jetzt lerne ich

Start ohne Vorwissen

# CSS

Standardkonformes Webdesign mit Cascading Style Sheets

FLORENCE MAURICE PATRICIA REX

Markt+Technik

# Kapitel 3

jetzt lerne ich

# Schrift- und Absatzformatierung

Das letzte Stündlein des `font`-Elements hat endgültig geschlagen. Mit CSS können Sie Schriftart, Schriftfarbe und Schriftgröße für das ganze Dokument, für bestimmte Elemente oder einzelne Bereiche festlegen. Sie können jedoch auch die Zeilenhöhe bestimmen und einzelnen Abschnitten eine Hintergrundfarbe zuweisen. Absatzeinrückungen lassen sich ebenfalls komfortabel vornehmen, ohne dass Sie zu Tricks wie mehreren geschützten Leerzeichen (` `) greifen müssen. Und positiv anzumerken ist außerdem: Die Unterstützung der Browser für die Schrift- und Absatzformatierungen ist durchwegs so gut, dass die meisten Methoden problemlos eingesetzt werden können.

## 3.1 Schriftart

Natürlich konnten Sie die Schriftfarbe für das ganze Dokument auch in HTML festlegen, dazu diente das Attribut `text` im `body`-Element. Um hingegen die Schriftart/Schriftfamilie zu bestimmen, gab es in HTML nichts anderes als das `font`-Element. Dieses musste dabei für jedes Element bzw. jeden Bereich einzeln definiert werden – höchst unpraktisch, wenn man sich entscheidet, dass die Überschriften doch anders aussehen sollen. Zudem werden die Dokumente durch die vermehrte Verwendung von `font`-Elementen auch unnötig aufgebläht, was zu mehr Traffic führt.

In CSS dient `font-family` zur Bestimmung der Schriftart. Soll diese für das ganze Dokument gelten, so legen Sie diese Eigenschaft am besten für `body` fest. Hinter `font-family` können Sie die Schriftart angeben. Es empfiehlt sich, eine Schriftliste zu schreiben. Dann wird – sofern auf dem Computer vorhanden – die erste Schrift der Liste gewählt, ansonsten die zweite. Ist auch diese nicht installiert, wird die nächste genommen usw. So könnte eine Schriftdefinition für `body` aussehen:

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

Arial und Helvetica sind zwei bekannte Schriften. Aber wahrscheinlich haben Sie die Schriftart »sans-serif« noch nicht gesehen. Dies ist auch keine Schriftart im engeren Sinne, sondern eine generische Schriftangabe oder Schriftfamilie.

Die generischen Schriftangaben bezeichnen nur eine Art Schrift und nicht eine konkrete Realisierung und sorgen dafür, dass der Browser eine entsprechende Schrift wählt. Eine solche generische Schriftangabe sollten Sie zum Schluss angeben – damit stellen Sie sicher, dass eine Schrift verwendet wird, die ungefähr der gewünschten entspricht. Ansonsten würde nämlich die Standardschrift des Browsers benutzt, wenn die davor angegebenen Schriften nicht vorhanden sind.

In CSS stehen folgende generische Schrifttypen zur Verfügung:

*Tabelle 3.1: Generische Schriftarten in CSS*

| Name der generischen Schriftfamilie | Steht für |
|---|---|
| `serif` | Schrift mit Serifen. Serifen sind kleine Verzierungen, die einen Buchstabenstrich am Ende abschließen. Eine typische Schrift mit Serifen ist Times oder Times New Roman. |
| `sans-serif` | Serifenlose Schrift. Typische Vertreter sind Arial, Verdana oder Helvetica. |
| `cursive` | Verbundene Buchstaben, die an eine Handschrift erinnern, typischer Vertreter ist Zapf-Chancery. |
| `fantasy` | Fantasie-Font mit dekorativen Elementen, z.B. Cotton Wood. |
| `monospace` | Diktengleiche Schrift: Alle Buchstaben nehmen gleich viel Platz ein. Erinnert an Schreibmaschinenschrift, Beispiele hierfür sind Courier oder Courier New. |

Zur Demonstration der verschiedenen generischen Schriftarten lässt sich das folgende Listing verwenden:

*Listing 3.1: Generische Schriftarten*

```
<p style="font-family: sans-serif">sans-serif</p>
<p style="font-family: serif">serif</p>
<p style="font-family: cursive">cursive</p>
<p style="font-family: fantasy">fantasy</p>
<p style="font-family: monospace">monospace</p>
```

Abbildung 3.1 zeigt die Darstellung in den Browsern Firefox und Opera.

Besonders bei der Fantasy-Schrift sieht man deutlich, dass ganz unterschiedliche Schriften benutzt werden. Abgesehen davon ist in der Abbildung klar erkennbar, dass bei derselben Schriftgröße die Schriften unterschiedlich groß wirken.

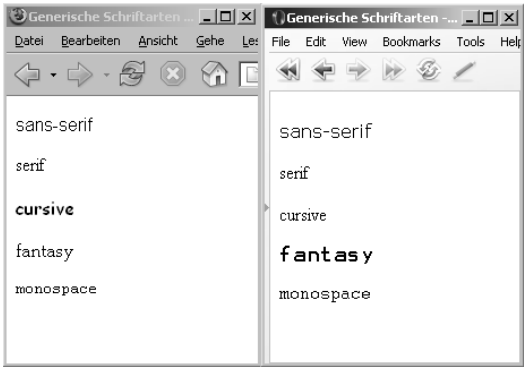



*Abb. 3.1: Die verschiedenen Schriftarten und wie sie in Firefox (links) und in Opera (rechts) dargestellt werden*

Wenn Sie eine Schrift verwenden möchten, deren Name aus mehr als einem Wort besteht, sollten Sie den Schriftnamen in Anführungszeichen setzen:

```
.qc { font-family: "Courier New", Courier, monospace; }
```

Beim Einsatz von Inline-Style-Angaben dürfen nicht dieselben Anführungszeichen verwendet werden wie zur Begrenzung der Attributwerte. Benutzen Sie am besten einfache Anführungszeichen um den Schriftnamen und doppelte Anführungszeichen um den gesamten Attributwert.

Das folgende Beispiel ist korrekt:

```
<p style="font-family: 'Courier New', Courier, monospace;">Monospace</p>
<!-- funktioniert -->
```

Hingegen wird das folgende Beispiel Probleme machen:

```
<p style="font-family: "Courier New", Courier, monospace;">Monospace</p>
<!-- funktioniert nicht -->
```

In CSS 2 ist eine weitere Technik für Schriften vorgesehen: die herunterladbaren Schriften. Damit ist es möglich, auch ausgefallenere Schriften einzusetzen, die nicht auf dem Rechner des Benutzers installiert sind. Da jedoch diese Methode nicht plattform- und browserübergreifend funktioniert, wird sie hier nur kurz vorgestellt.

Eingeleitet wird die Schriftartdefinition über `@font-face`. Bei `font-family` geben Sie einen Namen an, unter dem Sie die Schrift dann nachfolgend ansprechen können. Als Nächstes folgt `src`, dahinter steht hinter `url` der Pfad zur Schriftdatei.

```
@font-face {
   font-family: Fantasieschrift;
   src: url(http://www.domain.de/fonts/fantasie.eot);
}
```

Dann könnte man im weiteren Stylesheet ganz normal die Schriftart verwenden:

```
p { font-family: Fantasieschrift; }
```

Um entsprechende herunterladbare Zeichensätze zu erstellen, benötigen Sie für den Internet Explorer das Web Embedding Fonts Tool (WEFT), das die Schriftarten in komprimierte Definitionsdateien vom Typ `.eot` konvertiert. Das Tool und weitere

Informationen dazu finden Sie bei Microsoft unter *http://www.microsoft.com/typography/web/embedding/weft3/*.

Leider funktioniert das oben geschilderte Verfahren nur beim Internet Explorer unter Windows, und außerdem benötigt das Herunterladen der Schriftdatei auch eine gewisse Zeit, was die Darstellung der Seite verzögert. Zusätzlich müssen Sie natürlich eine entsprechende Lizenz besitzen, die das Herunterladen der Schrift gestattet. In CSS 2.1 sind diese `@font-face`-Deklarationen nicht mehr vorgesehen.

Bevor wir zum nächsten wichtigen Thema bei der Zeichenformatierung, der Schriftgröße kommen, folgt erst einmal eine Vorstellung der möglichen Größeneinheiten in CSS.

## 3.2 Längenangaben/Größeneinheiten

In CSS stehen mehrere Maße für relative Angaben und verschiedene Maße für absolute Angaben bereit. Diese können Sie einerseits bei Schriften verwenden, andererseits auch für Abstände zwischen verschiedenen Elementen oder auch bei der Platzierung von Bereichen – die letzten beiden Themen werden etwas später behandelt.

Tabelle 3.2 stellt die Längenangaben vor.

*Tabelle 3.2: Mögliche Größeneinheiten in CSS*

| Abkürzung | Bedeutung |
|---|---|
| cm | Zentimeter |
| mm | Millimeter |
| in | Zoll/Inch. Ein Zoll entspricht 2,54 Zentimetern. |
| pt | Punkt. Ein Punkt ist 1/72 Zoll, d.h. ca. 0,35 mm. |
| pc | Pica. Ein Pica entspricht 12 Punkt, ergibt also 0,42 cm. |
| em | Entspricht der Größe der aktuell gewählten Schriftgröße. |
| ex | Die x-Höhe des entsprechenden Fonts. Ist meist ungefähr so groß wie die Höhe des x und ca. 1/2 em. |
| px | Pixel sind relativ zur Auflösung des Ausgabegeräts – in den meisten Fällen des Bildschirms. |
| % | Prozentangaben beziehen sich auf das Elternelement. |

Die Längeneinheiten lassen sich in zwei Gruppen teilen: in die absoluten und in die relativen Längeneinheiten. Zentimeter, Millimeter, Inch, Punkt, Pica sind absolute Längeneinheiten. Punkt und Pica sind übliche Maßeinheiten in der Typographie; Punkt kennen Sie wahrscheinlich aus Word oder einem anderen Textverarbeitungsprogramm.

em, ex, px und % hingegen sind relativ. em und ex stammen ebenfalls aus der Typographie und bezeichnen die Höhe des großen M und des kleinen x. Sie sehen schon, warum em und ex relative Angaben sind: Sie bezeichnen keine feststehenden Werte, sondern Werte, die von der gerade verwendeten Schriftgröße abhängen.

Hingegen hängt die exakte Größe eines Pixels vom Umrechnungsfaktor des Ausgabegeräts ab. Bei einem 72-DPI-Ausgabemedium (Monitor) gibt es 72 Punkte je Inch. In Zentimeter umgerechnet bedeutet das: 72 Pixel/Inch entspricht 2,54 Inch/cm = 28,35 Pixel/cm. Dies ist eine häufige Einheit bei Macintosh-Rechnern, bei Windows-Rechnern ist sie jedoch 96-120 Punkt per Inch. Teilweise können Sie unter Windows die DPI-Werte selbst festlegen. Unter Windows XP, sofern das unterstützt wird, finden Sie die Konfiguration unter Start/Systemsteuerung/Anzeige/Einstellungen/Erweitert.

Das Problem bei Pixeln ist jedoch, dass das, was unter der einen Auflösung noch eine akzeptable Größe darstellt, bei der anderen Auflösung schon zu klein sein kann.

Auch wenn Pixel vom Ausgabegerät abhängen und damit relativ sind, werden sie manchmal zu den absoluten Einheiten gezählt, da sie unabhängig von einzelnen Elementen der Webseite sind.

Prozentangaben sind ebenfalls relativ und beziehen sich auf das Elternelement.

Die Verwendung der einzelnen Längeneinheiten wird dann später am Beispiel deutlich, zuerst sollen die Längeneinheiten für die Schriftgröße erläutert werden.




Noch ein Hinweis: Bei Fließkommazahlen müssen Sie in CSS immer einen Punkt anstelle des im Deutschen und in manchen europäischen Ländern gebräuchlichen Kommas schreiben. Außerdem werden die Größenangaben immer ohne Abstand hinter die Zahl geschrieben. Ein Beispiel:

```
body {font-size: 100.01%;}
```

## 3.3 Schriftgröße bestimmen

Zur Definition der Schriftgröße dient `font-size`. Dahinter notieren Sie den gewünschten Wert in der gewählten Einheit:

```
h1 { font-size: 1.2em; }
```

Neben den oben aufgeführten Größenangaben, die Sie prinzipiell überall, wo solche Angaben erwartet werden, in CSS einsetzen können, existieren noch die folgenden speziellen Schlüsselwörter für die Schriftgröße:

- `xx-small`
- `x-small`
- `small`
- `medium`
- `large`
- `x-large`
- `xx-large`

Daneben gibt es auch die relativen Werte `smaller` und `larger`, bei denen die Schriftgröße um eine Stufe verkleinert oder vergrößert wird. Hat das Elternelement die Schriftgröße `medium`, dann würde die Schriftgröße `larger` bei einem Kindelement `large` entsprechen.

Der Skalierungsfaktor, um ausgehend vom mittleren Wert `medium` die anderen Größen zu berechnen, hat sich über die verschiedenen Versionen von CSS hindurch geändert. Bei CSS 1 wurde noch 1,5 vorgeschlagen, bei CSS 2 hingegen sind es 1,2. Entspricht `medium` 12pt, dann ergibt sich für `large` 12pt*1,2, d.h. 14,4pt. In CSS 2.1 hingegen wird nicht mehr von einem festen Faktor ausgegangen, sondern werden unterschiedlich große Stufen festgelegt.

Abbildung 3.2 zeigt die Interpretation der Schriftgrößen in Firefox und Internet Explorer 6.

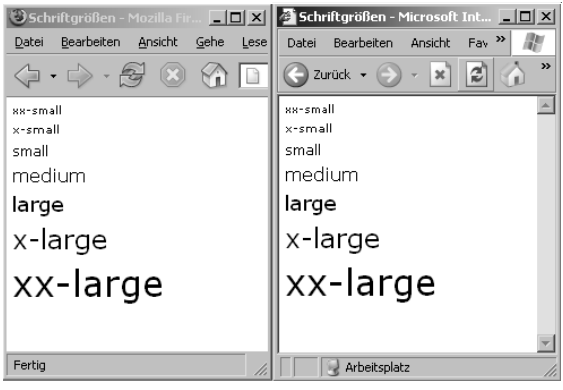



*Abb. 3.2: Schriftgrößenangabe über die Schlüsselwörter*

Der Standardwert bei der Schriftgröße ist `medium`. Alle CSS-Eigenschaften haben sogenannte Standard- oder Defaultwerte, die verwendet werden, wenn nicht explizit eine andere Angabe erfolgt.

Leider werden im Internet Explorer 4 und 5.x alle Schriftgrößen über Schlüsselwörter in einer Stufe größer dargestellt als in anderen Browsern. Der Standardwert ist also nicht `medium`, sondern `small`. Wie Sie mit solchen Problemen umgehen und gesonderte Angaben für diese älteren Internet Explorer-Versionen machen, erfahren Sie in Kapitel 14.

Bei so vielen Möglichkeiten, die Schriftgröße zu bestimmen, stellt sich natürlich die Frage, welches denn die richtige Einheit für Schriften in CSS ist.

Zentimeter und Millimeter, ebenso wie die aus der Typographie stammenden Punkt und Pica, sind wie gesagt absolute Maßeinheiten, da sie von keiner anderen Angabe abhängen. So eignen sie sich auch dann, wenn das Ausgabemedium ebenfalls absolute, feste Maße hat, beispielsweise bei einem Stylesheet für die Druckversion. Denn bei einem Blatt Papier weiß man, wie breit und hoch es in Zentimetern ist. Für den Bildschirm hingegen sind sie weniger geeignet.

Pixel sind hingegen eine relative Angabe, da Pixel in Verhältnis zur Auflösung bestimmt sind. Auf den ersten Blick sind sie damit für Webseiten prädestiniert, und Sie

benutzen Pixel üblicherweise beispielsweise bei der Angabe der Größe von Bildern. Es gibt jedoch zwei Nachteile, wenn Sie Pixel zur Angabe der Schriftgröße verwenden.

Normalerweise kann der Surfer die Schriftgröße von Seiten verändern. Beim Internet Explorer finden Sie diese Option unter Ansicht/Schriftgrad und beim Firefox ebenfalls unter Ansicht/Schriftgrad/vergrößern.

Beim Einsatz von Pixel für Schriftgrößen funktioniert dies jedoch im Internet Explorer nicht: Die Schriftgröße ist dann im Internet Explorer fix und kann nicht verändert, d.h. nicht vergrößert werden. Die absoluten Einheiten wie `pt` etc. ermöglichen natürlich ebenfalls im Internet Explorer keine Skalierung durch den Benutzer.

Auch im Internet Explorer 7 können in `px` angegebene Schriften nicht vergrößert werden, jedoch verfügt der neue Internet Explorer über eine zusätzliche Funktion: den Zoom oder Pagezoom. Damit wird die ganze Webseite vergrößert, inklusive der Grafiken. Die Zoom-Funktion finden Sie in der Statusleiste (vgl. Abbildung 3.3). Übrigens bietet auch der Opera-Browser so eine Zoom-Funktion. Ein Nachteil dieser Zoom-Funktion soll jedoch nicht verschwiegen werden: Es erscheinen üblicherweise schnell Scrollleisten.


*Abb. 3.3: Über die Zoom-Funktion im Internet Explorer 7 wird die gesamte Seite inklusive Grafiken vergrößert*

Solange jedoch Internet Explorer-Versionen kleiner als 7 so weit verbreitet sind, spricht die fehlende Möglichkeit der Schriftgrößenänderung gegen den Einsatz von px als Maßeinheit für die Schriftgröße.

Aber noch einen anderen Nachteil haben Schriftangaben in Pixeln: Es gibt heute immer mehr hochauflösende Displays, auf denen die Schrift in Pixeln eventuell unlesbar klein dargestellt wird.

Die Angabe `em` ist auf den ersten Blick ein bisschen gewöhnungsbedürftig. Wenn Sie für `body` eine Schriftgröße von `1em` festlegen, ist die wirkliche Größe eine andere – je nachdem, was der Surfer als Standardschriftgröße eingestellt hat. Ist es 14 Punkt, so entspricht `1em` 14 Punkt. Hat der Benutzer hingegen eine größere Schrift eingestellt, so verändert sich die Größe von `1em` entsprechend. Üblicherweise, d.h., wenn ein Benutzer keine Änderungen an der Anzeige vornimmt, können Sie davon ausgehen, dass 16px einem em entsprechen. Das kann auch die Basis für Umrechnungen sein.

Durch den Einsatz von em kann gewährleistet werden, dass das Verhältnis der Schriftgrößen untereinander immer gleich wirkt, wie in dem folgenden Beispiel:

```
h1 { font-size: 1.3em; }
p { font-size: 1em; }
```

`h1` erhält eine Schriftgröße von 1.3em, `p` hingegen `1em`, d.h., die Schrift für `h1` ist um 30% größer als die von `p`.

Abbildung 3.4 und Abbildung 3.5 zeigen das Ergebnis, einmal bei normaler Schriftgröße, einmal mit vergrößerter Schrift.

*Abb. 3.4: Normale Schriftgröße*

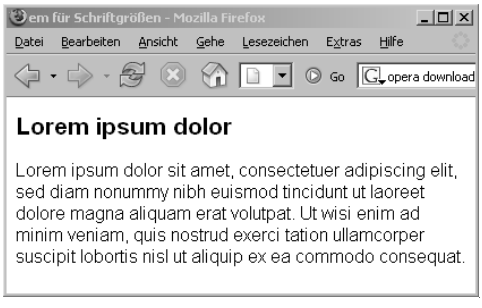
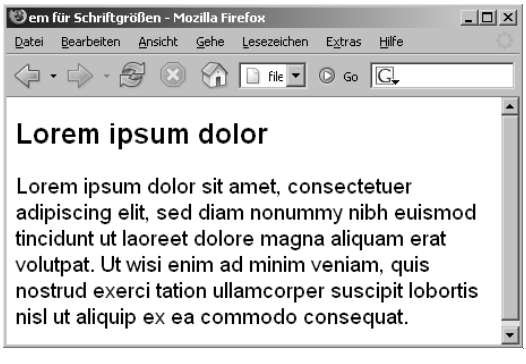



*Abb. 3.5: Vergrößerte Schrift*



Genauso funktionieren die Prozentangaben, wenn sie auf die Schriftgröße bezogen werden. 100% entspricht der normalen Schriftgröße, die Angabe 80% würde eine um ein Fünftel kleinere Schrift bewirken.

Wenn das Projekt entsprechend angelegt ist, ist für Schriftgrößen die erste Wahl em und %.

Leider gibt es jedoch einen Bug beim Internet Explorer, der sporadisch auftritt, wenn man die Schriftgröße relativ angibt; es kann dann passieren, dass die Schriftgrößen unter Ansicht immer auf Sehr klein gestellt sind. Die Auswirkung: unleserlich kleine Schrift. Genaueres dazu können Sie unter *http://www.einfach-fuer-alle.de/artikel/ietext/* nachlesen.

Dies lässt sich jedoch durch einen Trick umgehen: Wenn man bei `body` für die Schriftgröße 100.01% angibt, kann man für die anderen Elemente ganz nach Belieben mit % oder em arbeiten, ohne dass der Fehler auftritt.

Für den Internet Explorer würde auch die Angabe 100% korrekt funktionieren, doch das könnte beim Browser Opera wegen eines Bugs zu einer falschen Skalierung führen. Angaben wie 101% wiederum sind für manche Safari-Versionen problematisch. Deswegen ist diese etwas komische Zahl 100.01% der richtige Wert.




Vergessen Sie bei den Größenangaben nicht die Maßeinheit. Das passiert einem besonders leicht am Anfang, weil man es von HTML anders gewohnt ist. Es gibt nur zwei Fälle, in denen Sie die Maßeinheit in CSS weglassen können: einerseits bei 0, denn 0px sind auch nicht mehr oder weniger als 0pt etc. Der andere Fall, in dem es auch ohne Maßangabe geht, sind die Zeilenhöhen. Dazu kommen wir im nächsten Abschnitt.

Sie sehen, CSS stellt einen mit seinen vielen Möglichkeiten vor die Qual der Wahl, welche man am besten einsetzt.

Absolute Einheiten sind für die Schriftgröße bei der Ausgabe an einen Monitor nicht sinnvoll, fallen also schon einmal weg.

Wenn Sie ein pixelgenaues Design einsetzen, wird wahrscheinlich bei der Schriftgröße ebenfalls Pixel die erste Wahl sein. Damit nehmen Sie jedoch wie erwähnt in Kauf, dass der Benutzer im Internet Explorer die Schriftgröße nicht mehr an seine Bedürfnisse anpassen kann und dass die Schrift auf hochauflösenden Displays sehr klein wirken wird. Abhilfe kann ein sogenannter Fontsizer bieten.

Beim Einsatz eines Fontsizers kann der Benutzer eine vergrößerte Ansicht der Webseite durch Klick auf einen Link erhalten, sozusagen eine Großdruckvariante gerade für die steigende Anzahl von Silver Surfern. Die Verwendung von Fontsizern hat prinzipiell auch einen Vorteil in Kombination mit relativen Maßeinheiten, da man nicht unbedingt davon ausgehen kann, dass alle Surfer wissen, wie sie die Schriftgröße im Browser einstellen können.

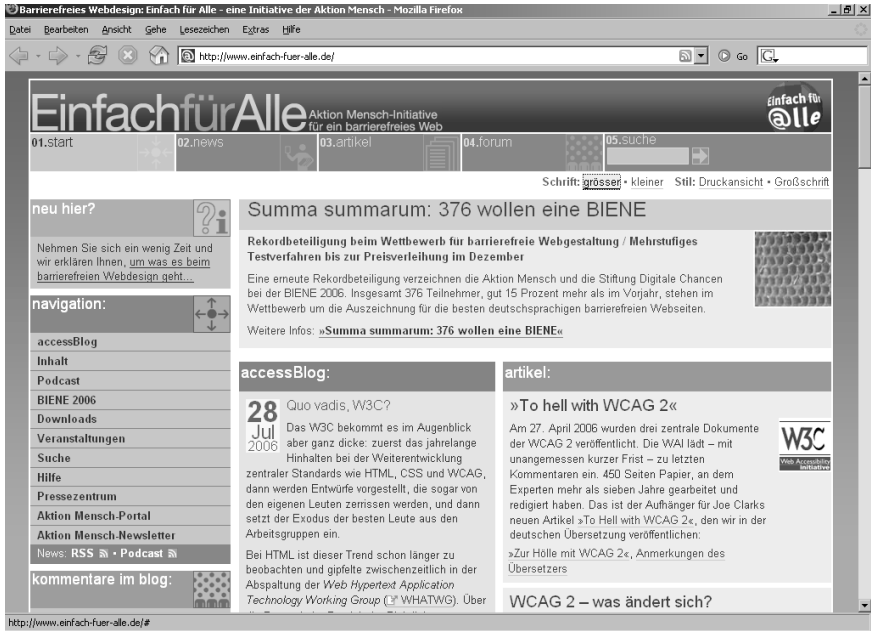



*Abb. 3.6: Der Fontsizer bei www.einfach-fuer-alle.de im Einsatz: vor dem Klick auf* GRÖSSER

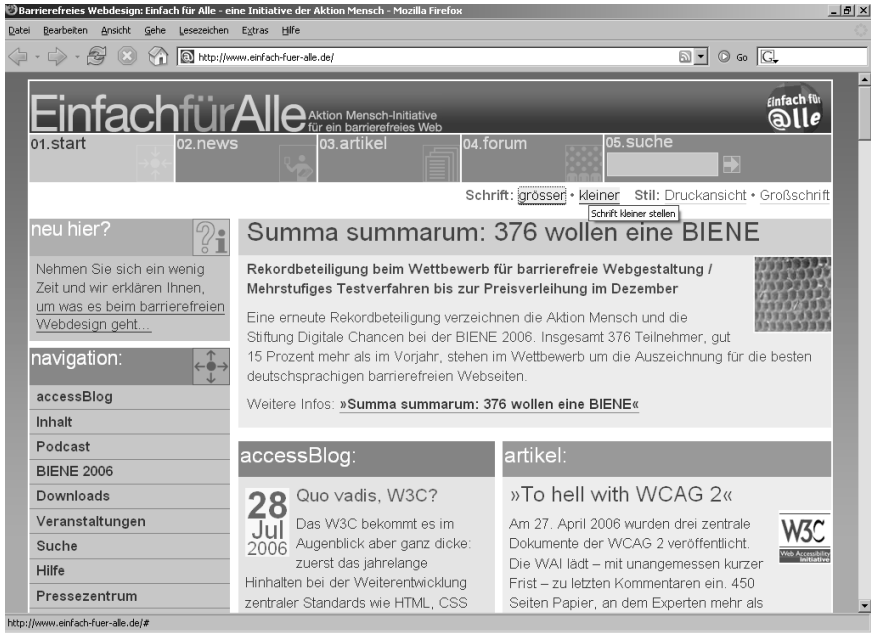



*Abb. 3.7: Fontsizer: nach mehrmaligem Klick auf* GRÖSSER

Den Efa-Fontsizer von *www.einfach-fuer-alle.de* können Sie übrigens in Ihren Projekten einsetzen. Sie finden ihn mit einer Anleitung unter *http://www.einfach-fuer-alle.de/artikel/fontsize/*.




Wenn möglich sollten Sie jedoch bei der Schriftgröße auf em und % setzen. Damit es nicht zu dem oben geschilderten Bug kommt, muss jedoch die Schriftgröße für den `body` auf diesen etwas seltsamen Wert 100.01% gesetzt werden.

Prinzipiell muss man sich bei der Gestaltung von Webseiten überlegen, wie viel Freiheit man den Benutzern lassen möchte. Häufig ist es vom Design her gerade erwünscht, die Schriftgröße fest zu bestimmen, und das ist an sich – im Unterschied zum `font`-Element, wo das noch nicht ging – mit CSS machbar. An sich eine Errungenschaft von CSS – wenn die Ausgabegeräte und die Bedürfnisse der Surfer nicht so unterschiedlich wären. Und über beides haben Sie keinerlei Kontrolle. Die Schriftgröße hat nicht nur etwas mit Optik und Geschmack zu tun, sondern auch mit Lesbarkeit. Im schlimmsten Fall können Surfer manche Informationen nicht mehr aufnehmen – weil die Schrift ihnen zu klein ist.

## 3.4 Zeilenhöhe

Über `line-height` können Sie die Zeilenhöhe festlegen. Damit bestimmen Sie den minimalen Abstand zwischen den Grundlinien der Textzeilen. Ein größerer Wert bewirkt einen größeren Abstand, nicht nur zwischen den Zeilen, sondern auch nach oben und unten.

Im folgenden Beispiel wurde einem Absatz eine Zeilenhöhe von 1.5em zugewiesen, der Absatz darunter hat keine besonderen Angaben und zeigt die Standardzeilenhöhe im Browser:

```
<p style="font-size: 1em; line-height: 1.5em"><strong>font-size: 1em;
line-height: 1.5.em: </strong>Lorem ipsum et everti vituperatoribus vim, eos
ei nemore oportere ...</p>
<p><strong>Standardwerte des Browsers: </strong>Lorem ipsum et everti
vituperatoribus vim, eos ei nemore oportere ...</p>
```

*Listing 3.2: Zeilenhöhe über `line-height` definieren*


*Abb. 3.8: Zeilenhöhe 1.5em und die Standardzeilenhöhe im Vergleich*

Als Werte können Sie die üblichen Längenangaben einsetzen, daneben ist aber auch eine Zahl ohne Maßeinheit möglich, diese gibt dann einen Skalierungsfaktor an. So bestimmt `line-height:1.2` den Faktor 1.2 für die Zeilenhöhe, d.h. eine um 20% größere Zeilenhöhe. Sie sollten die Zeilenhöhe immer zusammen mit der Schriftgröße bestimmen.

Demnach gibt es auf jeden Fall schon einmal drei Möglichkeiten, eine um 20% größere Zeilenhöhe zu bestimmen: 1.2em, 120% oder 1.2. Ein Unterschied zwischen 1.2em, 120% auf der einen Seite und dem Faktor 1.2 auf der anderen Seite zeigt sich jedoch bei der Vererbung. Er wird im nächsten Kapitel behandelt.

## 3.5 Farbangaben

Aus HTML kennen Sie die Farbangaben über hexadezimale Werte, die ein #-Zeichen vorangestellt bekommen. Dabei bestimmen die ersten beiden Stellen den Rotwert, die beiden mittleren den Grünwert und die letzten beiden den Blauwert (RGB-Farbangaben). Wenn Rot, Grün und Blau im gesättigtsten Farbton addiert werden, ist das Ergebnis Weiß (additives Farbschema). So steht dann `#FFFFFF` für Weiß oder `#000000` für Schwarz. Auch in CSS ist das eine Möglichkeit, Farben anzugeben.

So wird dann die Hintergrundfarbe auf Weiß und die Textfarbe für ein Dokument auf Schwarz festgelegt:

```
body { color: #00000; background-color: #FFFFFF; }
```

Wenn bei hexadezimalen Werten bei den jeweiligen Farbangaben die beiden Stellen dieselbe Zahl oder denselben Buchstaben haben, können Sie das auch verkürzt schreiben: Die Kurzform für Weiß ist `#FFF` und für Schwarz `#000`, `#6699CC` und `#69C` sind äquivalent. Übrigens ist hier die Groß- und Kleinschreibung nicht relevant.

Rechnen Sie hingegen lieber mit den gewohnten dezimalen Werten, steht Ihnen folgende Syntax zur Verfügung: `rgb(255, 0, 0)` für ein reines Rot. Hier notieren Sie hinter dem Schlüsselwort `rgb` in Klammern die drei Werte für Rot, Grün und Blau als dezimale Zahlen zwischen 0 und 255 und trennen diese durch Kommas.

Anstelle der dezimalen Werte sind auch Prozente erlaubt, z.B. `rgb(100%, 0%, 0%)`. Hierbei steht 100% für die größtmögliche Intensität einer Farbe und 0% für die Abwesenheit der entsprechenden Farbe.

Eine weitere Möglichkeit besteht darin, einen der 16 ebenfalls aus HTML bekannten englischen Farbnamen einzusetzen. Diese sind:

*Tabelle 3.3: Die 16 vordefinierten Farbnamen*

| | | | |
|---|---|---|---|
| `black` – schwarz | `green` – grün | `navy` – dunkelblau | `gray` – grau |
| `lime` – hellgrün | `blue` – blau | `maroon` – dunkelrot | `olive` – olivgrün |
| `purple` – violett | `red` – rot | `yellow` – gelb | `fuchsia` – magenta |
| `silver` – hellgrau | `aqua` – cyan | `teal` – blaugrün | `white` – weiß |

In CSS 2.1 wurde zusätzlich zu diesen sechzehn Farbnamen auch noch `orange` definiert.

Über diese 16 oder 17 Farbnamen hinaus gibt es noch wesentlich mehr, die zwar nicht offiziell standardisiert, jedoch von Browsern gut unterstützt werden.

Folgende Zeilen bewirken immer dasselbe: Für ein Dokument wird eine schwarze Textfarbe und eine weiße Hintergrundfarbe festgelegt.

```
body {color: black; background-color: white; }
body {color: #000000; background-color: #FFFFFF; }
body {color: #000; background-color: #FFF; }
body {color: rgb(0, 0, 0); background-color: rgb(255,255,255); }
body {color: rgb(0%, 0%, 0%); background-color: rgb(100%,100%,100%); }
```

Übrigens sollten Sie immer, wenn Sie eine Textfarbe bestimmen, auch gleichzeitig eine Hintergrundfarbe angeben. Falls nämlich ein Benutzer selbst eine Hintergrundfarbe über die Browsereinstellungen festlegt und diese Ihrer Textfarbe sehr ähnlich ist, ist die Seite nicht mehr lesbar.

## 3.6 Weitere Zeichenformatierungen

Auch über CSS können Sie einzelne Zeichen fett oder kursiv machen. Alle Eigenschaften haben als ersten Bestandteil `font` und sind so gut zu merken.

Zur Bestimmung der Dicke und Stärke einer Schrift dient `font-weight`. Die wichtigsten Werte sind `normal` und `bold`.

```
.wichtig {font-weight: bold; }
```

Darüber hinaus können Sie auch hier mit relativen Werten operieren, `bolder` für fetter und `lighter` für weniger fett stehen zur Verfügung. Zudem gibt es die numerischen Werte 100, 200 etc. bis 900 – je größer die Zahl, desto fetter die Schrift. `normal` entspricht 400, `bold` hingegen 700. Zwar gibt es bei den numerischen Werten 9 verschiedene Möglichkeiten, die meisten Schriften unterstützen jedoch keine derart feine Abstufung bei der Schriftgewichtung. Abbildung 3.9 zeigt einen Beispieltext in allen Abstufungen – Unterschiede sind teilweise bei den einzelnen Zwischenwerten nicht auszumachen.

Für kursiven Text ist `font-style` zuständig. Mögliche Werte sind `italic`, `oblique` und `normal`. Der Unterschied zwischen `italic` und `oblique` besteht darin, dass bei `italic` ein vorhandener kursiver Schriftschnitt verwendet wird, bei `oblique` hingegen die normale Schrift schräg gestellt wird. Wird `italic` eingesetzt und gibt es keinen vorhandenen kursiven Schriftschnitt, wird `oblique` benutzt. Sie sollten daher immer `italic` verwenden.

```
.bet {font-style: italic; }
```

```
<p class="bet">und hier folgt kursiver Text</p>
```

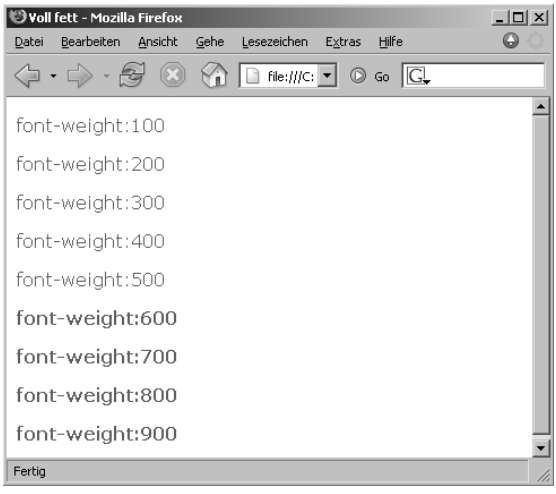



*Abb. 3.9: Angaben für `font-weight`*

Kapitälchen (`font-variant`) eignen sich für Überschriften. Kapitälchen zeichnen sich dadurch aus, dass alle Buchstaben zu Großbuchstaben unterschiedlicher Größe werden. `font-variant` kennt nur zwei Werte: `normal` und `small-caps` für Kapitälchen.

```
h2 {font-variant: small-caps; }
```

Das folgende Beispiel demonstriert `font-style` und `font-variant`:

*Listing 3.3: Ein Absatz mit kursiver Schrift, ein anderer in Kapitälchen*

```
<p style="font-style: italic">font-style:italic: Lorem ipsum dolor sit amet
... </p>
<p style="font-variant: small-caps">font-variant:small-caps: Lorem ipsum
dolor sit amet ... </p>
```


*Abb. 3.10: Kursiv und Kapitälchen*

An sich ist in CSS 2 auch vorgesehen, dass Sie die Zeichensätze mit der Eigenschaft `font-stretch` dehnen können, dies wird jedoch von keinem gängigen Browser unterstützt und ist nicht mehr Teil von CSS 2.1. Ebenso verhält es sich auch mit `font-size-adjust`, das dazu dienen sollte, die Schriftgröße anzupassen, um die Lesbarkeit zu erhöhen. Auch die eigentlich schöne Eigenschaft `text-shadow` zur Erzeugung von Textschatten teilt dieses Schicksal und wurde wegen mangelnder Browser-Unterstützung aus CSS 2.1 gestrichen. `text-shadow` ist derzeit nur in Safari implementiert.

Wenn Sie für ein Element alle möglichen Schriftformatierungen vornehmen möchten, bedeutet das eine Menge Schreibarbeit.

```
.bet {
   font-family: Verdana, sans-serif;
   font-style: italic;
   font-weight: bold;
   font-size: 0.9em;
   font-variant: normal;
   line-height: 1.2;
}
```

Für diese sechs Eigenschaften, die alle mit Ausnahme von `line-height` mit `font` anfangen, gibt es auch eine verkürzte Schreibweise. Dafür notieren Sie alle Angaben direkt hinter `font`:

```
.bet { font: italic bold normal 0.9em/1.2 Verdana, sans-serif; }
```

Die Reihenfolge der ersten drei Werte, d.h. von `font-family`, `font-style` und `font-weight`, ist beliebig. Wenn Sie für einen oder mehrere dieser drei Werte nichts angeben, werden die Standardwerte genommen. Damit ließe sich das Beispiel folglich genauso ohne `normal` schreiben:

```
.bet { font: italic bold 0.9em/1.2 Verdana, sans-serif; }
```

Die letzten beiden Werte, `font-size` und `font-family`, hingegen müssen bei der verkürzten Schreibung über `font` immer angegeben werden und auch in dieser Reihenfolge, d.h. zuerst Schriftgröße, dann Schriftart. Wenn Sie die Zeilenhöhe definieren möchten, dann notieren Sie die Zeilenhöhe hinter der Angabe zu `font-size` nach einem Schrägstrich, beispielsweise so: `x-large/1.2`. Beachten Sie außerdem, dass bei der Angabe einer Schriftenliste wie gewohnt mehrere Schriften durch Komma getrennt werden.

Die kürzestmögliche korrekte Verwendung von `font` besteht demnach in der Angabe der Schriftgröße und der Schriftfamilie:

```
p { font: 80% sans-serif; }
```

## 3.7 Weitere Text- und Absatzformatierungen

Die bisherigen Formatierungen haben sich immer auf Zeichen bezogen. Jetzt geht es um Absatzformatierungen.

### 3.7.1 Absätze einrücken

In der Textverarbeitung oder auch in gedruckten Werken werden häufig zur besseren Lesbarkeit Absätze nicht nur mit einem Abstand oben und unten versehen, sondern zusätzlich eingerückt. Über `text-indent` können Sie das mit CSS machen. Damit wird die erste Zeile eines Absatzes links eingerückt – in Schriften mit umgekehrter Schriftrichtung wäre die Einrückung von rechts:

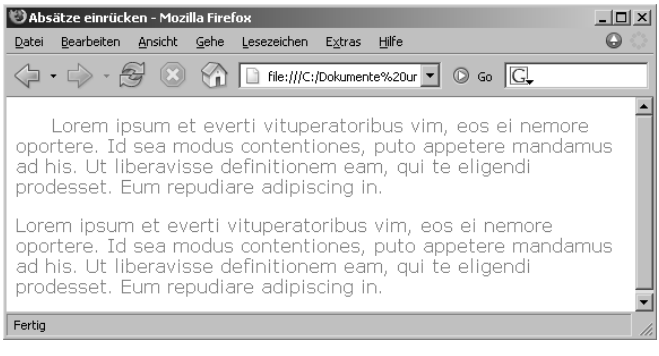
```
p { text-indent: 2em; }
```

Hierzu ein vollständiges Beispiel: Für die Absätze wird prinzipiell eine Einrückung von 2em festgelegt. Darüber hinaus gibt es jedoch die Klasse `.ne` ohne Einrückung. Für das ganze Dokument werden über den Selektor `body` eine Vorder- und Hintergrundfarbe sowie die Schriftart und -größe festgelegt.

*Listing 3.4: Absätze einrücken*

```
<!DOCTYPE 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="de" lang="de">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Abs&auml;tze einr&uuml;cken</title>
<style type="text/css">
/* <![CDATA[ */
body {
   font-size: 100.01%;
   background-color: #FFF;
   color: #C63;
   font-family: Verdana, Arial, sans-serif;
}
p { text-indent: 2em; }
.ne { text-indent: 0; }
/* ]]> */
</style>
</head>
<body>
<p>Lorem ipsum ...</p>
<p class="ne">Lorem ipsum ...</p>
</body>
</html>
```

*Abb. 3.11: Der erste Absatz ist um 2em eingerückt*




An den Absatzeinrückungen lässt sich schön die Wirkung der Größenangabe em verdeutlichen. Im folgenden Beispiel wird noch eine Überschrift `h1` ergänzt. Außerdem wird im Stylesheet definiert, dass die Überschrift `h1` dieselbe Einrückung wie die Absätze erhalten soll, nämlich `2em`:

*Listing 3.5: Unterschiedliche Wirkung von em*

```
p, h1 { text-indent: 2em; }
```

Abbildung 3.12 zeigt deutlich, dass bei der Überschrift, die standardmäßig eine größere Schrift hat, die Einrückung von 2em größer ist als beim Absatz.

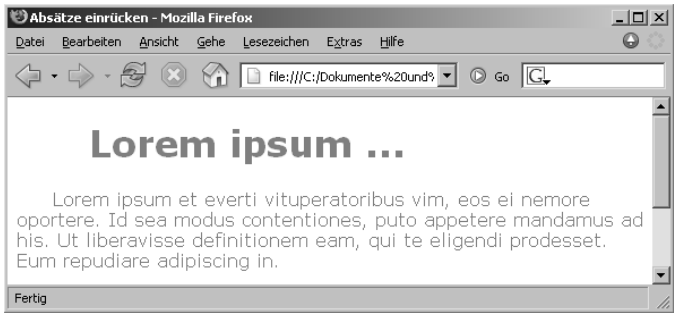



*Abb. 3.12: Einrückung 2em – unterschiedliche Wirkung in Relation zur Schriftgröße*

## 3.7.2 Absatz- und Textausrichtungen

Verwenden Sie zum Ausrichten von Absätzen die Eigenschaft `text-align`. Sie kann die Werte `left`, `right`, `center` und `justify` annehmen. `justify` bewirkt eine Blocksatzausrichtung. Das folgende Beispiel zeigt die verschiedenen Möglichkeiten von `text-align`.

```
<p style="text-align: left"><strong>text-align:left:</strong> Lorem ipsum et
...</p>
<p style="text-align: right"><strong>text-align:right</strong>: Lorem ipsum
et ...</p>
<p style="text-align: center"><strong>text-align:center;</strong> Lorem
ipsum et ...</p>
<p style="text-align: justify"><strong>text-align:justify</strong>: Lorem
ipsum et ...</p>
```

*Listing 3.6: Absatzausrichtung mit `text-align`*

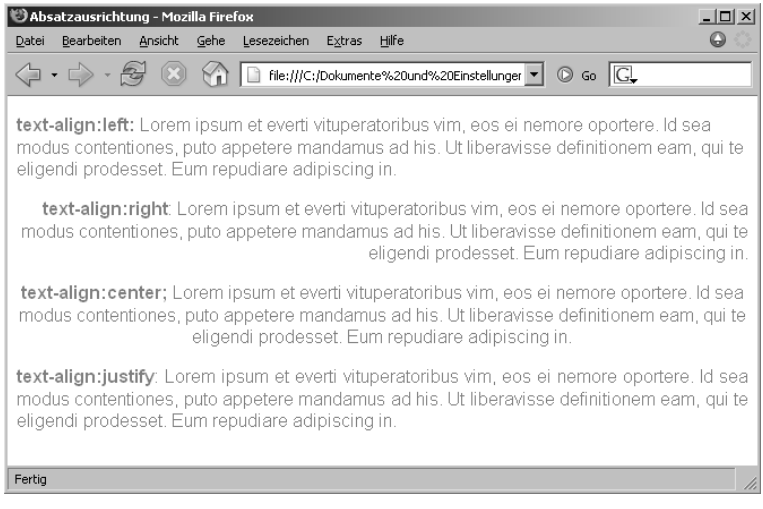



*Abb. 3.13: Absätze mit CSS ausrichten*

Blocksatz wird üblicherweise im Schriftsatz verwendet. Hierbei werden die Wörter getrennt, damit es nicht zu hässlichen großen Wortzwischenräumen kommt. Da jedoch im Browser keine Worttrennung stattfindet, ist `text-align: justify` mit Vorsicht zu genießen.




Zum Zentrieren beispielsweise eines Bereichs (`div`) auf einer Webseite können Sie in standardkonformen Browsern `text-align` hingegen nicht einsetzen. Welche Techniken man hierfür benötigt, erfahren Sie in Kapitel 15.

Für die vertikale Ausrichtung von Text ist `vertical-align` zuständig. Sie können dahinter eines von acht möglichen Schlüsselwörtern (`baseline`, `sub`, `super`, `top`, `text-top`, `middle`, `bottom`, `text-bottom`) angeben oder die Höhe über einen exakten Wert definieren. Bei der Angabe eines Wertes wird das Element entsprechend nach oben (positiver Wert) oder unten (negativer Wert) verschoben. Diese Eigenschaft ist nur für Inline-Elemente vorgesehen.

Noch einmal kurz zur Wiederholung der beiden Begriffe Inline-Element und Blockelement, die im letzten Kapitel bei der Beschreibung von `div` und `span` bereits eingeführt wurden: Inline-Elemente können innerhalb einer Textzeile angeordnet werden, der normale Textfluss wird von Inline-Elementen nicht unterbrochen. Typische Beispiele sind `strong`, `em`, `span` oder auch `a` zur Erzeugung von Links.

Dem gegenübergestellt werden die sogenannten Blockelemente wie Überschriften (`h1-h6`), `div` oder `p` für Absätze. Diese erzeugen am Anfang und am Ende des bezeichneten Blocks einen Zeilenumbruch.

`vertical-align` kann also nicht verwendet werden, um den Inhalt innerhalb eines Blockelements auszurichten. Wie man Elemente auf Webseiten zentriert, wird Thema von Kapitel 15 sein.

Der Standardwert von `vertical-align` ist `baseline` (Grundlinie), d.h., dass ohne weitere Angaben Inline-Elemente an der Grundlinie ausgerichtet werden.

Am häufigsten werden Sie für Text `vertical-align: sub` und `vertical-align: super` benötigen. Damit kann Text tiefer- oder höhergestellt werden, er wird dadurch aber nicht automatisch verkleinert, das müssen Sie zusätzlich angeben:

*Listing 3.7: Text höher- oder tieferstellen mit `vertical-align`*

```
<p>Normaler Text und dann ein Span-Element mit <span style="vertical-align:
sub">vertical-align:sub</span>. Hier geht es wieder normal weiter</p>

<p>Normaler Text und dann ein Span-Element mit <span style="vertical-align:
super">vertical-align:super</span>. Hier geht es wieder normal weiter</p>
```

*Abb. 3.14: Tiefer- und höhergestellter Text*

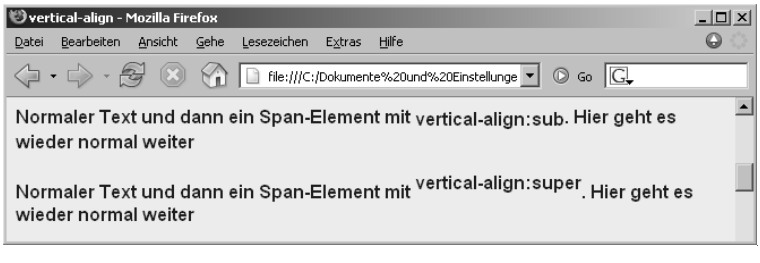


Tabelle 3.4 führt noch einmal die einzelnen Werte mit ihrer Bedeutung auf.

| Wert für vertical-align | Bedeutung |
|---|---|
| baseline | Grundlinie des Elternelements |
| middle | Der Mittelpunkt des Inline-Elements wird am mittleren Punkt des Elternelements ausgerichtet. |
| sub | Tiefergestellt |
| super | Höhergestellt |
| text-top | Die Oberkante der Inline-Box wird an der Oberkante der Schrift des Elternelements ausgerichtet. |
| text-bottom | Die Unterkante der Inline-Box wird an der Unterkante der Schrift des Elternelements ausgerichtet. |
| top | Oberkante der Zeilenbox |
| bottom | Unterkante der Zeilenbox |

*Tabelle 3.4: Werte für vertical-align*

Durch die vertikale Ausrichtung kann die Zeilenhöhe beeinflusst werden.

### 3.7.3 »Text-Ausschmückungen«

Unter dem Begriff »Text-Ausschmückungen« (text-decoration) werden ganz unterschiedliche Effekte zusammengefasst: Unterstreichungen (underline), Überstreichungen (overline), Durchstreichungen (line-through) und Blinken (blink). blink ist ein rascher Wechsel zwischen sichtbar und nicht sichtbar, so wie Sie es vielleicht noch von den browser-proprietären Elementen blink (Netscape) und marquee (Internet Explorer) kennen. In der CSS-Spezifikation heißt es explizit zu blink, dass es nicht von den Browsern unterstützt werden muss – und der Internet Explorer tut das auch nicht.

```
<p style="text-decoration: underline">underline - unterstrichen</p>
<p style="text-decoration: overline">overline - &uuml;berstrichen</p>
<p style="text-decoration: line-through">line-through - durchgestrichen</p>
<p style="text-decoration: blink">blink - blinkend</p>
```

*Listing 3.8: Verschiedene Werte für text-decoration*

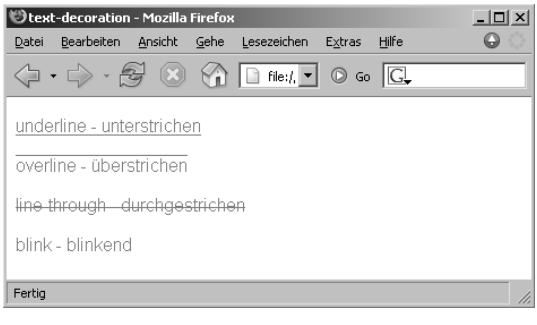



*Abb. 3.15: Verschiedene Werte für text-decoration*

Zugegebenermaßen kommt die Wirkung von `blink` im Druck nicht ganz zur Geltung. Wichtig ist auch noch der Wert `text-decoration: none`, um Unterstreichungen auszuschalten. Das ist gerade bei Links, die standardmäßig unterstrichen sind, eine interessante Alternative.

### 3.7.4 Abstände und andere nützliche Formatierungen

In CSS gibt es zwei Möglichkeiten, um den Abstand innerhalb von Absätzen zu definieren: `word-spacing` bestimmt den Abstand zwischen Wörtern. Sie geben als Wert an, um wie viel der normale Abstand verkleinert oder vergrößert werden soll. `letter-spacing` macht dasselbe wie `word-spacing` – aber jetzt bezogen auf die Abstände zwischen den einzelnen Buchstaben. Damit können Sie Wörter gesperrt ausgeben.

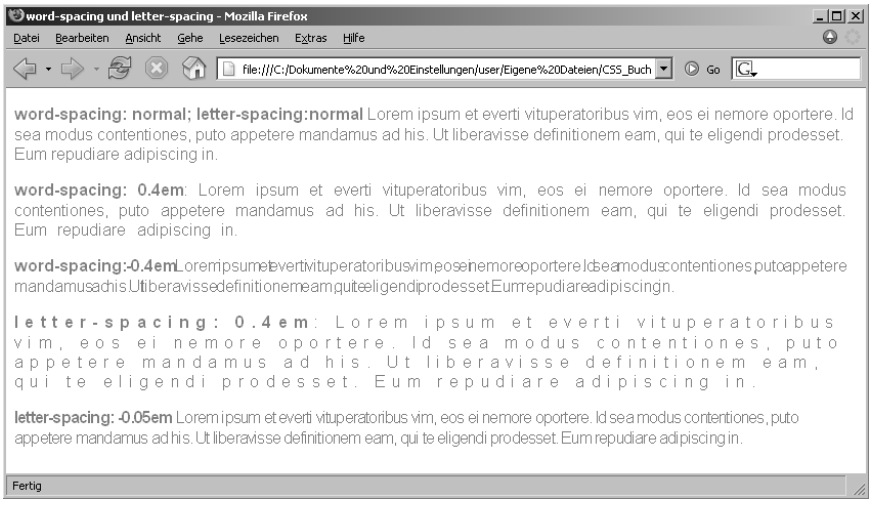
Das nächste Beispiel demonstriert die Wirkung: Der erste Absatz hat normale Werte, und im zweiten Absatz wird ein größerer Wert für `word-spacing` angegeben, der folgende hingegen zeigt die Wirkung eines negativen Werts. Im vierten und fünften wird `letter-spacing` angewendet, zuerst mit einem positiven, dann mit einem negativen Wert:

*Listing 3.9: `word-spacing` und `letter-spacing` (Ausschnitt aus dem Listing)*

```
<p style="word-spacing: normal; letter-spacing: normal"><strong>word-
spacing: normal; letter-spacing: normal</strong> Lorem ipsum ...</p>
<p style="word-spacing: 0.4em"><strong>word-spacing: 0.4em</strong>: Lorem
ipsum ...<p>
<p style="word-spacing: -0.4em"><strong>word-spacing: -0.4em</strong> Lorem
ipsum ...</p>

<p style="letter-spacing: 0.4em"><strong>letter-spacing: 0.4em</strong>:
Lorem ipsum ...</p>
<p style="letter-spacing: -0.05em"><strong>letter-spacing: -0.05em</strong>
Lorem ipsum ...</p>
```

*Abb. 3.16: `word-` und `letter-spacing` im Einsatz*



`word-spacing` wird vom Internet Explorer erst ab Version 6 vollständig unterstützt.




Ebenfalls nützlich ist die Eigenschaft `text-transform`, um Text in Großbuchstaben (`uppercase`) oder in Kleinbuchstaben (`lowercase`) zu verwandeln. Bei `capitalize` wird jeweils der erste Buchstabe jedes Wortes großgeschrieben, eine Funktion, die für das Englische praktischer ist als für das Deutsche.

Auch für fremdsprachige Elemente im Dokument, die in einer Schrift mit anderer Schreibrichtung verfasst sind, ist eine Eigenschaft in CSS vorgesehen: `direction: ltr` ist der Standard und bestimmt die bei uns eingesetzte Schreibrichtung von links nach rechts. `direction: rtl` hingegen legt die Schreibrichtung von rechts nach links fest. Dies wird allerdings erst von neueren Browsern unterstützt.

Die letzte Eigenschaft, die hier erwähnt werden soll, ist `white-space`. Sie dient zur Festlegung, wie Browser mit den Zwischenräumen zwischen Wörtern und Textzeilen umgehen sollen. Der Standardwert ist `normal`, dabei werden mehrere Leerzeichen zusammengefasst. Daneben sind die folgenden Werte möglich:

*Tabelle 3.5: Werte für `white-space`*

| **Werte für** `white-space` | **Bedeutung** |
|---|---|
| `nowrap` | Kein Umbruch auch bei langen Textzeilen – der Surfer muss im Zweifelsfall scrollen. |
| `pre` | Leerzeichen und Zeilenumbrüche werden nicht ignoriert, wie das standardmäßig der Fall ist. Damit hat der Wert `pre` dieselbe Funktion wie das (X)HTML-Element `pre`. |
| `pre-wrap` | Leerzeichen werden nicht ignoriert, Zeilenumbrüche schon. |
| `pre-line` | Zeilenumbrüche werden nicht ignoriert, Leerzeichen schon. |

`pre-wrap` und `pre-line` sind erst in CSS Version 2.1 eingeführt worden und werden derzeit außer von Opera, der `pre-wrap` implementiert, nicht von den aktuellen Browsern unterstützt.

Im folgenden Beispiel sorgt `white-space: pre` dafür, dass die Leerzeichen und Zeilenumbrüche wie im Quelltext ausgegeben werden:

*Listing 3.10: `white-space: pre` zur Formatierung von Quellcode*

```
<!DOCTYPE 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="de" lang="de">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
<title>white-space: pre</title>
<style type="text/css">
/* <![CDATA[ */
body {
   font: 100.01% Verdana, sans-serif;
   background-color: #FFECBF;
   color: #12127D;
}
```

```
h1 { font-size: 1.1em; }
code { white-space: pre; }
/* ]]> */
</style>
</head>
<body>
<h1>ein Code-Beispiel</h1>
<code>
body { font-size: 18px;
       font-family: Verdana, sans-serif;
       background-color: #FFECBF;
       color: #12127D;
}
</code>
</body></html>
```


*Abb. 3.17: `white-space: pre` sorgt für eine Formatierung von Leerzeichen und Zeilenumbrüchen wie im Quellcode*

## 3.8 Wiederholungsfragen und Übungen

1. **Welchen Nachteil hat die Verwendung von Pixeln für die Schriftgröße?**
2. **Wie groß ist ein em?**
3. **Wie lässt sich die folgende Farbe `#FF0000` (Rot) in CSS noch angeben?**
4. **Wie lässt sich der folgende Ausdruck kürzer schreiben?**

```
p {
   font-family: Verdana, Arial, sans-serif;
   font-size: 0.8em;
   font-weight: bold;
   line-height: 1.2;
   color: red;
   background-color: white;
}
```

5. **Wie schaltet man eine Unterstreichung aus?**
6. **Experimentieren Sie mit den unterschiedlichen Angaben!**