabstand. (Falls Ihnen nicht klar ist, was das bedeutet, werfen Sie noch einmal einen Blick auf Abbildung 1.) Deren Ausmaße müssen bei der Ermittlung der Gesamtbreite und -höhe in die Berechnung mit einfließen, wie in Abbildung 8 zu sehen ist.

Angenommen, Sie verwenden die folgenden Eigenschaften:

```
margin: 10px;
border-width: 5px;
padding: 15px;
width: 100px;
```

Wurde für die Eigenschaft `width` ein Wert festgelegt, wissen Sie immer, wie viel Platz für den Inhaltsbereich reserviert wurde – unabhängig von den Werten, die für andere Eigenschaften definiert wurden. Hierfür brauchen Sie keine Mathematikkenntnisse, weil der Wert von `width` genau der Breite des Inhaltsbereichs entspricht (im obigen Beispiel 100 Pixel). Wenn Sie allerdings herausfinden wollen, wie breit das gesamte Element inklusive Innen- und Außenabständen sowie Rahmen ist, kommen Sie um etwas Rechnerei nicht herum. Im obigen Beispiel hat das gesamte Element eine Breite von 160 Pixeln: je 10 Pixel für den linken und rechten Außenabstand, jeweils 5 Pixel für die linke und rechte Rahmenseite sowie je 15 Pixel für den linken und rechten Innenabstand und die 100 Pixel für die Breite des Inhaltsbereichs. (Internet Explorer 6 und seine Vorgänger tun sich hier ziemlich schwer, sodass Sie für diese Browser etwas zusätzliche Arbeit leisten müssen.)

Seien Sie bei der Definition von Höhenangaben für Ihre Seitenelemente vorsichtig. Sofern Sie nicht genau wissen, wie groß der Inhalt eines Elements sein wird, ist es äußerst schwer, dessen genaue Höhe vorherzusagen. Angenommen, Sie haben für eine **Pull Quote** eine Box von 100 x 100 Pixeln Größe angelegt. Enthält die Pull Quote zu viel Text, wird die Höhe der Box automatisch angepasst, was zu einigen seltsamen Problemen führen kann (s. Abbildung 9). Selbst wenn der Text eigentlich in den 100 Pixel hohen Bereich passt, kann der Benutzer Ihrer Site die Schriftgröße in seinem Browser erhöhen, und schon reicht der Platz möglicherweise nicht mehr aus.

Textüberlauf im Griff mit der Eigenschaft overflow

Wenn der Inhalt größer ist als die definierte Höhe oder Breite eines Elements, können seltsame Dinge passieren. Wie in Abbildung 9 zu sehen ist, vergrößern Internet Explorer 6 und seine Vorgänger die Box einfach, um den größeren Inhalt aufzunehmen. Bei anderen Browsern läuft die Box einfach über, und der Inhalt verdeckt möglicherweise den Inhalt anderer Elemente.

Zum Glück können Sie das Verhalten des Browsers in solchen Situationen steuern, indem Sie der Eigenschaft `overflow` einen entsprechenden Wert zuweisen. Sie können aus insgesamt vier Schlüsselwörtern wählen, die festlegen, wie der Inhalt in solchen Situationen dargestellt wird:

- `visible`. Diese Option entspricht dem Standardverhalten der Browser. Der Inhalt läuft einfach über die Grenzen des Elements hinaus (Abbildung 10 oben).
- `scroll`. Sorgt dafür, dass innerhalb des Elements Scrollbalken angezeigt werden, falls der Inhalt zu groß ist (Abbildung 10 Mitte). Es wird eine Art Mini-Browser-Fenster in Ihrer Seite erzeugt, das so ähnlich aussieht wie Oldschool-HTML-Frames oder die Standarddarstellung des `<iframe>`-Tags. Mit dem Schlüsselwort `scroll` ist es möglich, auch große Mengen Text auf kleinem Raum unterzubringen. Leider werden bei dieser Option die Scrollbalken immer angezeigt, selbst wenn der Inhalt dies eigentlich nicht erfordert.
- `auto`. Damit die Scrollbalken nur angezeigt werden, wenn sie wirklich gebraucht werden, verwenden Sie die Option `auto`. Sie funktioniert genau wie der Wert `scroll`, lediglich die Scrollbalken werden nur bei Bedarf angezeigt.
- `hidden`. Diese Option sorgt dafür, dass Inhalt, der über die Box hinausgeht, versteckt (nicht angezeigt) wird (Abbildung 10 unten). Diese Option ist ein wenig gefahrlich, weil möglicherweise wichtiger Inhalt einfach von der Seite verschwindet. Allerdings kann dieser Wert beim Umgehen einiger Programmierfehler im Internet Explorer hilfreich sein.

Das defekte Boxmodell des Internet Explorer 5 reparieren

Und hier ein paar schlechte Nachrichten: Internet Explorer 5.5 für Windows und frühere Versionen berechnen die Breite und Höhe nicht korrekt. Anstatt die Summe aus Innen- und Außenabständen, Rahmen und den Werten für `width` bzw. `height` zu verwenden, benutzt IE 5.5. – und IE 6 im „Quirks“-Modus nur den Außenabstand und die Breite/Höhe. Innenabstand und Rahmen werden als Teil der Werte

für `width` bzw. `height` angesehen, wodurch der Inhaltsbereich kleiner ist als in standardkonformen Browsern (s, Abbildung 11).

Der IE 5 berechnet die Box in Abbildung 8 folgendermaßen: Alle aktuellen Browser reservieren 160 Pixel für die Breite der Box. Im IE 5 werden die Innenabstände und Rahmen als Teil des Werts für `width` angesehen. Um die tatsächliche Breite des Inhaltsbereichs zu ermitteln, müssen Sie also die Werte für `padding` und `border-width` vom Wert für `width` subtrahieren: 100 (width) – 30 (padding-left und padding-right) – 10 (border-left-width und border-right-width) ergibt gerade noch 60 Pixel Breite für den Inhaltsbereich. Die Gesamtbreite für das Element ergibt sich im IE 5 aus dem Wert für `width` und den Werten für `margin-left` und `margin-right`: 100 + 10 + 10 {linker und rechter Außenabstand} = sondern auch niedriger als in standardkonformen Browsern, 120 Pixel Gesamtbreite. Das Gleiche gilt übrigens für die vertikalen Rahmen, Innen- und Außenabstände und die Höhe des Inhaltsbereichs. Die Box erscheint im IE 5 also nicht nur schmaler.

Dieses Problem lässt sich auf verschiedene Arten umgehen. Hier sind einige der am häufigsten verwendeten Methoden:

- Verwenden Sie für Elemente, die einen Wert für `height` bzw. `width` besitzen, keine Innenabstände oder Rahmen. Das Problem tritt nur auf, wenn für das Element Innenabstände oder Rahmen definiert werden. Wenn Sie diese Eigenschaften vermeiden, umgehen Sie das Problem komplett. Soll das Element mit einem dekorativen Rahmen versehen werden, kommen Sie allerdings um die Eigenschaft `border` kaum herum.
- Verwenden Sie zwei verschachtelte Tags, eines zur Definition von Breite und Höhe und ein zweites für Innenabstände und Rahmen. Angenommen, Sie wollen eine 100 Pixel breite Pull Quote {wörtlich Lockzitat, aus einem Text hervorgehobenes Zitat} erstellen und können auf Rahmen und Innenabstände nicht verzichten {und Sie müssen auch noch mit IE 5 klarkommen}. Dann können Sie für die Pull Quote folgenden HTML-Code verwenden:

```
<div class="pullquote">
<div class="pullquote_innen">

Dies ist der Inhalt der Pull Quote.

</div>
</div>
```

Im Stylesheet definieren Sie nun die folgenden Regeln:

```
.pullquote {
width: 100px;
}
.pullquote_innen {
padding: 10px;
background-color: #333;
border: 1px dotted red;
}
```

Die erste Regel legt die Breite des ersten (äußeren) `<div>`-Tags mit 100 Pixeln fest. Dies ist jetzt das Eltern-Element für die „inneren“ Tags der Pull Quote. Da hier nur die Breite definiert wurde, stellen alle Browser (inklusive IE 5) die Box mit einer Breite von 100 Pixeln dar. Das zweite (innere) `<div>`-Tag erhält die Werte für `border` und `padding`, aber keine Breite, sodass es auch hier keine Probleme mit dem IE 5 gibt. Diese Lösung ermöglicht die Verwendung von **Rahmen und Innenabständen** auf eine Weise, die mit allen Browsern funktioniert. Der Preis für die Unterstützung eines eigentlich toten Browsers ist der zusätzliche HTML-Code und ein eigentlich unnötiges `<div>`-Tag.

- Definieren Sie für den IE 5 eigene Werte für die Höhe und Breite. Viele Webentwickler bevorzugen diese elegantere Lösung, obwohl sie mit ein wenig Rechnerei verbunden ist. Die grundsätzliche Idee ist, dem Internet Explorer 5 und seinen Vorgängern eigene Werte für die Breite und Höhe eines Elements zu geben, die den Programmierfehler wieder ausgleichen. Für die Angabe spezieller Stile für bestimmte Browser gibt es mehrere Methoden. Ein einfacherer Weg ist die Verwendung des sogenannten Star-HTML-Hacks.

Sehen Sie sich noch einmal das Beispiel in Abbildung 8 an. Der normale Wert für `w\idth` beträgt 100px. Hinzu kommen insgesamt 30 Pixel für den linken und rechten Innenabstand sowie 10 Pixel für die linke und rechte Rahmenseite. Zusammen ergibt das 140 Pixel. In diesem Beispiel verwenden wir eine CSS-Klasse namens `.pullquote`. Diese Stildefinition enthält alle nötigen Einstellungen für Innen- und Außenabstände, Rahmen, Breite usw. Jetzt brauchen Sie nur einen speziellen Wert für `width`, damit auch der IE 5 mitspielt. Fügen Sie hierfür unterhalb von `.pullquote` folgenden Code ein:

```
html .pullquote {
width: 140px;
width: 100px;
}
```

Diese Regel muss unterhalb der eigentlichen `.pullquote`-Regel stehen, weil Sie den Wert für `width` überschreibt. Diese Regel ist nicht standardkonform und wird von allen Browsern außer Internet Explorer 6 und früher ignoriert. (Auch der IE 7 sieht sie nicht.) Die erste Deklaration setzt die Breite auf 140 Pixel (der korrekte Wert für IE 5). Da aber auch der IE 6 diesen neuen Wert erkennt, obwohl er (zumindest im sogenannten Standards compliant-Modus) das Boxmodell richtig interpretiert, muss er mit einem weiteren

Hack für den IE 6 wieder zurückgesetzt werden. Die Schreibweise `width` wird vom IE 5 ignoriert, daher verwenden wir sie, um für den IE 6 den richtigen Wert von 100 Pixeln wiederherzustellen – und alle sind glücklich (bis auf die armen **Webdesigner**, die die verrückten Hilfskonstruktionen lernen müssen).

- Ignorieren Sie den IE 5 komplett. Diese letzte Option klingt vermutlich ein wenig diskriminierend. Dennoch lohnt es sich, einen Blick in die Logdatei Ihres Webservers zu werfen, um festzustellen, wie viele Besucher Ihrer Site tatsächlich noch mit IE 5.5 oder früher unterwegs sind. Ende 2008 waren gerade mal 0.1 Prozent. Auch wenn dieses antiquierte Modell ein paar unverbesserliche Fans hat, wird ihre Zahl immer kleiner. Und wenn sich Ihre Site an Leute richtet, die sich ein wenig mit Technik auskennen, brauchen Sie vermutlich überhaupt keinen Gedanken mehr an den IE 5.5 zu verschwenden. (Sie Glückliche/r!)

Elemente umfließen lassen

Der normale HTML-Textfluss im Browserfenster verläuft von links nach rechts und von oben nach unten, ein Block-Element (überschrift, Absatz. was auch immer) nach dem anderen. Diese Art der Darstellung, die an ein Textverarbeitungsprogramm erinnert, ist visuell eher langweilig (s. Abbildung 12 oben). Mit CSS können Sie das aber leicht ändern. Aber schon eine kleine CSS-Eigenschaft kann ein eine Menge bewirken: `float`.

Mithilfe der Eigenschaft `float` können Sie beliebige Elemente nach links oder rechts verschieben. Hierbei werden sie vom übrigen Inhalt der Seite umflossen (s. Abbildung 12 unten). Umflossene Elemente, sogenannte Floats, sind hervorragend geeignet, um zusätzliche Informationen aus dem Haupttext einer Seite herauszubewegen. Dieses Verhalten kennen Sie von Bildern, die an eine Seite des Ansichtsbereichs verschoben werden, während sie der Text auf der anderen Seite elegant umfließt. Mithilfe der Eigenschaft `float` können Sie das gleiche Verhalten z.B. für Seitenleisten oder Blöcke mit Zusatzinformationen definieren.

Wie Sie später sehen werden, können Floats auf ziemlich komplexe (und verwirrende) Weise benutzt werden. Die grundsätzliche Eigenschaft ist aber recht einfach zu verstehen. Als Wert wird eines der vier Schlüsselwörter `left`, `right`, `none` oder `inherit` verwendet, wie hier:

```
float: left;
```

- `left`. Das betreffende Element wird nach links verschoben. Der restliche Inhalt der Seite umfließt das Element auf der rechten Seite.
- `right`. Verschiebt das Element nach rechts. Der restliche Inhalt der Seite umfließt das Element auf der linken Seite.
- `none`. Dieser Wert verhindert, dass ein Element verschoben wird.
- `inherit`. Das betreffende Element erbt den Wert von seinem Vorfahren-Element.

Ist kein Wert definiert, wird der Standardwert `none` verwendet.

Info

Das Umfließen eines Bilds ähnelt stark der Verwendung des HTML-Attributs `align` mit den Werten `left` oder `right` für das `<img>`-Tag. Die Verwendung des Attributs gilt aber mittlerweile als veraltet. Stattdessen sollten Sie die Eigenschaft `float` verwenden, sofern Sie mit den Standardeinstellungen des Browsers nicht einverstanden sind.

Floats bewegen sich an den linken oder rechten Rand des Ansichtsbereichs oder ihres umgebenden Elements, falls es eines gibt. Vielleicht enthält Ihre Seite einen Kasten mit 300 Pixeln Breite, der mit der Deklaration `float: right;` an die rechte Kante des Ansichtsbereichs verschoben wurde. Innerhalb dieser Box gibt es wiederum ein Bild, das per `float: left;` nach links verschoben wurde. In dieser Situation wird das Bild bündig mit der linken Seite der Box und nicht des Ansichtsbereichs dargestellt.

Sie können die Eigenschaft `float` sogar mit Inline-Elementen benutzen, z.B. mit dem `<img>`-Tag. Tatsächlich ist die Definition von Bildern als Floats eine der häufigsten Anwendungen von float. Sobald ein Inline-Element als Float definiert wurde, behandelt es der Webbrowser wie ein Block-Element. Die Probleme mit Innen- und Außenabständen für Inline-Elemente können hier also nicht auftreten.

Natürlich können Sie auch überschriften oder Absätze als Float definieren – häufiger kommt diese Technik jedoch zusammen mit `<div>`-Elementen zum Einsatz. Das `<div>` dient hierbei als eigenständiges umgebendes Element für die enthaltenen HTML-Tags. Auf diese Weise lassen sich Seitenleisten, Pull Quote und andere in sich geschlossene Elemente einer Webseite anlegen. Wenn Sie ein Element als Float definieren, ist es vorteilhaft, auch einen Wert für die Eigenschaft `width` zu setzen. Auf diese Weise können Sie steuern, wie viel horizontalen Platz das Element einnimmt und wie viel Platz der übrige Inhalt hat, um das Float zu umfließen.

Info

Die sogenannte Quellcode-Reihenfolge (source order), also die Reihenfolge der HTML-Tags in Ihrer Seite, hat einen großen Einfluss auf die Darstellung von Floats. Das HTML-Tag, das umflossen werden soll, muss im Quellcode vor den Elementen stehen, die es umfließen sollen. Nehmen wir zum Beispiel eine Webseite, in der auf eine überschrift (`<h1>`) ein Absatz (`<p>`) folgt. Der Absatz enthält gegen Ende ein Bild, das als Float definiert wird. In diesem Fall erscheinen die überschrift und der größte Teil des Absatzes weiterhin oberhalb des Bilds, denn nur Inhalt, der auf das `<img>`-Tag folgt, kann das Bild auch umfließen.

Hintergründe, Rahmen und Floats

Sehr zur Bestürzung vieler Webdesigner funktionieren Rahmen und Hintergründe bei der Verwendung von Floats nicht so wie erwartet. Wenn Sie ein Element als rechten Float definieren (z.B. eine Seitenleiste), bewegt sich der darunter befindliche Inhalt nach oben und umfließt wie erwartet das Float. Wurde für den Inhalt allerdings ein Hintergrund oder ein Rahmen definiert, wird dieser nun vom Float überlagert, anstatt diesem Platz zu machen (Abbildung 13 links). Und ob Sie es glauben oder nicht – dieses Verhalten ist

absolut korrekt und regelkonform. Vielleicht wollen Sie sich aber nicht an diese Regel halten, sondern sehen es lieber, wenn auch Hintergründe und Rahmen dem Float Platz machten (Abbildung 13 rechts). Alles was Sie hierfür brauchen, ist ein wenig CSS-Magie.

Als Erstes müssen Sie für die Regel, die den Hintergrund oder den Rahmen definiert, eine zusätzliche Deklaration einfügen: `overflow: hidden;`. Die Eigenschaft `overflow` sorgt dafür, dass Rahmen und Hintergründe nicht mehr vom Float überlagert werden.

Eine weitere Möglichkeit besteht darin, das Float selbst mit einem Rahmen zu umgeben. Wenn Sie die Linie dick genug machen und die gleiche Farbe verwenden wie für den Seitenhintergrund, erscheint der Rahmen einfach wie Leerraum, obwohl eigentlich Hintergrundfarbe und Rahmen damit verdeckt werden.

Umfließen verhindern

In manchen Fällen sollen bestimmte Tags einen Float nicht umfließen, sondern auf jeden Fall auf einer eigenen Zeile stehen, z.B. ein Copyright-Hinweis, der prinzipiell für sich am Ende der Seite stehen soll. Wenn Sie jetzt eine besonders lange als Float definierte Seitenleiste haben, kann es passieren, dass der Copyright-Hinweis nach oben verschoben wird und neben der Seitenleiste angezeigt wird. Sie wollen aber, dass er auf jeden Fall unterhalb der Seitenleiste angezeigt wird und kein anderes Element umfließen soll.

Ein anderes Problem tritt auf, wenn Sie mit mehreren umflossenen Elementen arbeiten, die nahe beieinander stehen. Wenn die Floats sehr breit sind, versuchen sie, sich gegenseitig zu umfließen, und es kommt zum Stau (Abbildung 14 oben). In diesem Fall sollten sich die einzelnen Floats aber untereinander befinden. Für Situationen wie diese gibt es die CSS-Eigenschaft `clear`.

Die Eigenschaft `clear` weist ein Element an, ein als Float definiertes Element nicht zu umfließen. Wird für ein Tag die Eigenschaft `clear` definiert, wird es quasi gezwungen, eine Position unterhalb des davorstehenden Floats einzunehmen. Sie können außerdem festlegen, welche Art von Float (links oder rechts) nicht umflossen werden soll oder einfach alle Floats zu ignorieren.

Die Eigenschaft `clear` kann die folgenden Werte annehmen:

- `left`. Das Element wird unterhalb von links positionierten Floats angezeigt, umfließt aber rechts positionierte Floats weiterhin.
- `right`. Das Element wird unterhalb von rechts positionierten Floats angezeigt, umfließt aber links positionierte Floats weiterhin.
- `both`. Das Element wird prinzipiell unterhalb von rechts oder links positionierten Floats platziert, ohne diese zu umfließen.
- `none`. Diese Einstellung deaktiviert die Verschiebung nach unten komplett und sorgt dafür, dass das Element linke und rechte Floats umfließt. Dies ist das Standardverhalten.

Im Fall des Copyright-Hinweises, der auf jeden Fall am Schluss der Seite erscheinen soll, müssen Sie die Eigenschaft `clear` für Floats auf beiden Seiten aktivieren, sodass ein Umfließen anderer Elemente aufjeden Fall verhindert wird. Hier eine CSS-Klasse, die genau das tut:

```
.copyright {
clear: both;
}
```

In Abbildung 14 sehen Sie, wie die Eigenschaft `clear` verhindern kann, dass Floats unterschiedlicher Höhe miteinander „verklumpen“. Alle drei Fotos wurden als rechter Float defIniert. In der oberen Abbildung ist das Foto von den Tomaten (1) das erste Bild auf der Seite. Es erscheint direkt am rechten Seitenrand. Das zweite Bild (2) respektiert den Wert für den Float des ersten Bilds und versucht, es links zu umfließen. Das dritte Foto (3) ist zu breit, um neben den anderen beiden Bildern zu stehen, versucht aber (1) und (2) zu umfließen. Leider sind die Bemühungen nur teilweise erfolgreich.

Durch die Verwendung von `clear: right;` für die Bilder verhindern Sie, dass die Fotos alle nebeneinander dargestellt werden (s. Abbildung 14 unten). Die Definition von `clear` für das zweite Bild sorgt dafür, dass es auf jeden Fall unter dem ersten Bild angezeigt wird. Entsprechendes gilt für das Foto darunter: Auch hier verhindert `clear: right;` das „Verklumpen“ und sorgt dafür, dass Bild (3) ebenfalls unter den anderen beiden dargestellt wird.

Info

Die Verwendung von linken und rechten Floats und das Verhindern des Umfließens erscheint auf den ersten Blick vielleicht etwas kompliziert. Und das ist es auch. In diesem Abschnitt geben wir Ihnen daher nur einen kurzen Einblick.

Neben den normalen Varianten zur Formatierung, die Sie bisher kennengelernt haben, gibt es einige weitere spezielle Formatierungsarten.

Besonders in solchen Fällen, wenn Sie mehreren Elementen die gleiche Schriftart, Farbe oder Ähnliches zuweisen möchten, wäre es sehr angenehm, eine verkürzende Schreibweise verwenden zu können. Dies ist möglich, wenn Sie bei der Notation des Selektors mehrere Selektoren durch Kommata trennen.

```
Selektor1, Selektor2, SelektorN {
Eigenschaften;
}
```

Dies ist besonders bei ähnlichen Elementen wie z.B. h1 bis h6 sinnvoll. Jedem Element nun einzeln die Schriftart Arial zuzuweisen, würde den Stylesheet-Umfang versechsfachen.

```
h1 {
font-family:Arial,sans-serif;
}
h2 {
font-family:Arial,sans-serif;
}
h3 {
font-family:Arial,sans-serif;
}
h4 {
font-family:Arial,sans-serif;
}
h5 {
font-family:Arial,sans-serif;
}
h6 {
font-family:Arial,sans-serif;
}
```

Die kürzere Schreibweise lautet nun:

```
h1, h2, h3, h4, h5, h6 {
font-family:Arial,sans-serif;
}
```

Ein vollständiges Beispiel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Gruppenformatierungen -->
<html>
<head>
<title>Spezielle Formatierungen</title>
<style type="text/css">
h1, h2, h3, h4, h5, h6 {
font-family:Arial,sans-serif;
}
</style>
</head>
<body>
```

```
<h1>Überschrift der 1. Ordnung</h1>
<h2>Überschrift der 2. Ordnung</h2>
<h3>Überschrift der 3. Ordnung</h3>
<h4>Überschrift der 4. Ordnung</h4>
<h5>Überschrift der 5. Ordnung</h5>
<h6>Überschrift der 6. Ordnung</h6>
</body>
</html>
```

Info

Dies lässt sich natürlich auch mit allen anderen Elementen durchführen. Sie können also auch h1–h6-Elemente mit p-Elementen und kbd-Elementen gruppieren.

CSS-Formatierungen lassen sich auch auf Elemente begrenzen, bei denen z.B. ein bestimmtes Attribut oder ein Attribut mit einem speziellen Parameter gesetzt wurde. Ein Textabsatz, der mit dem Attribut `align` ausgerichtet wurde, könnte, abhängig von seiner Ausrichtung, in einer unterschiedlichen Textfarbe dargestellt werden.

Die Formatierungsbedingung wird in eckigen Klammern direkt nach dem Selektor notiert (also noch vor der geöffneten geschweiften Klammer). Wenn eine Überschrift `h1` z. B. mit dem Attribut `align` ausgerichtet wurde und deshalb eine andere Schriftfarbe bekommen soll, würde die Anweisung wie folgt lauten:

```
h1 {
color:#336699;
}
h1[align] {
color:#993366;
}
```

In diesem Beispiel würden alle Überschriften der 1. Ordnung die Farbe #336699 zugewiesen bekommen, es sei denn, sie wurden explizit mit dem Attribut `align` ausgerichtet. Dann bekommen sie die Farbe #996633.

Genauso gut ließe sich die Formatierung mit der Farbe #996633 nur auf Überschriften der 1. Ordnung anwenden, die mit dem `align`-Attribut zentriert (`center`) ausgerichtet wurden.

```
h1 {
color:#336699;
}
```

```
h1[align=center] {
color:#993366;
}
```

Der Parameter würde dann hinter dem Attributbezeichner stehen, getrennt durch das Gleichheitszeichen und ohne jegliche Anführungszeichen.

Sie können sich bei dem Parameter aber auch lediglich auf eine Zeichenkette beschränken, die innerhalb des Attributs vorkommt. Dabei müssen Sie aber unterscheiden, ob die Zeichenketten im Attribut mit Leerzeichen oder Bindestrichen getrennt werden. Dementsprechend müssen Sie eine andere Schreibweise verwenden. Bei einem Leerzeichen als Trennzeichen müssen Sie ~= und bei Bindestrichen |= verwenden.

```
p[name~=Text] {
background-color:#336699;
color:#FFFFFF;
}
p[name|=Absatz] {
font-variant:small-caps;
}
```

Möchten Sie alle HTML-Elemente zur Formatierung ansprechen, können Sie den Platzhalter * verwenden. Die Angabe

```
*[align=center] {
color:#996633;
}
```

würde allen HTML-Elementen, die mit dem `align`-Attribut zentriert ausgerichtet wurden, die Farbe #996633 zuweisen.

Info

Den Platzhalter * können Sie natürlich auch für andere Formatierungen verwenden, die nicht attributbedingt sind. So würde `* { font-family:Arial; }` allen HTML-Elementen die Schriftart Arial zuweisen. Achtung: Dies gilt auch für das `pre`-Element!

Attributbedingte Formatierungen gehören zum CSS 2.0-Standard.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- attributbedingte Formatierung -->
<html>
```

```
<head>
<title>Spezielle Formatierungen</title>
<style type="text/css">
*[align] {
font-family:sans-serif;
}
p[align=center] {
background-color:#00FFFF;
}
p[name~=Text] {
background-color:#336699;
color:#FFFFFF;
}
p[name|=Absatz] {
font-variant:small-caps;
}
</style>
</head>
<body>
<h1 align="center">Attributbedingte Formatierung</h1>
<p>Dies ist ein normaler Textabsatz.</p>
<p align="right">Rechtsbündig ausgerichteter Textabsatz.</p>
<p align="center">Zentrierter Textabsatz</p>
<p name="Text Absatz">Der Name dieses Absatzes lautet: <Text Absatz>.</p>
<p name="Text-Absatz">Der Name dieses Absatzes lautet etwas anders: <Text-
Absatz></p>
</body>
</html>
```

Klassen

In HTML gibt es das Universalattribut `class`, das Sie im Start-Tag eines jeden Elements notieren können. So können Sie z.B. verschiedene HTML-Elemente zu einer Klasse zusammenfassen. Diese Elemente gehören dann logisch zusammen. Auch Elemente der gleichen Sorte lassen sich zu Klassen zusammenfassen. Mit CSS können Sie nun für solche Klassen spezielle Formatierungen anwenden. Um allen p-Elementen, die zur Klasse absatz gehören, die Farbe #00FF00 zuzuweisen, müssen Sie in CSS zuerst den Selektor p notieren und dann, getrennt durch einen Punkt, die Klasse.

```
p.absatz {
color:#00FF00;
}
```

Jedes p-Element, das im Attribut `class` den Parameter `absatz` zugewiesen bekommen hat, wird mit der Textfarbe #00FF00 dargestellt.

```
<p class="absatz">Textabsatz der Klasse absatz</p>
```

Wenn zu einer solchen Klasse unterschiedliche HTML-Elemente gehören und die CSS-Formatierung für alle angewendet werden soll, lassen Sie den Selektor weg, oder Sie notieren den Platzhalter *.

```
.absatz {
color:#00FF00;
}
*.absatz {
color:#00FF00;
}
```

Zur Übersicht folgt nun noch einmal ein vollständiges Beispiel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Formatierung mit Klassen -->
<html>
<head>
<title>Klassen</title>
<style type="text/css">
.rot {
color:#FF0000;
}
*.gruen {
color:#00FF00;
}
p.fett {
font-weight:bold;
}
</style>
</head>
<body>
<h1 class="rot">Formatierung mit Klassen</h1>
<p class="rot">Dieser Absatz ist rot.</p>
<p class="fett">Dieser Absatz ist fett gedruckt.</p>
<p class="gruen">Dieser Absatz ist grün.</p>
</body>
</html>
```

Info

Achten Sie bei der Verwendung von Klassen darauf, dass Sie z.B. nicht allen Überschriften-Elementen die gleiche Schriftgröße zuweisen. Dies würde den Sinn und Zweck von Überschriften zur Textstrukturierung zunichte machen.

Verschachtelte Elemente

Gelegentlich kann es sinnvoll sein, Formatierungen abhängig von der Verschachtelung der HTML-Elemente zu setzen, z. B. kann das em-Element sowohl im Gültigkeitsbereich des h1-Elements als auch des p-Elements notiert werden. Davon abhängig, können Sie das em-Element unterschiedlich formatieren. Während es innerhalb eines p-Elements den Text fett, kursiv und in einer blauen Schriftfarbe darstellen soll, könnte es im h1-Element den Text rot und normal darstellen. Um eine solche Formatierung in CSS zu notieren, müssen Sie zuerst den Selektor des Elternelements und anschließend den Selektor des Kindelements notieren. Beide Selektoren werden durch ein Leerzeichen voneinander getrennt.

```
SelektorE SelektorK {
Eigenschaften;
}
```

Die Syntax für das Beispiel von eben sieht dann wie folgt aus:

```
h1 em {
font-style:normal;
color:#FF0000;
}
p em {
font-weight:bold;
color:#0000FF;
}
```

Solche Anweisungen gehören zum CSS 1.0-Standard und werden vom Internet Explorer und Netscape ab Version 4 interpretiert. Die Bedingungen lassen sich nach dem CSS 2.0-Standard noch genauer festlegen, und zwar durch spezielle Zeichen, die zwischen den beiden Elementen notiert werden.

```
p > em {
color:#FF0000;
}
```

Diese Anweisung bedeutet, dass ein mit dem Element em ausgezeichneter Text nur dann rot dargestellt wird, wenn er direkt ein Kindelement des p-Elements ist, z.B.

```
<p>...<em>...</em>...</p>.

p * em {
color:#00FF00;
}
```

Der mit dem em-Element ausgezeichnete Text wird nur dann grün dargestellt, wenn das Element mindestens zwei Ebenen nach dem p-Element notiert wurde, z.B.

```
<p>...<i>...<em>...</em>...</i>...</p>.

p + em {
color:#0000FF;
}
```

Der mit em ausgezeichnete Text wird erst dann blau dargestellt, wenn das em-Element nach dem p-Element notiert wurde bzw. auf einer gleichen Ebene liegt, z.B.

```
<p>...</p><em>...</em>.
```

Info

Die Eingrenzungen, die durch die CSS 2.0-Bedingungen „>“ und „*“ möglich sind, werden vom IE ab Version 5 und ab dem Netscape 6 interpretiert. Die Anweisung „+“ funktioniert lediglich ab dem Netscape 6.

Das folgende Listing stellt alle Bedingungen noch einmal in einem vollständigen Beispiel dar.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Formatierungen mit Bedingung -->
<html>
<head>
<title>Verschachtelte Elemente</title>
<style type="text/css">
h1 em {
font-style:normal;
color:#FF0000;
}
```

```
p em {
font-weight:bold;
color:#0000FF;
}
p > em {
color:#FF0000;
}
p * em {
color:#00FF00;
}
p + em {
color:#0000FF;
}
</style>
</head>
<body>
<h1>Formatierungen mit <em>Bedingung</em></h1>
<p>Dieser Textabsatz stellt die Anweisung <em>p > em</em> dar.</p>
<p>Dieser Textabsatz stellt <i>die Anweisung <em>p * em</em> dar.</i></p>
<p>Dieser Textabsatz stellt die Anweisung p + em dar.</p><em>blau</em>
</body>
</html>
```

Das Beispiel aus dem Code stellt die unterschiedlichen Formatierungsmöglichkeiten von verschachtelten Elementen dar.

Individuelle Formate

Mit individuellen Formaten können Sie Formatierungen festlegen, die nur auf ein spezielles Element angewendet werden dürfen. Mit dem Universalattribut `id` können Sie einem HTML-Element eine Identifikation (ID) zuweisen. Da der Parameter, der dem Attribut `id` zugewiesen worden ist, nur einmal vorkommen darf bzw. sollte, können Sie anhand dieser ID ein Individualformat definieren.

Die ID müssen Sie in CSS mit dem Präfix # als Selektor notieren, in geschweiften Klammern dahinter die Eigenschaften.

```
#id {
Eigenschaften;
}
```

Für das Element p mit dem Parameter ersterAbsatz im id-Attribut würde der Selektor #ersterAbsatz lauten. Außerdem können Sie festlegen, dass ein Individualformat nur dann zugewiesen wird, wenn die ID zu einem bestimmten Element gehört.

```
h1#nummerEins {
color:#FF0000;
}
```

Dieses Individualformat wird nur dann zugewiesen, wenn das Element mit der ID nummerEins ein h1-Element ist. Ansonsten würde z.B. ein p-Element mit der ID nummerEins von dieser Formatierung ausgenommen werden.

Solche Formate werden vom IE ab Version 3 und von Netscape ab Version 6 unterstützt.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Individualformat -->
<html>
<head>
<title>Individuelle Formate</title>
<style type="text/css">
h1#eins {
color:#800000;
}
#zwei {
color:#008000;
}
#drei {
color:#000080;
}
</style>
</head>
<body>
<h1 id="eins">Roter Text</h1>
<p id="zwei">Grüner Text</p>
<p id="drei">Blauer Text</p>
</body>
</html>
```

Testen Sie einmal Folgendes: Ersetzen Sie in der Stylesheet-Definition den Selektor #zwei durch den Selektor h1#zwei, und testen Sie das Listing erneut im Browser. Sie werden feststellen, dass der Text des p-Elements mit der ID zwei nun nicht mehr in einer grünen Schriftfarbe dargestellt wird.

Syntaxregeln für CSS-Formatdefinitionen

Betrachten wir nun den Aufbau der Formatdefinitionen. Die gesamte CSS-Syntax steht in der Wertzuweisung an das `style`-Attribut. Für die Formatdefinition gelten folgende wichtige Regeln:

- Jede Formatdefinition hat die Form Eigenschaft:Wert. Ein Doppelpunkt trennt den Namen der CSS-Eigenschaft von der Wertzuweisung.
- Wenn mehr als ein Eigenschaft-Wert-Paar notiert wird, müssen alle außer dem letzten jeweils mit einem Semikolon (`;`) abgeschlossen werden. Wird nur ein Eigenschaft-Wert-Paar notiert, darf es ebenfalls ein Semikolon am Ende erhalten (muss aber nicht). Das Gleiche gilt für das letzte von mehreren notierten Eigenschaft-Wert-Paaren.
- Bei den Namen von CSS-Eigenschaften spielt Groß-/Kleinschreibung keine Rolle. Üblich ist es jedoch, alle CSS-Eigenschaften klein zu schreiben.
- Vor und nach dem Doppelpunkt zwischen Eigenschaft und Wert dürfen auch Leerzeichen notiert werden.
- Wenn eine Eigenschaft mehrere Werte zugewiesen bekommt und Werte dabei selbst Leerzeichen enthalten, sollten die Werte in Anführungszeichen (`""`) gesetzt werden.

```
font-family:"MS Sans Serif", Helvetica, sans-serif;
```

Info

Die Namen der möglichen CSS-Eigenschaften sowie das Format oder das Schema der erlaubten Werte, die den Eigenschaften zugewiesen werden können, sind in der CSS-Spezifikation des W3-Konsortiums festgelegt.

In unserem Beispiel aus dem Artikel style-Attribut sind vermutlich noch einige Fragen zu den Wertzuweisungen an die CSS-Eigenschaften offen geblieben. Der Quelltext enthält CSS-Wertzuweisungen wie `background-color:#FFFFC0`, `color:red`, `border-bottom:solid blue 4px` oder `font-size:15px`. Daran ist zu erkennen, dass es bestimmte Maßeinheiten und Schemata für die Notation von Werten gibt. In den beiden folgenden Abschnitten werden wir zunächst auf die Regeln bei den Wertzuweisungen eingehen, bevor wir daran anschließend einzelne CSS-Eigenschaften beschreiben.

Die Formatierungsfahigkeiten von **CSS** reichen weit über Text, Bilder und Links hinaus. Rahmen, Hintergründe und andere visuelle Verbesserungen sind auch für Tabellen möglich. z.B. für Sportergebnisse, Musik-Playlisten usw. Ähnliches gilt für Formulare. Hier können Sie CSS einsetzen, um die einzelnen Elemente so anzuordnen und zu gestalten, dass die Bestellung von Artikeln, das Abonnement eines Newsletters oder die Benutzung Ihrer neuesten Webapplikation erleichtert wird.

Dieses Kapitel zeigt Ihnen, wie Sie **Tabellen** und **Formulare** mit HTML erstellen und mit CSS gestalten können.

Tabellen richtig verwenden

In der kurzen Geschichte des Webs spielten Tabellen oft eine wichtige Rolle. Ursprünglich war es ihre Aufgabe, Daten nach Zeilen und Spalten geordnet darzustellen. Häufiger wurden sie aber als Layoutwerkzeug benutzt. Durch die damaligen Einschränkungen von **HTML** mussten die Webdesigner kreative Wege finden, um ihr Layout umzusetzen. Mithilfe der Zeilen und Spalten in Tabellen konnten sie Elemente wie Überschriften und Seitenleisten präzise positionieren. Wie Sie sehen werden, ist CSS für das Layout von Webseiten jedoch wesentlich besser geeignet. Sie können sich bei der Benutzung von Tabellen auf deren eigentlichen Zweck beschränken: die Darstellung von Daten (s. Abbildung 1).

HTML (und XHTML) enthält eine erstaunliche Anzahl von Tags zur Erstellung von Tabellen. Der unten stehende Code erzeugt eine sehr einfache Tabelle (s. Abbildung 2).

```
<table>
<caption align="bottom">
Tabelle 1: Der richtige Rasenmäher für Ihre Wohnung
</caption>
<colgroup>
<col id="marke" />
<col id="preis" />
<col id="antrieb" />
</colgroup>
<thead>
<tr>
<th scope="col">Marke</th>
<th scope="col">Preis</th>
<th scope="col">Anstrieb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinook Schiebomat Innenmäher</td>
<td>247,00 €</td>
<td>mechanisch</td>
</tr>
<tr>
<td>Sampson Deluxe Appartement-Mäher</td>
<td>370,00 €</td>
```

```
<td>mechanisch</td>
</tr>
</tbody>
</table>
```

Schon für drei Zeilen und drei Spalten werden hier neun verschiedene **HTML-Tags** benutzt: `<table>`, `<caption>`, `<colgroup>`, `<col>`, `<thead>`, `<tbody>`, `<tr>`, `<th>` und `<td>`. Normalerweise ist es nicht besonders günstig, viele HTML-Tags zu verwenden. Allerdings geben Ihnen die zahlreichen Tabellen-Tags sehr detaillierte Möglichkeiten für den gestalterischen Einsatz von CSS. Da für die Tabellenüberschrift ein anderes Tag verwendet wird (`<th>`) als für die Tabellenzellen (`<td>`), können Sie einfach per Typ-Selektor (`th {...}`) einen eigenen Stil fur die Überschrift definieren. Eine spezielle Klasse wie .tabellen_ueberschrift ist also nicht nötig. Im folgenden Abschnitt werden Sie sehen, wie Sie die verschiedenen Tags zu Ihrem Vorteil nutzen können.

Tabellen mit Stil

Die meisten bereits bekannten CSS-Eigenschaften können Sie auch für Tabellen und ihren Inhalt benutzen. So bestimmt wie gehabt die Eigenschaft `color` die Farbe der in der Tabelle verwendeten Schrift. Allerdings gibt es auch ein paar Eigenschaften, die für Tabellen besonders sinnvoll sind, und einige, die nur für die Formatierung von Tabellen benutzt werden.

Da Tabellen aus mehreren verschachtelten HTML-Tags bestehen, ist es hilfreich, wenn man weiß, welche CSS-Eigenschaften zu welchen Tags passen. Beispielsweise hat die Definition von `padding` für das `<table>`-Tag keine Bedeutung. In den folgenden Abschnitten stellen wir Ihnen die CSS-Eigenschaften für die Tabellenformatierung vor und zeigen Ihnen, welche Tags dafür infrage kommen.

Innenabstände

Wie Sie schon wissen, ist der Innenabstand als Leerraum zwischen dem Inhaltsbereich und dem Rahmen eines Elements definiert. Bei Tabellen werden die Tabellenzellen von Rahmen begrenzt. Durch Innenabstände können Sie also bestimmen, wie viel Platz zwischen dem Inhalt einer Zelle und ihrem Rahmen sein soll (s. Abbildung 2). Das Resultat ähnelt dem des HTML-Attributs `cellpadding` für das `<table>`-Tag. Allerdings haben Sie bei CSS den Vorteil, dass Sie den Abstand für jede Seite (links, rechts, oben, unten) einzeln definieren können.

Der Innenabstand wird entweder für Tabellenüberschriften oder einzelne Tabellenzellen definiert, aber nicht für das `<table>`-Tag selbst. Um Tabellenzellen mit einem Innenabstand von 10 Pixeln zu versorgen, könnten Sie beispielsweise die folgende Regel verwenden:

```
td, th {
padding: 10px;
}
```

Wie gesagt, Sie können die Innenabstände für die einzelnen Seiten auch separat steuern. Mit der folgenden Regel erhält die Zelle einen oberen Innenabstand von 10 Pixeln, die linke Seite 3 Pixel, und die rechte Seite sowie die Unterseite erhalten jeweils 5 Pixel:

```
td {
padding-top: 10px;
padding-right: 5px;
padding-bottom: 3px;
padding-left: 5px;
}
```

Info

Wenn Sie ein Bild in einer Tabellenzelle platzieren, werden Sie feststellen, dass unterhalb des Bilds zu viel Leerraum ist. Dies können Sie unterbinden, wenn Sie für das Bild die Eigenschaft `display` auf den Wert `block` setzen.

Vertikale und horizontale Ausrichtung anpassen

Um zu bestimmen, wie der Inhalt innerhalb der Tabellenzelle ausgerichtet wird, können Sie die Eigenschaften `text-align` und `vertical-align` verwenden. Hierbei steuert `text-align` mit den Werten `left` (linke Ausrichtung), `right` (rechte Ausrichtung), `center` (horizontal zentriert) und `justify` (Blocksatz) die horizontale Ausrichtung. Da der Wert von `text-align` vererbt wird, können Sie die Ausrichtung aller Zellen einer Tabelle mit der folgenden Regel steuern:

```
table {
text-align: right;
}
```

Diese Eigenschaft eignet sich gut für `<th>`-Tags, da Browser Tabellenüberschriften oftmals automatisch zentrieren. Mit einer einfachen Regel wie `th { text-align; left; }` können Sie das wieder rückgängig machen.

Für Tabellenzellen kann eine bestimmte Höhe definiert werden. Standardmäßig richten Webbrowser den Inhalt vertikal in der Zellenmitte aus (s. Beispiel zum Wert `middle` in Abbildung 4). Die vertikale Ausrichtung im `<td>`-Tag lässt sich mit der Eigenschaft `vertical-align` einstellen, indem Sie einen dieser vier Werte verwenden: `top` (oben), `baseline` (Grundlinie), `middle` (mittig) oder `bottom` (unten). Der Wert `top` verschiebt den Inhalt an die Oberkante der Zelle, `middle` zentriert ihn in der Mitte, und `bottom` schiebt ihn an die Unterkante. Der Wert `baseline` funktioniert ähnlich wie `top`, nur richtet der

Browser hier die Grundlinie an der ersten Textzeile aus (s. Abbildung 4). (Sie und Ihre Besucher werden den subtilen Unterschied nur bemerken, wenn Sie echte Perfektionisten sind.) Im Gegensatz zu `text-align` wird `vertical-align` nicht vererbt. Sie müssen die Eigenschaften also direkt für die <th>– und <td>-Tags definieren. (Für `vertical-align` gibt es noch zwei weitere Werte: `text-top` und `text-bottom`. Diese werden aber so selten benutzt, dass wir hier nicht weiter darauf eingehen.)

Info

Die bisher gezeigten CS5-Beispiele wurden auf alle Tabellen einer Seite angewandt Wenn Sie nur bestimmte Tabellen (oder Zellen) formatieren wollen, müssen Sie einen anderen Selektor verwenden. Soll eine bestimmte Tabelle speziell gestaltet werden, weisen Sie ihr eine eigene Klasse zu, z.B. `<table closs="aktienkurse">`. Mithilfe von Nachfahren-Selektoren wie .aktienkurse td oder .aktienkurse th können Sie die Zellen dieser Tabelle nun getrennt vom Rest der Seite formatieren.

Rahmen anlegen

Die CSS-Eigenschaft `border` funktioniert für Tabellen im Prinzip genau wie für andere Elemente auch. Allerdings gibt es ein paar Besonderheiten zu beachten. Wenn Sie für das `<table>` -Tag einen Rahmen definieren, gilt dieser nur für die Außenkanten der Tabelle, aber nicht für die einzelnen Zellen. Weisen Sie den einzelnen Zellen einen Rahmen zu (z.B. `td { border: 1px solid black; }`), wird standardmäßig zwischen den einzelnen Zellen eine kleine Lücke gelassen, wie in Abbildung 5 oben. Um dieses Verhalten zu steuern, müssen Sie wissen, wie das HTML-Attribut `cellspacing` bzw. die CSS-Eigenschaften `border-spacing` und `border-collapse` funktionieren.

- Den Leerraum zwischen den Tabellenzellen kontrollieren.

Standardmäßig stellen Browser die einzelnen Zellen um ein paar Pixel voneinander entfernt dar. Diese Lücke wird besonders gut sichtbar, wenn Sie die Zellen mit einem Rahmen versehen. Anhand der CSS-Eigenschaft `border-spacing` können Sie selbst bestimmen, wie groß dieser Zwischenraum sein soll. Da diese Eigenschaft aber erst ab der Version 8 vom Internet Explorer unterstützt wird, ist es möglicherweise sinnvoll, für ältere IE-Versionen das HTML-Attribut `cellspacing` zu verwenden. Um einen Zellenzwischenraum von 10 Pixeln zu definieren, könnten Sie beispielsweise folgenden HTML-Code verwenden: `<table cellspacing="10">`. (Wenn Sie hier den Wert 0 einsetzen, wird der Zwischenraum komplett entfernt. Allerdings sollten Sie dann auch einen passenden Wert für die Eigenschaft `border-collapse` verwenden, zu der wir gleich kommen.)

- Doppelte Rahmenlinien entfernen.

Wenn Sie den Zwischenraum zwischen den Zellen entfernen, rücken die Rahmenlinien zusammen und erscheinen verdoppelt. Es wird sowohl die untere Rahmenlinie einer Zelle angezeigt als auch die obere Rahmenlinie der darunter befindlichen Zelle (Abbildung 5 Mitte). Am besten lässt sich dies (und die Zellenzwischenräume gleich dazu) vermeiden, indem Sie die Eigenschaft `border-collapse` verwenden.

Zwei Werte sind möglich: `separate` und `collapse`. Die Option `separate` entspricht der Standarddarstellung mit Zellenzwischenräumen und verdoppelten Rahmenlinien. Durch die Einstellung `collapse` werden die Rahmen angrenzender Zellen dagegen zusammengefasst, und die Zwischenräume verschwinden (Abbildung 5 unten). Diese Eigenschaft wird für das `<table>`-Tag definiert, wie hier gezeigt:

```
table {
border-collapse: collapse;
}
```

Info

HTML-Tags zum Erstellen von Tabellen besitzen eine große Zahl von Attributen, mit denen Sie die gleichen Dinge erreichen können wie mit CSS. So könnten Sie auch das HTML-Attribut `border` verwenden, um die Rahmen der Tabelle und deren Zellen zu steuern. Normalerweise sollten Sie diese Attribute allerdings vermeiden. Mit CSS geht das wesentlich besser; gleichzeitig brauchen Sie deutlich weniger Code.

Zeilen und Spalten gestalten

Häufig werden Tabellenzeilen mit abwechselnden Hintergrundfarben versehen (Abbildung 6). Dadurch können Benutzer den gewünschten Inhalt leichter in der Tabelle finden. Leider bietet CSS gegenwärtig noch keine Möglichkeit, dem Browser zu sagen: „Gestalte jede zweite Zeile der Tabelle auf diese Weise.“ (Das soll sich mit CSS 3 jedoch ändern.) Stattdessen wird jeder zweiten Zeile manuell eine **CSS-Klasse** zugewiesen. z.B. `<tr class="ungerade">`, die dann wie folgt gestaltet werden kann:

```
tr.ungerade {
background-color: red;
}
```

Hierbei ist Ihre Gestaltung nicht auf Farben beschränkt. Für eine anspruchsvolle Gestaltung können Sie auch Hintergrundbilder verwenden. Ein Beispiel ist der leichte Farbverlauf für Tabellenüberschriften in Abbildung 6. Mithilfe eines Nachfahren-Selektors können Sie auch bestimmte Zellen einer Zeile gesondert formatieren. Weisen Sie den Zellen einfach eine eigene Klasse zu. z.B. `<td class="preis">`. Um der Zelle ein besonderes Aussehen zu geben, wenn sie sich in einer ungeraden Zeile befindet, können Sie den Selelctor `tr.ungerade td.preis` verwenden.

Info

Eine schnellere, JavaScript-basierte Lösung finden Sie unter Zebratables.

Die Formatierung von Spalten ist ein bisschen komplizierter. In HTML können Sie die Tags `<colgroup>` und `<col>` verwenden, um einzelne Spalten oder Spaltengruppen mit einer Klasse oder ID zu markieren. Für diese Tags funktionieren allerdings nur zwei Arten von Eigenschaften: `width` und die Hintergrundeigenschaften (`background-color`, `background-image` usw.). Aber die können wirklich praktisch sein. Wenn Sie dagegen eine einheitliche Breite für alle Tabellenspalten festlegen wollen, brauchen Sie keine HTML-Attribute und können einfach mit dem `<col>`-Tag arbeiten:

```
col#preis {
width: 200px;
}
```

Sie dürfen aber nicht vergessen, für das `<col>`-Tag eine Klasse oder ID zu definieren, wie in diesem Beispiel #preis.

Mit dem Tag `<colgroup>` können Sie mehrere Spalten zusammenfassen. Wenn Sie für dieses Tag eine Breite festlegen, wendet der Browser diesen Wert automatisch auf alle Spalten dieser Gruppe an. Eine Tabelle für Flugpläne könnte beispielsweise Spalten für die verschiedenen Termine enthalten, an denen ein Flug von Hamburg nach Stuttgart stattfindet. Mithilfe von `<colgroup>` können Sie diese Spalten zusammenfassen, ihnen eine ID geben (`<colgroup id="daten">`) und dann mit dieser Regel eine gemeinsame Breite zuweisen:

```
colgroup#daten {
width: 10em;
}
```

Obwohl die Eigenschaft `width` dem `<colgroup>`-Tag zugewiesen wird, wendet der Browser den Wert auf die einzelnen Spalten, jeweils 10 em-Einheiten an.

Um eine Spalte hervorzuheben, können Sie die verschiedenen Hintergrund-Eigenschaften verwenden:

```
col#preis {
background-color: #F33;
}
```

Sie sollten wissen, dass Hintergründe für Spalten von denen für Tabellenzellen überlagert werden können. Wenn Sie ein Hintergrundbild oder eine Hintergrundfarbe für `<td>` oder `<th>` definieren, ist der Hintergrund der Spalte möglicherweise nicht sichtbar.

Formulare mit Stil

Wenn Besucher mit einer Website interagieren, sind hierfür fast immer Formulare im Einsatz. Die

möglichen Anwendungen reichen vom Abonnement einer Mailingliste, der Suche in einer Produktdatenbank bis hin zur Aktualisierung Ihres Proftls bei MySpace, Facebook oder StudiVZ.

Hierbei gibt es keinen Grund, warum Formulare alle gleich aussehen müssen. Mit etwas CSS können Sie **Formularfelder** nahtlos in das Design Ihrer Website integrieren und dabei die gleichen Schriften, Hintergrundfarben und Abstände verwenden. Sie können fast alle behandelten Eigenschaften auch für Formulare benutzen, spezielle Formulareigenschaften gibt es nicht.

Die Ergebnisse sind jedoch stark vom verwendeten Browser abhängig (s. Abbildung 7). So waren die Möglichkeiten der Formulargestaltung bei älteren Versionen von Safari (1 und 2) noch stark eingeschränkt. Das Aussehen von Buttons, Checkboxen, Radiobuttons und Aufklappmenüs konnte nicht geändert werden. Und selbst Internet Explorer und Firefox stellen das gleiche Formularelement wahrscheinlich unterschiedlich dar. Im nächsten Abschnitt zeigen wir Ihnen, welche Eigenschaften am besten für Formularelemente geeignet sind und welche Browser diese unterstützen.

HTML-Formularelemente

Für die Erstellung von Formularen gibt es spezielle HTML-Tags. Einige (z.B. Textfelder) lassen sich erfolgreicher mit Stildeftnitionen versehen als andere (z.B. Radiobuttons). Hier einige häufig verwendete Formular-Tags und die dazu passenden Eigenschaften:

- **Fieldset**.

Mit dem `<fieldset>`-Tag können Sie zusammengehörige Formularfelder gruppieren. Die meisten Browser verstehen das auch recht gut, nur der Internet Explorer (bis zur Version 7) hat Probleme bei der Darstellung. So werden Hintergrundbilder teilweise außerhalb des Fieldset dargestellt, während die Beschriftung nicht mittig mit der Begrenzungslinie erscheint, sondern innerhalb des Fieldsets (s. Abbildung 7, linke Spalte, Mitte). Innenabstände schaffen Platz zwischen dem Inhalt und der Begrenzungslinie. (Leider ignoriert der Internet Explorer den oberen Innenabstand. Hier können Sie Abhilfe schaffen, indem Sie für das erste Element im Fieldset einen Wert für die Eigenschaft `margin-top` definieren.)

Info

Matt Heerema hat eine Lösung gefunden, die verhindert, dass der Hintergrund über die obere Begrenzungslinie des Fieldset hinausragt. Weitere Informationen finden Sie hier: Getting-fieldset-Backgrounds.

- **Beschriftung**.

Auf das `<fieldset>`-Tag folgt der **HTML-Code** für das `<legend>`-Tag. Hiermit können Sie eine Beschriftung für eine Gruppe von Formularfeldern festlegen. Diese Beschriftung erscheint, vertikal zentriert, in der oberen Begrenzungslinie eines Fieldset. Die Überschrift eines Fieldset für eine Lieferadresse könnten Sie also beispielsweise so deftnieren: `<legend>Lieferadresse</legend>`. Mit CSS können Sie die dann die Schriftart, Hintergrundfarben und – bilder sowie eigene Begrenzungslinien für das `<legend>`-Tag definieren.

- **Textfelder**.

Die Tags `<input type="text">` (<input type="text" /> in XHTML), `<input type="password">` (<input type="password" />) und `<textarea> . . . </textarea>` erzeugen verschiedene Formen von Textfeldern. Diese Tags können mit CSS am ehesten browserunabhängig gestaltet werden. Sie können für Textfelder eigene Schriftarten, -größen und -farben, aber auch andere Texteigenschaften bestimmen. Auch die Verwendung von Hintergrundfarben und Rahmen ist möglich. Neben IE, Opera und Firefox wird dies jetzt auch von Safari (ab Version 3) unterstützt. Mit der Eigenschaft `width` können Sie Textfeldern eine eigene Breite geben, allerdings wird eine Deftnition der Höhe mittels `height` nur von `<textarea>` unterstützt, weil es sich hierbei um ein Block-Element handelt.

- **Buttons**.

Formular-Buttons wie `<input type="submit" />` ermöglichen es Ihren Besuchern, Formulare abzuschicken, den Inhalt zurückzusetzen oder Aktionen zu steuern. Auch hier hat Safari die anderen Browser inzwischen eingeholt, sodass Stildefinitionen für Buttons von IE, Firefox, Opera und Safuri gleichermaßen unterstützt werden. Da Buttons Inline-Elemente sind, können Sie sie mithilfe der Eigenschaft `text-align` links, mittig oder rechts ausrichten.

- **Auswahllisten**.

Mit dem `<select>`-Tag erzeugte Auswahllisten können zumindest teilweise browserunabhängig mit CSS gestaltet werden. Während mittlerweile so gut wie alle gängigen Eigenschaften für Schriften, Hintergründe und Rahmen browserübergreifend funktionieren, tut sich Safari mit der Gestaltung der aufgeklappten Listen und Größenänderungen noch schwer und benutzt weiterhin die Systemfarben und Schriften. Rahmen werden mittlerweile unterstützt.

Info
Weitere Informationen zur Darstellung von Formularelementen mit verschiedenen Browsem finden Sie unter Styling form Controls und Styling even more form Controls.

- **Checkboxen und Radiobuttons**.

Die meisten Browser erlauben keine Formatierung dieser Elemente. Opera unterstützt allerdings eine Hintergrundfarbe, die sich interessanterweise auf die Innenseite des Elements auswirkt. Der Internet Explorer zeigt die Hintergrundfarbe um das Element herum an. Rahmen sind im IE ebenfalls möglich. Da die Darstellung in den verschiedenen Browsern äußerst unterschiedlich ausfallt, sollten Sie bei Radiobuttons und Checkboxen besser auf CSS-Stile verzichten.

Formulardesign mit CSS

Zum Erzeugen eines Formulars reicht es im Prinzip bereits aus, einfach die richtigen Tags in die Webseite einzubauen. In der Darstellung endet so etwas allerdings schnell im Chaos (s. Abbildung 8 links). Am besten sehen Formulare aus, wenn sich die Fragen und die Antwortfelder in separaten Spalten befinden (s. Abbildung 8 rechts).

Diesen Effekt können Sie auf verschiedene Weise erzielen. Der einfachste Weg besteht darin, die Einzelteile einfach in eine **HTML-Tabelle** zu packen. Auch wenn die Beschriftungen und Formularfelder keine echten tabellarischen Daten sind, lassen sie sich doch wunderbar in einer Zeilen- und Spaltenstruktur organisieren. Stellen Sie einfach die Beschriftungen („Vorname“, „Telefonnummer“ usw.) in eine Spalte und die entsprechenden Formularfelder in die andere.

Aber auch mit CSS können Sie ohne viel Stress ein zweispaltiges Formular erstellen (und damit eine Menge HTML-Code einsparen). Hier die grundsätzliche Methode:

- 1. Geben Sie jeder Beschriftung das Tag, das sie verdient.

Offensichtlich ist das `<label>`-Tag die erste Wahl für Beschriftungen, denn nichts anderes bedeutet das Wort „label“ im Deutschen. Bei Radiobuttons gibt es meistens eine Frage wie „Was ist Ihre Lieblingsfarbe?“, gefolgt von weiteren Beschriftungen für jeden Radiobutton. Welches Tag ist für die Frage am besten geeignet? Hier ist es sinnvoll, die Frage mit `<span>`-Tags zu umgeben und diesen eine eigene Klasse zuzuweisen: `<span class="dumme_frage">Was ist Ihre Lieblingsfarbe?</span>`.

- 2. Umfließen (Float) und Breite zuweisen.

Das Geheimnis dieser Technik besteht darin, einen Stil anzulegen, der die Beschriftungen als nach links verschobene Floats definiert und ihnen eine feste Breite zuweist. Hierbei sollte der Wert für `width` groß genug gewählt werden, sodass die gesamte Beschriftung auf einer Zeile stehen kann. Da wir, wie oben beschrieben. das spezielle `<label>`-Tag verwenden, können wir hier (wenn man einmal von den Radiobuttons absieht) mit einem Typ-Selektor arbeiten:

```
beschriftung {
float: left;
width: 20em;
}
```

Die Eigenschaften `width` und `float` verwandeln die Beschriftungen in gleich große Blöcke und sorgen dafür, dass sie vom folgenden Inhalt, sprich, dem Formularfeld, rechts umflossen werden können.

- 3. Den Stil anpassen.

Es fehlen nur noch ein paar Verbesserungen, um die Aufgabe fertigzustellen. Der Text der Beschriftung wird rechts ausgerichtet, damit die Beschriftung beim dazugehörigen Formularfeld angezeigt wird. Die Deklaration `clear: left`, sorgt dafür, dass die Beschriftungen untereinander und nicht nebeneinander angezeigt werden. Durch etwas Außenabstand auf der rechten Seite erzeugen wir zum Schluss noch ein wenig Leerraum, der die Beschriftungen und die Formularfelder visuell voneinander trennt. Die fertige Regel sieht nun so aus:

```
beschriftung {
float: left;
width: 20em;
text-align: right;
clear: left;
margin-right: 15px;
}
```

Damit haben Sie ein schlichtes, übersichtliches Formular. Wenn Sie wollen, können Sie weitere Verbesserungen vornehmen, z.B. die Beschriftungen fett gestalten oder in einer anderen Farbe darstellen.

Info

Ein ausführliches Beispiel für ein rein CSS-basiertes Layout für Formulare finden Sie unter Code CSS.

Die meisten Websites setzen immer noch auf Text, um ihre Nachrichten zu verbreiten. Zwar sind auch Fotos, Filmschnipsel und Flash-Animationen recht beliebt, aber am Ende ist es der lesbare Inhalt, der die Menschen wirklich interessiert. Sie hungern nach Neuigkeiten, Klatsch, Selbsthilfeanleitungen, Rezepten, FAQs, Witzen, Informationslisten und anderen schriftlichen Inhalten. Mit **CSS** können – und sollten – Sie Ihre Überschriften und den Fließtext so fesselnd gestalten wie jedes Foto.

CSS bietet Ihnen ein mächtiges Arsenal an Optionen zur **Textformatierung**, mit dem Sie Schriftarten, -farben und -größen festlegen sowie Zeilenabstände einstellen können, die die optische Wirkung Ihrer Überschriften, Listen und einfachen Absätze deutlich erhöhen können. Dieses Kapitel behandelt alle diese Dinge.

Text formatieren

Eine Möglichkeit, das Aussehen Ihrer Website aufregender zu gestalten, besteht in der Verwendung von verschiedenen Schriftarten fur Überschriften, Absätze und andere schriftliche Inhalte. Um einem HTML-Element per CSS einen bestimmten Zeichensatz zuzuordnen, verwenden Sie die Eigenschaft `font-family`, wie hier gezeigt:

```
font-family: Arial;
```

Info

Im wahren Leben reicht eine einfache Deklaration wie die im obigen Beispiel natürlich nicht aus. Wie schon beschrieben, brauchen Sie eine vollständige CSS-Regel, damit der Browser Ihre Anweisung auch versteht, z.B. `p { font-family: Arial; }`. Wenn Sie allein stehende Beispiele wie `font-family: Arial;` sehen, geschieht dies aus Gründen der Lesbarkeit. Eine Deklaration wie diese würde ohne äußere Struktur nicht funktionieren.

Die richtige Schrift auswählen

Die Auswahl der richtigen Schrift sorgt dafür, dass Ihr Text ins Auge sticht (besonders wichtig bei Überschriften) und gut lesbar ist (besonders wichtig für Fließtext). Leider können Sie im Web immer noch nicht jede gewünschte Schrift benutzen. Na gut, eigentlich geht das schon – allerdings nur, wenn der Zeichensatz auch auf dem Computer des Benutzers installiert ist, der Ihre Seite gerade betrachtet. Der handgestochene Font, den Sie in einer kleinen Schriftenboutique in einem kleinen Dorf nahe der dänischen Grenze erworben haben, wird Ihnen daher für Ihre Website wenig nutzen, sofern nicht jeder Besucher Ihrer Seite die Schrift ebenfalls gekauft und auch installiert hat. Ist das nicht der Fall, verwendet der Browser eine seiner Standardschriften – normalerweise irgendeine Variante von Times für den Haupttext und Arial oder Helvetica für Überschriften.

Info

Eine Möglichkeit, zumindest für Überschriften beliebige Fonts zu benutzen, besteht in der Verwendung der Flash-basierten sIFR-Technik (Scalable Inman Flash Replacement). Außerdem gibt es noch eine CSS-eigene Technik, die mit der CS5-Direktive `@font-face` arbeitet. über die Eigenschaft `src` können Sie die URL einer Schriftdatei angeben, die bei Bedarf geladen wird. Leider wurde `@font-face` mangels ausreichender Umsetzung in den Browserprogrammen mit der Version 2.1 wieder aus der **CSS-Spezifikation** gestrichen. Allerdings ist es gut möglich, dass diese Technik mit CSS 3 und dem WebFonts-Modul wieder zu neuem Leben erwacht.

Die häufigste Variante ist, die von Ihnen gewünschte Schrift und eine Reihe von Alternativen anzugeben. Ist Ihre erste Wahl auf dem Benutzerrechner installiert, wird diese für die Darstellung verwendet. Ansonsten geht der Browser die Liste von links nach rechts durch, bis er eine passende Schrift gefunden hat. Der Grundgedanke ist, eine Liste mehrerer ähnlicher Schriften anzugeben, die auf den meisten Betriebssystemen zu finden sind:

```
font-family: Arial, Helvetica, sans-serif;
```

In diesem Beispiel versucht der Browser zuerst, den Zeichensatz Arial zu finden und gegebenenfalls zu benutzen. Ist er nicht installiert, probiert der Browser es mit der Schrift Helvetica. Konnte auch diese nicht gefunden werden, haben Sie hoffentlich noch eine allgemeine Variante wie z.B. `sans-serif` angegeben. (Weitere Informationen zu gängigen PC- und Mac-Schriften finden Sie in Abbildung 6-2.) Bei der Angabe einer allgemeinen Schriftfamilie (wie sans-serif oder serif) wählt der Browser die seiner Meinung nach passendste verfügbare Schrift aus. Hier sehen Sie eine Liste der häufigsten Kombinationen für die Eigenschaft `font-family`, inklusive einer allgemeinen Ausweichlösung am Ende der Liste.

- Arial, Helvetica, sans-serif
- „Times New Roman“, Times, serif
- „Courier New“, Courier, monospace
- Georgia, „Times New Roman“, Times, serif
- Verdana, Arial, Helvetica, sans-serif
- Geneva, Arial, Helvetica, sans-serif
- Tahoma, „Lucida Grande“, Arial, sans-serif
- „Lucida Console“, Monaco, monospace
- „Marker Feh“, „Comic Sans MS“, fantasy
- „Century Gothic“, „Gill Sans“, Arial, sans-serif

Info

Schriftnamen, die Leerzeichen oder nicht alphanumerische Zeichen enthalten, müssen, um Missverständnisse zu vermeiden, mit Anführungszeichen umgeben werden, z.B. „Times New Roman“.

Farbiger Text

Schwarz-Weiß ist großartig für Casablanca und Filme von Woody Allen. Wenn es um Texte geht, sieht ein nettes Himmelblau oft viel interessanter und edler aus als das eintönige Standardschwarz. Mit CSS ist die Farbänderung für Ihren Text ganz einfach. Wenn Sie das Buch von vorne nach hinten durcharbeiten, ist Ihnen die Eigenschaft color sicherlich schon aufgefallen. Für die Definition der Schriftfarbe gibt es mehrere Möglichkeiten, die aber alle die gleiche Struktur haben. Der Unterschied besteht in der Art, wie die Farbe angegeben wird, zum Beispiel:

```
color: #3E8988;
```

In diesem Beispiel wurde die Farbe als hexadezimale Zahl angegeben, hier ein zartes Blaugrün (mehr zu hexadezimalen Werten in einem Moment).

In allen Grafikprogrammen von Fireworks über Photoshop bis zu GIMP ist es möglich, Farben als hexadezimalen oder als RGB-Wert auszuwählen. Auch die in Windows und Mac OS X eingebauten Farbwähler oder -paletten wandeln die von Ihnen gewählte Farbe letztendlich in einen hexadezimalen oder

RGB-Wert um.

Info

Wenn Sie Hilfe bei der Auswahl der richtigen Farbkombinationen für Ihre Website benötigen, finden Sie eine Reihe attraktiver aufeinander abgestimmter Farben. Noch einen Schritt weiter geht der automatische Farbschema-Generator.

Hexadezimale Farbwerte

Die hexadezimale Schreibweise wird von **Webdesignern** bei Weitem am häufigsten für die Beschreibung von Farben verwendet. Farbwerte wie #6600FF bestehen tatsächlich aus drei hintereinanderstehenden hexadezimalen Zahlen, hier 66, 00 und FF, die sich, wie bei den im Folgenden beschriebenen RGB-Werten, auf die Farbanteile der Farben Rot, Grün bzw. Blau beziehen, aus denen die eigentliche Farbe „gemischt“ wird.

Info

Enthält jede der drei Gruppen zweimal den gleichen Wert, können Sie die Schreibweise abkürzen. #6600FF können Sie als #60F schreiben, und #FFFFFF lässt sich auch als #FFF ausdrücken.

RGB

Neben der hexadezimalen Schreibweise können Sie die aus vielen Grafikprogrammen bekannten RGB-Werte verwenden, um eine Farbe zu definieren. Hierbei bestehen die Werte aus drei aufeinanderfolgenden Zahlen (zwischen 0 und 255) oder Prozentwerten (zwischen 0 und 100 Prozent), die für die einzelnen Farben (Rot, Grün bzw. Blau) stehen. Soll Ihre Textfarbe beispielsweise Weiß sein (vielleicht um sie besser von einem geheimnisvollen schwarzen Hintergrund unterscheiden zu können), können Sie eine der beiden folgenden Schreibweisen verwenden:

```
color: rgb(100%,100%,100%);
```

oder auch

```
color: rgb(255,255,255);
```

Info

Wenn Ihnen die Farbwerte und Hexawasauchimmers Kopfschmerzen bereiten, können Sie immer noch auf eine der klassischen benannten **HTML-Webfarben** zurückgreifen. Offiziell gibt es 17 Farben mit den Namen aqua, black. blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal. white und yellow. (Allerdings sollten Sie in diesem Fall nicht erwarten, dass Ihre Website einen Kreativitätspreis gewinnt.) Inzwischen verwenden allerdings immer mehr Anwender Monitore, die deutlich mehr Farben als die „sicheren“ 17 Webfarben beherrschen. Eine umfassende Liste möglicher benannter Farben finden Sie unter Farben definieren in HTML.

Die Schriftgröße ändern

Unterschiedliche Schriftgrößen machen Ihre Webseite interessant und leiten die Aufmerksamkeit Ihrer Besucher auf die wichtigen Bereiche. Überschriften mit einer hohen Schriftgröße erhalten mehr Aufmerksamkeit, während `Copyright`– Hinweisen in kleiner Schrift weniger Beachtung geschenkt wird.

Fuür die Definition von Schriftgrößen ist die Eigenschaft `font-size` zuständig. Die Größe kann auf unterschiedliche Weise angegeben werden, zum Beispiel:

```
font-size: 1em;
```

Der Wert und die verwendete Maßeinheit (hier: `1em`) legen die Schriftgröße fest. CSS bietet eine verwirrende Vielzahl von möglichen Werten an: Schlüsselwörter, relative Längeneinheiten wie `em` und `ex`, Prozentwerte oder absolute Angaben in Pixeln, Pica, Punkt, Zoll, Zentimeter oder Millimetern.

Die absoluten Maßeinheiten wie Pica, Punkt, Zoll usw. werden meistens in der Druckindustrie eingesetzt. Auf Webseiten funktionieren sie dagegen nur beschränkt, weil Sie nicht genau vorhersagen können, wie sie auf dem Monitor des Betrachters tatsächlich dargestellt werden. Bei der Erstellung von **Stylesheets** für den Ausdruck kann die Verwendung der Maßeinheit „Punkt“ (`pt`) dagegen sehr praktisch sein. Bei der Definition von Schriftgrößen für den Monitor sind allerdings nur wenige Maßeinheiten (Pixel, Schlüsselwörter, `em`– Einheiten und Prozentwerte) wirklich sinnvoll.

Pixel

Die verschiedenen Pixelgrößen sind am einfachsten zu verstehen, weil sie vollkommen unabhängig von den Browsereinstellungen sind. Wenn Sie beispielsweise für eine `<h1>`-Überschrift eine Schriftgröße von 36 Pixeln angeben, zeigt der Webbrowser die überschrift mit eine Höhe von 36 Pixeln an. Fertig. Manche Webdesigner schätzen Pixelwerte, weil sie unabhängig von Betriebssystem und Browser konsistente Textgrößen liefern. (Allerdings nicht alle Webdesigner.)

Wenn Sie mit der Eigenschaft `font-size` eine Schriftgröße in Pixeln definieren wollen, verwenden Sie die Abkürzung `px`:

```
font-size: 36px;
```

Info

Zwischen der Zahl und der Maßeinheit darf kein Leerzeichen stehen. `36px` ist korrekt, `36 px` nicht.

Schlüsselwörter, Prozentwerte und em-Einheiten

Drei Größenangaben, namentlich Schlüsselwörter, Prozentangaben und `em`-Einheiten, basieren auf der

Standardschriftgröße des Browsers und werden deshalb auch als relative Größenangaben bezeichnet. Die Standardgröße wird entweder erhöht oder verringert, um den definierten Wert zu erreichen. Geben Sie in Ihren **CSS-Regeln** keine Schriftgröße an, verwendet der Browser seine eigenen Einstellungen. In den meisten Browsern beträgt die Standardschriftgröße für Elemente, die keine Überschriften sind, 16 Pixel.

Websurfer können die Standardschriftgröße selbst anpassen. Im Internet Explorer gibt es hierfür den Menübefehl Ansicht – Textgröße. Hier wählen Sie Optionen wie Größer oder Kleiner, um die Schriftgföße im Browserfenster anzupassen. Ab der Version 7 können Sie, wie in Firefox, mit den Tastaturbefehlen Strg-+ und Strg– aber auch die gesamte Seite zoomen. In Firefox 3 heißt der Befehl Ansicht – Zoom – Vergrößern. Hiermit wird der Tatsache Rechnung getragen, dass Firefox inzwischen standardmäßig die gesamte Seite (also auch die Bilder!) vergrößert. Wollen Sie tatsächlich nur den Text vergrößern, können Sie das mit dem Befehl Ansicht – Zoom -. Nur Text zoomen entsprechend angeben. In Safari lautet der entsprechende Menübefehl Darstellung – Schrift größer bzw. Darstellung – Schrift kleiner. Ab Version 4 soll auch Safari das Zoomen der ganzen Seite unterstützen.

Bei der Verwendung von relativen Schriftgrößen, wie Schlüsselwörtern, Prozentwerten oder em-Einheiten, benutzt der Browser die Standardschriftgröße (unabhängig davon, ob es sich um die standardmäßigen 16 Pixel oder eine vom Benutzer gewählte Einstellung handelt) als Referenz und passt diese je nach Schlüsselwort, em-Wert oder Prozentwert an.

Schlüsselwörter

In **CSS** gibt es sieben Schlüsselwörter, mit denen Sie die Schriftgröße relativ zur Standardschriftgröße einstellen können: xx-small, x-small, small, medium, large, x-large und xx-large. Der CSS-Code sieht so aus;

```
font-size: large;
```

Hierbei entspricht das Schlüsselwort medium der Standardschriftgröße des Browsers. Die anderen Schlüsselwörter vergrößern oder verkleinern den Text jeweils um den Faktor 1,2. Beträgt die Standardgröße etwa 16 Pixel, sorgt das Schlüsselwort `large` für eine Vergrößerung auf 19 Pixel, während die Angabe small sie auf 13 Pixel verkleinert. Wurde die Standardgröße vom Benutzer verändert, wird dieser Wert bei der Angabe `large` einfach mit 1,2 multipliziert. Wenn ein Benutzer mit frisch gelaserten Augen seine neuen Fähigkeiten testen will und die Standardgröße per Ansicht – Textgröße – Kleiner verringert, wird der 13 Pixel große Text durch das CSS-Schlüsselwort large andererseits auf 16 Pixel angehoben. Das ist zwar nicht riesig, aber immer noch mehr als die Standardgröße für die betreffende Seite.

Bei der Verwendung von Schlüsselwörtern haben Sie gerade einmal sieben Wahlmöglichkeiten. Eine feinere Kontrolle über die Textgröße erhalten Sie mit den Möglichkeiten, die wir im Folgenden besprechen.

Info

Aufgrund eines Programmierfehlers im Internet Explorer 5 für Windows wird der Text bei der Verwendung von Schlüsselwörtern eine Größe kleiner dargestellt als in anderen Browsern. Das heißt small wird in IE 5 dargestellt wie `extra-small`.

Prozentwerte

Bei der Verwendung von Prozentwerten entspricht die Standardschriftgröße 100 Prozent. In den meisten Browsern entsprechen 100 Prozent daher einer Größe von 16 Pixeln.

Um eine bestimmte überschrift doppelt so groß wie die Standardschriftgröße darzustellen, setzen Sie den Wert der Eigenschaft `font-size` einfach auf 200 Prozent:

```
font-size: 200%;
```

Wenn Ihr Text dagegen ein wenig kleiner sein soll als die Standardgröße, können Sie dies mit dem Wert 90% ebenfalls problemlos erreichen.

Die obigen Beispiele sind noch recht einfach. Jetzt wird es ein wenig komplizierter. Der Wert der Eigenschaft `font-size` wird vererbt. Das heißt, die Schriftgröße von verschachtelten Tags wird möglicherweise von ihren Vorfahren-Elementen bestimmt, was bedeutet, dass bei geerbten Werten für `font-size` die Schriftgröße, die 100 Prozent entspricht, unterschiedlich ausfallen kann.

So gibt es in Abbildung 6-3 ein `<div>`-Tag, dessen Schriftgröße auf 200 Prozent festgesetzt wurde. Das entspricht der doppelten Standardschriftgröße des Browsers bzw. 32 Pixeln. Sämtliche Tags innerhalb des `<div>`-Tags erben diesen Wert und verwenden ihn als Grundlage für weitere Berechnungen. Anders gesagt: Innerhalb dieses `<div>`-Tags entspricht die Standardschriftgröße – und damit 100 Prozent – jetzt 32 Pixeln! Die primäre überschrift (`<h1>`) im `<div>`-Tag besitzt eine Schriftgröße von 100 Prozent, also ebenfalls 32 Pixeln.

em-Einheiten

Sobald Sie verstanden haben, wie Prozentwerte funktionieren, wissen Sie auch, wie em-Einheiten zu verwenden sind. Das Prinzip ist exakt das gleiche. Viele Webdesigner bevorzugen em-Einheiten aufgrund ihrer Wurzeln in der **Typografie**.

Das Wort em kommt aus der Welt der (auf echtem Papier) gedruckten Typografie. Hier steht eine em-Einheit für die ungefähre Größe des Buchstabens M einer bestimmten Schrift. Im Zusammenhang mit dem Web und CSS hat sich die Bedeutung allerdings etwas verändert. Dort bezieht sich eine em-Einheit ebenfalls auf die Standardschriftgröße. 1em bedeutet also das Gleiche wie 100 Prozent, wie im vorigen

Abschnitt beschrieben. Oder andersherum: Der Prozentwert entspricht dem em-Wert multipliziert mit 100, O.5em sind 50%. 0,75em sind 75%, 3em entsprechen 300% und so weiter.

Die folgende Deklaration funktioniert genau so wie `font-size; 200%:`

```
font-size: 2em;
```

Info

Wie bei Pixelwerten darf sich auch bei der Verwendung von em-Einheiten zwischen der Zahl und der Maßeinheit kein Leerzeichen befinden. In der englischsprachigen Welt, aus der CSS stammt, wird für Nachkommastellen kein Komma benutzt, sondern ein Punkt. Wenn Sie also „eine halbe em-Einheit“ angeben wollen, schreiben Sie O.5em usw.

Wenn es um Vererbung geht, funktionieren em-Einheiten ebenfalls wie Prozentwerte. Ein Beispiel hierfür sehen Sie in Abbildung 6-3. Für den unteren Absatz wurde die Schriftgröße mit O.75em festgelegt. Der Absatz erbt seine Standardschriftgröße vom umgebenden `<div>`-Tag, die mit 2em (entsprechend 32 Pixeln) angegeben wurde. Die absolute Schriftgröße des Absatzes berechnet sich wie folgt: O.75em multipliziert mit 32 Pixeln ergibt eine tatsächliche Größe von 24 Pixeln für den Absatz. Innerhalb des Absatzes befindet sich ein `<strong>`-Tag, dessen Größe auch mit O.75em angegeben wurde. Diese bezieht sich ebenfalls auf den vererbten Wert. Noch mehr Rechnerei: 32 Pixel (geerbt vom `<div>`-Tag) multipliziert mit O.75em (geerbt vom <p>-Tag) multipliziert mit 0.75em (geerbt vom umgebenden `<em>`-Tag) multipliziert mit O.75em (der Schriftgröße für das eigentliche `<strong>`-Tag). Das Ergebnis dieses Hirnverdrehers lautet: ungefahr 14 Pixel.

Info

Der Internet Explorer 6 und frühere Versionen haben gelegentlich Darstellungsprobleme, wenn nur em-Einheiten verwendet werden. Das lässt sich auf zwei Arten umgehen: Entweder verwenden Sie Prozentwerte, oder Sie definieren die Schriftgröße für das `<body>`-Tag als Prozentwert und verwenden dann em-Einheiten für die übrigen Angaben. Aus irgendeinem mysteriösen Grund scheint dieser Trick bestimmte Programmierfehler in Internet Explorer zu beheben.

Info

Schrift kann auf viele verschiedene Arten hervorgehoben werden. Sie können bestimmte Wörter vergrößern, Text dunkler oder heller machen und ihn visuell vom umgebenden Text trennen. Kontrast ist eines der wichtigsten Prinzipien für gutes Grafikdesign. Er kann helfen, wichtige Informationen hervorzuheben, das Auge des Lesers zu führen und das allgemeine Verständnis einer Seite zu erleichtern. Einen (englischsprachigen) überblick zum Thema typografischer Kontrast.

Wörter und Buchstaben formatieren

Auch wenn Sie viel Zeit damit verbringen werden, Farbe, Größe und Schrift für den Text Ihrer **Webseiten** anzupassen, bietet CSS noch wesentlich mehr Möglichkeiten der Textgestaltung. Hierzu gehören häufig verwendete Gestaltungsmerkmale wie fette oder kursive Schrift, aber auch einige weniger bekannte Dinge

wie Kapitälchen und änderungen des Buchstabenabstands.

Info

Auch wenn CSS es ermöglicht, mehrere Eigenschaften zur Textformatierung zu kombinieren, sollten Sie das nicht übertreiben. Zu viel Formatierung verringert die Lesbarkeit, und Ihre harte Arbeit verliert ihre Wirkung.

Kursiv- und Fettsthrift

Die meisten Webbrowser stellen Text in – und (den mittlerweile veralteten) <i>-Tags kursiv dar. Text innerhalb von – und <th>-Tags sowie Überschriften (<h1> usw.) und Text innerhalb von (den ebenfalls veralteten) <b>-Tags werden dagegen fett dargestellt. Mithilfe von CSS können Sie diese Einstellungen an die eigenen Wünsche anpassen. Sollen Überschriften nicht fettgedruckt anzeigt werden oder bestimmte Textteile kursiv erscheinen, können Sie dies mit den Eigenschaften `fant-style` und `font-weight` festlegen.

Um Text kursiv darzustellen, verwenden Sie die folgende Deklaration:

```
font-style: italic;
```

Andersherum können Sie auch dafür sorgen, dass der Text nicht kursiv angezeigt wird, indem Sie schreiben:

```
font-style: normal;
```

Info

Die Eigenschaft `fant-style` besitzt noch eine dritte Option: `oblique`, die so gut wie immer den gleichen Effekt hat wie `italic`.

Mithilfe der Eigenschaft `font-weight` können Sie das Schriftgewicht (einfach gesagt, die „Liniendicke“) Ihres Texts bestimmen. Laut der **CSS-Speziftkation** können Sie bis zu neun verschiedene „Gewichtsklassen“ (von 100 bis 900) angeben, von ultrafett (900) bis magersüchtig (100). Allerdings muss die von Ihnen gewählte Schrift auch tatsächlich alle Stufen unterstützen, damit das Ganze für die Besucher Ihrer Website überhaupt sichtbar ist. Und weil die Zahl der von Webbrowsern unterstützten Zeichensätze mit diesen Fähigkeiten äußerst gering ist, haben Sie schlicht ein Problem weniger. Im Prinzip reicht es aus, die folgenden zwei Befehle zu kennen. Um Text fett darzustellen, verwenden Sie diese Deklaration:

```
font-weight: bold;
```

Um der Schrift wieder ein normales Gewicht zu verleihen, schreiben Sie:

```
font-weight: normalj
```

Info

Da Überschriften standardmäßig fett dargestellt werden, brauchen Sie vielleicht eine andere Methode, um Textteile innerhalb einer überschrift hervorzuheben. Hier ist eine Möglichkeit:

```
h1 strong {
color: #3399FF;
}
```

Diese Regel ändert (durch den Nachfahren-Selektor) die Farbe aller `<strong>`-Tags, die sich innerhalb von `<h1>`-Tags befinden.

Großbuchstaben

Das Umwandeln von Text in Großbuchstaben ist ebenso einfach. Drücken Sie einfach die Feststelltaste und tippen Sie los – oder? Das wäre zwar eine Möglichkeit, aber CSS kann das besser. Angenommen, Sie wollen alle Überschriften einer Seite in Großbuchstaben darstellen. Dummerweise ist der Text, den Sie per Copy-and-Paste aus einem Word-Dokument übernommen haben, dort kleingeschrieben. Anstatt die Überschriften neu einzugehen, können Sie die CSS-Eigenschaft `text-transform` verwenden, wie hier gezeigt:

```
text-transform: uppercase;
```

Daneben können Sie den Text auch komplett in Kleinbuchstaben umwandeln lassen (jeweils ohne dass der eigentliche Text dabei verändert wird) oder jeweils den ersten Buchstaben eines Worts großschreiben lassen (z.B. bei TIteln und Überschriften). Die Werte hierfür lauten `lowercase;` bzw. `capitalize;`.

Da die Werte dieser Eigenschaft vererbt werden, wirkt sich die Einstellung von `text-transform` auch auf verschachtelte Tags aus. Um CSS davon abzuhalten, zu eifrig zu Werke zu gehen, verwenden Sie den Wert `none`, wie hier:

```
text-transform: none;
```

Kapitälchen

Noch mehr typografische Raffinesse erreichen Sie, wenn Sie die Eigenschaft `font-variant` einsetzen. Hiermit können Sie Ihren Text in sogenannten **Kapitälchen** anzeigen lassen. Kleinbuchstaben werden hierbei wie Großbuchstaben dargestellt, die einfach etwas kleiner sind. Nicht verstanden? Hier ein Beispiel: POMP AND CIRCUMSTANCE. Das ist auf lange Strecken zwar schwer zu lesen, für Überschriften und Beschriftungen können Kapitälchen dagegen den Eindruck einer anderen Zeit vermitteln oder auch sehr

edel wirken. Um Kapitälchen in Ihrem Text zu verwenden, schreiben Sie:

```
font-variant: small-caps;
```

Auch hier verwenden Sie zum Zurücksetzen den Wert `normal`.

Ausschmückungen

Eine weitere CSS-Eigenschaft, die in diesem Zusammenhang nicht fehlen darf, ist – `text-decoration`. Hiermit können Sie den Text mit verschiedenen Unter-, über- und sogar Durchstreichungen versehen (s. Abbildung 6-4). Wenn Sie sich so richtig blamieren wollen, können Sie Ihren Text sogar blinken lassen wie die LEDs an Ihrem DSL-Router. Für die Eigenschaft `text-decoration` können Sie aus den Schlüsselwörtern `underline` (unterstreichen), `overline` (überstreichen; Linie über dem Text), `line-through` (durchstreichen) und `blink` wählen. Um Text zu unterstreichen, schreiben Sie beispielsweise:

```
text-decoration: underline;
```

Um die Effekte zu kombinieren, können Sie auch mehrere Schlüsselwörter gemeinsam angeben. Mit der folgenden Deklaration wird über und unter dem Text eine Linie angezeigt:

```
text-decoration: underline overline;
```

Allerdings sollten Sie diese Effekte nur einsetzen, wenn sie wirklich sinnvoll sind. Jeder, der lange genug im Web unterwegs war, hält unterstrichenen Text intuitiv für einen Link und versucht, ihn anzuklicken. Wörter, die nicht zu einem Link gehören, sollte man deshalb nur mit Bedacht unterstreichen. Der Wert `blink` hat übrigens nur die eine Funktion: Sie als Anfänger bloßzustellen!

Info

Unter- und überstreichungen können Sie auch mithilfe der Eigenschaft `border` erreichen. Der große Vorteil liegt darin, dass Sie die Platzierung, Größe und Farbe bestimmen können. So sind über- und Unterstreichungen möglich, die Ihren Text nicht wie einen link aussehen lassen.

Der Wert `overline` sorgt dafür, dass eine Linie oberhalb des Texts erzeugt wird, während `line-through` den Text tatsächlich durchstreicht. Manche Webdesigner verwenden diesen Effekt, um anzuzeigen, dass bestimmter Text auf einer Seite entfernt wurde oder nicht mehr aktuell ist. Wenn Sie dies mit dem Selektor `a:visited` kombinieren, können Sie einen coolen Effekt erzeugen, bei dem bereits besuchte Links durchgestrichen werden, wie die Artikel auf einem Einkaufszettel.

Um sämtliche Textdekorationen zu deaktivieren, gibt es außerdem noch das Schlüsselwort none, das verwendet wird wie hier gezeigt:

```
text-decoration: none;
```

Warum sollten Sie eine Textausschmückung verwenden, die die Dekorationen entfernt? Am häufigsten kommt das vor, wenn die Unterstreichung unter einem Link entfernt werden soll.

Buchstaben- und Wortabstände

Eine weitere Möglichkeit, Text hervorzuheben, besteht in der änderung der Abstände zwischen Buchstaben und Wörtern. Mithilfe der CSS-Eigenschaft `letter-spacing` können Sie Überschriften noch mehr Gewicht verleihen, indem Sie mehr Buchstaben in einer Zeile unterbringen. Erhöhen Sie den Abstand dagegen, erscheinen Überschriften ruhiger und majestätischer. Negative Werte verringern den Buchstabenabstand:

```
letter-spacing: -1px;
```

Ein positiver Wert schafft mehr Platz zwischen den Buchstaben:

```
letter-spacing: 10px;
```

Analog dazu können Sie auch die Abstände zwischen ganzen Wortern vergrößern oder verringern. Hierfür ist die Eigenschaft `word-spacing;` zuständig. Sie wirkt sich nur auf den Leerraum zwischen den Wörtern aus, ohne dabei die Wörter selbst zu verändern:

```
word-spacing: 2px;
```

Für diese beiden Eigenschaften können Sie die gleichen Maßeinheiten verwenden wie beispielsweise für die Schriftgröße: Pixel, em-Einheiten. Prozentwerte usw. – selbst negative Werte sind erlaubt. Sofern Sie nicht irgendeinen völlig verdrehten Designeffekt erzielen wollen, z.B. vollkommen unleserlichen Text, sollten diese Werte aber nicht zu groß ausfallen. Bei zu hohen negativen Werten kommt es bei Wörtern und Buchstaben zu überschneidungen. Damit der Inhalt Ihrer Site klar und verständlich bleibt, sollten Sie diese Eigenschaften daher mit Vorsicht anwenden.

Ganze Absätze formatieren

Einige CSS-Eigenschaften wirken nicht auf einzelne Wörter, sondern auf komplette Textblöcke. Mit den Eigenschaften in diesem Abschnitt können Sie komplette Absätze, Überschriften usw. formatieren.

Zeilenabstand anpassen

Neben den Abständen zwischen Wörtern und Buchstaben ermöglicht CSS mit der Eigenschaft `line-height` auch eine Anpassung des Leerraums zwischen den Textzeilen. Je höher der Wert, desto größer wird der Abstand zwischen zwei Zeilen (s. Abbildung 6-6).

Zeilenabstand in Pixeln, em-Einheiten oder Prozentwerten

Wie bei der Eigenschaft `font-size`, so können Sie auch mit `line-height` die in CSS gültigen Maßeinheiten verwenden. Sinnvoll sind aber auch hier in erster Linie Pixelwerte, em-Einheiten und Prozentwerte. z.B.:

```
line-height: 150%;
```

Im Allgemeinen sind Prozentwerte oder em-Einheiten besser geeignet als Pixel, weil sie sich automatisch an den Wert der Eigenschaft `font-size` anpassen. Wenn Sie die Zeilenhöhe mit 10 Pixeln angeben und die Schriftgröße später erhöhen (z.B. auf 36 Pixel), werden sich die Zeilen überschneiden, weil die Zeilenhöhe gleich bleibt. Verwenden Sie dagegen einen Prozentwert (beispielsweise 150%), ändert sich die Zeilenhöhe proportional zum Wert von `font-size`.

Die Standard-Zeilenhöhe beträgt in den meisten Browsern 120 Prozent. Soll der Zeilenabstand kleiner sein, muss der Wert also unter 120 Prozent liegen; um den Abstand zu erhöhen, muss der Wert größer als 120 Prozent sein.

Info

Um zu ermitteln, wie viel Leerraum zwischen den einzelnen Textzeilen vorhanden sein soll, subtrahiert der Browser die Schriftgröße vom Wert der Zeilenhöhe. Das Ergebnis ist der sogenannte Durchschuss – der leerraum zwischen den Zeilen eines Absatzes. Beträgt die Schriftgröße beispielsweise 12 Pixel und die Zeilenhöhe 18 Pixel (ISO Prozent), betlägt der Durchschuss 18-12 Pixel, also 6 Pixel.

Numerischer Zeilenabstand

Als Wert für `line-height` können Sie in CSS auch eine einfache Zahl verwenden (Nachkommastellen sind erlaubt, negative Werte nicht)

```
line-height: 1.5;
```

Für diesen Wert gibt es keine Maßeinheit (wie `em` oder `px`). Der Browser multipliziert diese Zahl mit dem Wert für `font-size`, um die Zeilenhöhe zu ermitteln. Hat der Text also die Höhe 1em und die Zeilenhöhe den Wert 1.5 (Achtung: Punkt statt Komma verwenden!), dann ist die berechnete Zeilenhöhe 1.Sem. In den meisten Fällen können Sie als Wert auch 1.5em oder 150% schreiben. Manchmal kann dieser Multiplikationsfaktor aber auch ganz praktisch sein, z.B. wenn verschachtelte Tags den Wert für `line-height` von ihren Vorfahren geerbt haben.

Legen Sie beispielsweise den Wert von `line-height` für das <body>-Tag auf ISO Prozent fest, wird dieser an alle enthaltenen Tags vererbt. Allerdings wird nicht der Prozentwert selbst vererbt, sondern die berechnete Zeilenhöhe. Beträgt die Schriftgröße 10 Pixel, entsprechen 150% einer Zeilenhöhe von 15 Pixeln. Die Tags erben also eine Zeilenhöhe von 15 Pixeln und nicht den Prozentwert. Wenn Sie jetzt einen Absatz haben, in dem der Text 36 Pixel groß ist, reicht die Zeilenhöhe von 15 Pixeln nicht mehr aus. Die Zeilen würden übereinandergeschoben und wären kaum noch lesbar.

Anstelle des Prozentwerts von 150 Prozent für das <body>-Tag könnten Sie allen Elementen die gleiche relative Zeilenhöhe geben, indem Sie die Regel

```
body {
line-height: 1.5;
}
```

verwendeten. So wird nicht der berechnete Pixelwert, sondern der zur jeweiligen Textgröße proportionale Wert weitergegeben. Ein Absatz mit einer Textgröße von 36 Pixeln hätte demnach eine Zeilenhöhevon 1,5 x 36 bzw. 54 Pixeln.

Texlausrichtung

Eine der schnellsten Methoden, das Aussehen einer Webseite zu ändern, besteht in der Anpassung der **Textausrichtung**. Mit der Eigenschaft `text-align` können Sie den Text eines Absatzes zentrieren, ihn linksbündig oder rechtsbündig ausrichten oder auch im Blocksatz darstellen. Standardmäßig wird der Text einer Seite linksbündig angezeigt. Vielleicht sollen aber die Überschriften zentriert werden, um ein formaleres Aussehen zu erreichen. Sprachen, die von rechts nach links gelesen werden (z.B. Hebräisch oder Arabisch), werden normalerweise rechtsbündig ausgerichtet. Die Eigenschaft `text-align` übernimmt eines der Schlüsselwörter `left`, `right`, `justify` oder `center;`

```
text-align: center;
```

Text im Blocksatz sieht vor allem aufgedruckten Seiten gut aus, weil die feine Auflösung besonders kleine Anpassungen in den Buchstaben- und Wortabständen ermöglicht. Hiermit werden große, unansehnliche Zwischenräume vermieden. Bei Webseiten sind aufgrund der verhältnismäßig geringen Monitorauflösung

diese feinen Anpassungen nicht möglich, und die Abstände wirken grober. Wenn Sie den Wert `justify` verwenden, können die Zwischenräume von Zeile zu Zeile sehr unterschiedlich ausfallen, wodurch die Leserlichkeit möglicherweise leidet. Sie sollten die Auswirkungen daher ausgiebig mit verschiedenen Browsern testen, damit Ihr Text überall attraktiv und lesbar aussieht.

Zeilen einrücken und unerwünschte Ränder entfernen

In vielen Büchern wird die erste Zeile eines Absatzes eingerückt. Hiermit wird der Beginn des Absatzes markiert, wenn es zwischen den Absätzen keinen Leerraum gibt. Im Web ist es normalerweise umgekehrt: Hier gibt es einen gewissen Abstand zwischen den Absätzen, aber keine Einrückungen.

Soll sich das Aussehen Ihrer Website von dem anderer Seiten unterscheiden und mehr nach einem feinen, gedruckten Buch aussehen, können Sie sich die **CSS-Eigenschaften** `text-indent` und `margin` zunutze machen. Hiermit können Sie nur die erste Zeile Einrückungen definieren und den vertikalen Leerraum zwischen den Absätzen verringern (oder vergrößern).

Einrückungen

Um die erste Zeile eines Elements (meistens eines Absatzes) einzurücken, verwenden Sie die Eigenschaft `text-indent` mit einer absoluten oder relativen Längenangabe:

```
text-indent: 2Spx;
```

oder auch:

```
text-indent: 5em;
```

Hierbei entspricht die Angabe in Pixeln der tatsächlichen Anzahl von Bildschirmpunkten, während die em-Einheiten in etwa der Anzahl der Buchstaben entspricht, um die der Text (basierend auf der aktuellen Schriftgröße) eingerückt werden soll.

Info

Mithilfe von negativen Werten können Sie sogenannte hängende Einzüge erzeugen. Hierbei wird die erste Zeile nicht ein-, sondern ausgerückt und steht demnach weiter links als die folgenden Zeilen. (Die Zeile „hängt“ über die linke Kante.)

Die Verwendung von Prozentwerten funktioniert hier anders als bei den Eigenschaften, die Sie schon kennen. In diesem Fall beziehen sich Prozentwerte nicht auf die Schriftgröße, sondern auf die Breite des Elements. das den Absatz enthält. Wurde für `text-indent` ein Wert von 50 Prozent festgelegt und der Absatz erstreckt sich über die gesamte Breite des Browserfensters, beansprucht der Absatz nur noch die Hälfte des möglichen Bereichs.

Den Leerraum zwischen den Absätzen anpassen

Die meisten Designer hassen den Leerraum, den viele Browser standardmäßig zwischen einzelnen Absätzen erzeugen. Vor CSS konnte man kaum etwas dagegen tun. Inzwischen gibt es die CSS-Eigenschaften `margin-top` und `margin-bottom`, mit denen Sie den Abstand selbst verringern (oder auch vergrößern) können. Um sämtlichen Leerraum ober- und unterhalb eines beliebigen HTML-Elements zu entfernen, schreiben Sie:

```
margin-top: 0;
margin-bottom: 0;
```

Zum Entfernen der Zwischenräume zwischen allen Absätzen, können Sie eine Regel wie diese verwenden:

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

Wie bei der Eigenschaft `text-indent` können Sie die üblichen **CSS-Maßeinheiten** verwenden, und die Prozentwerte beziehen sich auch hier auf die Breite des umgebenden Elements. Da es ziemlich verwirrend sein kann. den Raum ober- und unterhalb eines Absatzes anhand seiner Breite zu berechnen, ist es sinnvoll, bei em-Einheiten oder Pixelwerten zu bleiben.

Info

Da viele Browser eigene Werte für die Abstände zwischen Absätzen und Überschriften verwenden, ist es häufig üblich, die Einstellungen am Anfang eines Stylesheets zu neutralisieren (auf null zu setzen).

Einen speziellen Effekt erreichen Sie, wenn Sie `margin-top` oder `margin-bottom` einen negativen Wert zuweisen. So wird ein Absatz bei einem Wert von `-10 Pixeln margin-top` um 10 Pixel nach oben verschoben. Es kann sogar passieren, dass es zu überschneidungen mit dem darüberliegenden Element kommt. (Ein Beispiel hierfür fmden Sie in Abbildung 6-1.)

Den ersten Buchstaben oder die erste Zeile eines Absatzes formatieren

Soll nur ein Teil eines Absatzes formatiert werden, helfen die Pseudoelemente `:first- letter` und `:first-line`. Dies sind keine Eigenschaften, sondern spezielle Erweiterungen für Selelktoren, mit denen festgelegt wird, auf welchen Teil eines Elements (z.B. eines Absatzes) eine bestimmte CSS-Eigenschaft angewandt werden soll. Mit dem Pseudoelement `:first-letter` ist es beispielsweise möglich, Versalien (Großbuchstaben, die größer sind als der übrige Text) am Anfang eines Absatzes zu verwenden und Ihrer Seite so das Aussehen eines handgeschriebenen Dokuments zu geben. Um den ersten Buchstaben eines Absatzes rot und fett darzustellen, können Sie beispielsweise folgende Regeln verwenden:

```
p:first-letter {
font-weight: bald;
color: red;
}
```

Weniger pauschal ist die Verwendung einer Klasse. Soll beispielsweise nur der erste Buchstabe eines Absatzes formatiert werden, dem die Klasse `.einleitung` zugewiesen wurde, können Sie schreiben: `p.einleitung:first-letter`.

Auf ähnliche Weise wie `:first-letter` formatiert das Pseudoelement `:first-line` die gesamte erste Zeile eines Elements (s. Abbildung 6-7). Auch `:first-line` können Sie auf beliebige Textpassagen anwenden, z.B. auf Überschriften (`h2:first-line`) oder Absätze (`p:first-line`). Auch hier können Sie die Wirkung einschränken, indem Sie eine Klasse verwenden. Um zum Beispiel die gesamte erste Zeile des ersten Absatzes einer Seite in Großbuchstaben zu verwandeln, müssen Sie zuerst dem entsprechenden Absatz die richtige Klasse zuweisen: `<p class="einleitung">`. Dann erstellen Sie einen CSS-Regel, die die komplette erste Zeile dieses Absatzes in Großbuchstaben umwandelt:

```
p.einleitung:first-line {
text-transform: uppercase;
}
```

Stildefinitionen für Listen

Die Tags `<ul>` und `<ol>` erzeugen ungeordnete bzw. nummerierte Listen. Hiermit werden beispielsweise Links gruppiert oder mehrere Arbeitsschritte in der Darstellung zusammengefasst. Auch bei den Listen sind Sie nicht an die Standarddarstellung des Webbrowsers gebunden. So können Sie beispielsweise eigene Aufzählungszeichen (Bullets) verwenden oder anstelle von arabischen Zahlen (1, 2, 3…) römische Ziffern (I, II, III, IV …) oder Buchstaben einbinden.

Verschiedene Aufzählungszeichen oder: die Quadratur des Kreises

Die meisten Webbrowser stellen Listeneinträge von ungeordneten Listen (`<ul>`) mit einem vorangestellten schwarzen Kreis dar. Für die Einträge von geordneten Listen (`<ol>`) wird dagegen eine Zahl verwendet. Für ungeordnete Listen können Sie mit CSS aus drei verschiedenen Listenmarkern wählen: `disc` (der standardmäßige schwarze Kreis), `circle` (ein schwarzer Kreisumriss) und `square` (ein schwarzes Quadrat). Für geordnete (nummerierte) Listen können Sie sogar aus sechs verschiedenen Variationen wählen: `decimal`, `decimal-leading-zero`, `upper-alpha`, `lower-alpha`, `upper-roman` und `lower-roman` (s. Abbildung 6-8). Sie vergeben diese Optionen mit der Eigenschaft `list-style-type:`

```
list-style-type: square;
```

oder auch

```
list-style-type: upper-alpha;
```

Info

Wenn Sie zeigen wollen, wie sprachgewandt Sie sind, können Sie anstelle der Zahlen auch griechische Buchstaben verwenden. Hierfür verwenden Sie die Option `lower-greek`. Diese ist zwar nicht mehr Teil der CSS 2.1-Spezifikation, wird aber von vielen Browsern noch unterstützt.

Meistens werden Sie diese Eigenschaft per Typ-Selektor auf ein `<ul>`– oder `<ol>`-Tag anwenden, z,B, als `ul { list-style-type: square; }`, oder auch in Form einer Klasse, die einem dieser Tags zugewiesen wird. Sie können die Eigenschaft aber auch einzelnen Listeneinträgen (`<li>`) zuweisen und so mehrere Formate in der gleichen Liste verwenden. Vielleicht verwenden Sie als Standardformat für Listen quadratische Aufzählungszeichen (`square`) und definieren für einzelne Einträge mithilfe einer Klasse den Wert `circle:`

```
.kreis {
list-style-type: circle;
}
```

Jetzt können Sie die Klasse jedem zweiten Listenelement zuweisen. wodurch ein wechselndes Muster auch runden und quadratischen Aufzählungszeichen entsteht:

- Eintrag 1
- Eintrag 2
- Eintrag 3
- Eintrag 4

Gelegentlich wollen Sie die Aufzählungszeichen aber auch komplett verstecken oder stattdessen Ihre eigenen Grafiken zeigen. Bei der Definition einer Navigationsleiste aus mehreren Links wird oft eine `<ul>`-Liste verwendet, wobei auch hier die Listenmarkierungen eher unerwünscht sind. Um die Aufzählungszeichen zu deaktivieren, verwenden Sie das Schlüsselwort `none:`

```
list-style-type: none;
```

Aufzählungszeichen positionieren

Standardmäßig werden die Aufzählungszeichen links vom eigentlichen Text der Listenelemente dargestellt (s. Abbildung 6-9). Mit der CSS-Eigenschaft `list-style-position` können Sie die Platzierung

(zumindest teilweise) steuern. Hierbei können Sie wählen, ob das Aufzählungszeichen außerhalb oder innerhalb des Textblocks (s. Abbildung 6-9) dargestellt werden soll. Für die Standardeinstellung außerhalb des Texts schreiben Sie:

```
list-style-position: outside;
```

oder aber:

```
list-style-position: inside;
```

Info

Mit der Eigenschaft `padding-left` (s. Seite 139) können Sie den Abstand zwischen Aufzählungszeichen und listeneintrag steuern. Um diese Methode nutzen zu können, benötigen Sie eine CSS-Regel für die einzelnen listenelemente (`<li>`). Allerdings funktioniert diese Methode nur, wenn `list-style-position` den Wert `outside` hat (oder Sie einfach gar keinen Wert definieren).

Wenn Ihnen die standardmäßige linke Einrückung von Listen nicht gefallt, können Sie den Leerraum mit den beiden Eigenschaften `margin-left` und `padding-left` für die Liste auf den Wert 0 setzen. Mit folgender Regel können Sie die Einrückung für alle ungeordneten und geordneten Listen entfernen:

```
ul, ol {
padding-left: 0;
margin-left: 0;
}
```

Oder Sie erstellen eine **CSS-Klasse** mit diesen Eigenschaften und wenden sie nur auf bestimmte `<ul>`– und `<ol>`-Tags an. Einige Browser verwenden die CSS-Eigenschaft `padding-left`, um die Einrückung umzusetzen (Firefox, Mozilla, Safari), andere dagegen `margin-left` (Internet Explorer). Daher werden hier beide Eigenschaften auf null gesetzt.

Normalerweise stellen Browser die einzelnen Listenelemente direkt untereinander dar. Mithilfe der Eigenschaften `margin-top` und `margin-bottom` können Sie die Abstände für einzelne oder auch alle Listenelemente an die eigenen Bedürfnisse anpassen. Die Funktionsweise ist für Listen wie für Absätze gleich. Sie müssen nur dafür sorgen, dass die Regel für einzelne Listenelemente `<li>`-Tags) angewendet wird. Sie können entweder eine entsprechende Klasse definieren, die Sie den einzelnen `<li>`-Tags zuweisen, oder, und das geht einfacher, Sie wählen die `<li>`-Tags mit einem Typ-Selektor aus. Sie sollten eine solche Regel nicht auf `<ul>`– oder `<ol>`-Tags anwenden. Hier würden Sie nur den Zwischenraum zwischen der Liste und den umgebenden Elementen (Absätzen, Überschriften usw.) verändern, nicht aber

zwischen den einzelnen Listeneinträgen.

Grafische Aufzählungszeichen

Wenn Ihnen die Kreise und Quadrate als Aufzählungszeichen nicht gefallen, können Sie stattdessen auch eigene Grafiken verwenden. Diese lassen sich mit Bildbearbeitungsprogrammen wie Photoshop, Fireworks oder GIMP schnell erstellen. Clipart-Sammlungen und die meisten Symbolschriften (z.B. Webdings) können als Inspirationsquelle dienen.

Info

Im Web gibt es viele Beispiele für Aufzählungszeichen. Eine liste mit über 200 kostenlosen Grafiken finden Sie beispielsweise unter stylegala

Mit der Eigenschaft `list-style-image` können Sie eine URL zur Grafik Ihrer Wahl angeben. Dies funktioniert auf die gleiche Weise wie die Definition eines Bilds mit dem `src`-Attribut für das `<img>`-Tag. Sie verwenden `list-style-image` wie folgt:

```
list-style-image: url(bilder/mein_aufzaehlungszeichen.gif);
```

Die Angabe von `url` und die runden Klammern sind hier notwendig. Der Teil innerhalb der Klammern (bilder/mein_aufzaehlungszeichen.gif) entspricht der URL zur gewünschten Grafik. Im Gegensatz zu **HTML** wird die URL hier nicht mit Anführungszeichen umgeben, obwohl die meisten Browser nichts dagegen haben.

Info

Wenn Sie in einem externen **Stylesheet** eine relative URL zur gewünschten Grafik angeben, ist diese Angabe `relativ` zum Speicherort des Stylesheets und nicht zur HTML-Seite, die das Stylesheet einbindet.

Zwar können Sie mit der Eigenschaft `list-style-image` ein eigenes Aufzählungszeichen definieren, dessen Platzierung lässt sich aber nicht steuern. Möglicherweise wird es zu weit ober- oder unterhalb der Zeile angezeigt, und Sie müssen die Grafik entsprechend anpassen, bis alles stimmt. Gibt es einen besseren Weg, der die Eigenschaft `background-image` benutzt. Hiermit können Sie die Platzierung Ihrer Aufzählungszeichen sehr genau steuern.

Info

Wie bei der Eigenschaft `font` gibt es auch eine Kurzschrift-Eigenschaft für die formatierung von Listen. Mit der Eigenschaft `list-style` können Sie die Werte für die Listen-Eigenschaften `list-style-imoge`, `list-style-position` und `list-style-type` in einer Deklaration zusammenfassen. Oie Regel `ul { list-style: circle inside; }` formatiert ungeordnete Listen, indem ein Kreisumriss als Aufzählungszeichen verwendet wird, das als Teil des jeweiligen Eintrags dargestellt wird. Wenn Sie sowohl für `list-style` als auch für `list-style-imoge` einen Wert angeben (z.B. `list-style: circle`

`url(bilder/bullet.gif) inside;`), verwendet der Webbrowser das angegebene Aufzählungszeichen (in diesem Fall den schwarzen Kreis), sofern die Grafik nicht geladen werden konnte.