

Heinrich-Hertz-Schule Karlsruhe

In Kooperation mit dem HERDT-Verlag stellen wir Ihnen eine PDF inkl. Zusatzmedien für Ihre persönliche Weiterbildung zur Verfügung. In Verbindung mit dem Programm HERDT|Campus ALL YOU CAN READ stehen diese PDFs nur Lehrkräften und Schüler*innen der oben genannten Lehranstalt zur Verfügung. Eine Nutzung oder Weitergabe für andere Zwecke ist ausdrücklich verboten und unterliegt dem Urheberrecht. Jeglicher Verstoß kann zivil- und strafrechtliche Konsequenzen nach sich ziehen.

Isolde Kommer, Marc Haunschild

1. Ausgabe, Januar 2024

ISBN 978-3-98569-177-7

CSS3

Cascading Style Sheets Level 3

Grundlagen

(Stand 2024)

CSS3_2024



Bevor Sie beginnen ...	4	4.4 Text hervorheben mit <code>text-decoration</code> und <code>text-shadow</code>	74
		4.5 Textabstand	76
		4.6 Text einrücken	80
		4.7 Bekannte Schriften neu entdecken	82
CSS-Grundlagen		4.8 Zeilenumbruch	83
		4.9 Übungen	86
1 Verwendung von CSS	6	5 Farben	88
1.1 Warum Formatvorlagen verwenden?	6	5.1 Farbwerte	88
1.2 Vorteile von CSS	7	5.2 Textfarbe ändern	91
1.3 Dokumententyp und Zeichensatz	7	5.3 Hintergrundfarben	93
1.4 Formatvorlagen	8	5.4 Übung	95
1.5 Stylesheets einbinden	10		
1.6 Rangfolge der Stylesheets	13		
1.7 Ausgabemedien	15		
1.8 Alternative Stylesheets	19		
1.9 Übungen	22		
2 CSS-Syntax und Selektoren	23	6 Hintergrundbilder und Filter nutzen	96
2.1 Aufbau von Stylesheets	23	6.1 Einleitung zu Hintergründen	96
2.2 Wie Browser Ihre Webseite untergliedern	24	6.2 Hintergrundgrafiken verwenden	97
2.3 Vererbung	25	6.3 Hintergrund wiederholen	98
2.4 Universal-Selektor	26	6.4 Hintergrund positionieren	100
2.5 Element-Selektor	27	6.5 Hintergrund festsetzen	103
2.6 Nachfahren-Selektor	27	6.6 Mehrfache Hintergründe verwenden	104
2.7 Kind-Selektor	29	6.7 Größe der Hintergrundgrafik	105
2.8 Selektor für Geschwisterelemente	30	6.8 Bereich für Hintergrundposition und Hintergrundanzeige festlegen	107
2.9 Selektor für benachbarte Elemente	30	6.9 Angabe des Hintergrunds in Kurzform	109
2.10 Klassen-Selektor	31	6.10 CSS-Filter	110
2.11 ID-Selektor	34	6.11 Übungen	113
2.12 Attribut-Selektor	37		
2.13 Pseudoklassen	39	7 Listen gestalten	115
2.14 Pseudoelemente	44	7.1 Einleitung zu Listen	115
2.15 Weitere Selektoren	46	7.2 Listentyp	115
2.16 Kommentare	51	7.3 Aufzählungszeichen einrücken	118
2.17 Übung	51	7.4 Listengrafik ändern	118
		7.5 Kurzform der Aufzählung	119
		7.6 Übung	120

HTML-Elemente mit CSS formatieren

3 Eigenschaften von Schrift	52	8 Abstände und Rahmen	121
3.1 Schriftfamilie	52	8.1 Einführung in das Box-Modell	121
3.2 Schnitte einer Schrift	56	8.2 Rahmen	122
3.3 Maßeinheiten	59	8.3 Separate Wertangaben	125
3.4 Schriftgröße	60	8.4 Kurzform von Rahmen	126
3.5 Eigenschaften vererben oder zurücksetzen	62	8.5 Konturen mit <code>outline</code>	127
3.6 Angabe der Schrift in Kurzform	65	8.6 Abgerundete Ecken	128
3.7 Übungen	67	8.7 Rahmen mit Bildern	131
4 Texte gestalten	68	8.8 Innenabstand mit <code>padding</code>	134
4.1 CSS-Eigenschaften zur Textgestaltung	68	8.9 Probleme mit Innenabständen und Rahmen bewältigen	136
4.2 Text ausrichten	68	8.10 Außenabstand mit <code>margin</code>	138
4.3 Groß- und Kleinschreibung beeinflussen	72	8.11 Übungen	141

9 Tabellen	143	12 Layout mit CSS erstellen	188
9.1 Die Eigenschaft <code>display</code> und Einführung in CSS-Tabellen	143	12.1 Zweispalter mit Kopfbereich	188
9.2 Darstellung von Elementen ändern	143	12.2 Flexbox – flexible, responsive Layouts ohne @media-Regel	192
9.3 Tabellen beschriften	147	12.3 Eine responsive Navigation mit Flexbox umsetzen	194
9.4 Rahmen darstellen	148	12.4 Benutzerfreundliche Menüs erstellen	196
9.5 Rahmenabstand	148	12.5 CSS-Grid	200
9.6 Leere Tabellenzellen	149	12.6 Unterschiede der Layoutmethoden	204
9.7 Tabellenlayout	149	12.7 Übungen	206
9.8 Übungen	151		
Fortgeschrittene CSS-Techniken		13 Formulare gestalten	207
10 Positionieren mit CSS	153	13.1 Einführung in die Gestaltung von Formularen	207
10.1 Elemente positionieren	153	13.2 Einzelige Textfelder formatieren	207
10.2 Größe und Seitenverhältnis eines Elements angeben	157	13.3 Mehrzeilige Textfelder, <code>legend</code> und <code>fieldset</code> formatieren	210
10.3 Überlauf festlegen	162	13.4 Auswahlboxen formatieren	212
10.4 Textüberlauf beeinflussen	163	13.5 Schaltflächen gestalten	214
10.5 Element umschließen	165	13.6 HTML5 und CSS3: Formulartypen und Pseudoklassen	217
10.6 Elemente verbergen	167	13.7 Komplettes Formular	221
10.7 Ebenen in HTML-Dokumenten meistern	169	13.8 Übung	222
10.8 Übungen	172		
11 Inhalte generieren: Zähler, Variablen und Berechnungen	173	14 Wie geht es weiter?	224
11.1 Einführung in CSS-generierte Inhalte	173	14.1 CSS ist umfangreich und komplex	224
11.2 Anführungszeichen setzen	173	14.2 CSS ist universell	228
11.3 Inhalte einfügen	176	14.3 CSS ist individuell	229
11.4 Zähler einbinden	178	14.4 Wo Sie sich weiterbilden können	230
11.5 Rechnen in CSS	181	14.5 Sinnvoll weiterbilden	232
11.6 Custom Properties (CSS-Variablen)	182		
11.7 Mauszeiger anpassen	183	Stichwortverzeichnis	234
11.8 Übungen	186		

Bevor Sie beginnen ...

Voraussetzungen und Ziele

Hinweise zu Soft- und Hardware

Nachdem CSS als Standard verabschiedet wurde, begann die Implementierung in den Browsern. Inzwischen ist diese in allen aktuellen Browsern sehr weit fortgeschritten. Um fehlerhafte Darstellungen von Webseiten zu vermeiden, sollten Sie daher aktuelle Browserversionen verwenden. Generell lässt sich sagen: Je älter die verwendete Browserversion, desto schwächer ist die Unterstützung von Webstandards (das betrifft nicht nur CSS, sondern zum Beispiel auch HTML und JavaScript).

Sofern nicht anders vermerkt, wird im Folgenden davon ausgegangen, dass Sie die unten genannten Browser oder neuere Versionen verwenden.

	Name	Erhältlich unter der Internetadresse:
	Microsoft Edge	https://www.microsoft.com/de-de/windows/microsoft-edge
	Firefox	https://www.mozilla.org/de/firefox/
	Opera	https://www.opera.com/de
	Safari	https://www.apple.com/de/safari/
	Chrome	https://www.google.com/chrome/browser/desktop/index.html

Die meisten Anwender verwenden relativ neue Versionen. Das liegt nicht zuletzt an den immer komfortableren Update-Mechanismen der Browser – bei Google Chrome bekommt der Anwender davon gar nichts mehr mit. Dennoch gilt das nicht für jeden Besucher Ihrer Website. Daher ist es sinnvoll, sich die eigene Seite auch mal mit älteren Browsern anzusehen.

Konventionen

Hervorhebungen im Text

Damit Sie bestimmte Elemente auf einen Blick erkennen und zuordnen können, werden diese im Text durch eine besondere Formatierung hervorgehoben.

Kursivschrift

Bezeichnungen für Programmelemente wie Register etc. sowie Dateinamen, Ordnernamen und Internetadressen werden *kursiv* geschrieben.

Fettschrift

Wichtige Begriffe werden **fett** hervorgehoben.

Courier new Programmtext, Programmnamen, Funktionsnamen, Variablennamen, Datentypen, Zeichen, Operatoren etc.

Courier New Kursiv kennzeichnet Zeichenfolgen, die vom Anwendungsprogramm ausgegeben oder ins Programm eingegeben werden.

- [] Bei Darstellungen der Syntax einer Programmiersprache kennzeichnen eckige Klammern optionale Angaben.
- / Bei Darstellungen der Syntax einer Programmiersprache werden alternative Elemente durch einen Schrägstrich voneinander getrennt.

In den Beispielen wird teilweise ein Blindtext verwendet, der nur dem Anschein nach in lateinischer Sprache verfasst ist. Dieser Blindtext entspricht im Satzbild (Länge der Wörter und Sätze) durchschnittlichem deutschen Mengentext. Der Text ergibt inhaltlich keinen Sinn und dient lediglich als Beispiel.

HERDT BuchPlus – unser Konzept:

Problemlos einsteigen – Effizient lernen – Zielgerichtet nachschlagen

(weitere Infos unter www.herdt.com/BuchPlus)

Nutzen Sie dabei unsere maßgeschneiderten, im Internet frei verfügbaren Medien:



1

Verwendung von CSS

1.1 Warum Formatvorlagen verwenden?

In Apps zur Textverarbeitung wie etwa Microsoft Word oder Open Office können Sie Text direkt formatieren (zum Beispiel eine Überschrift markieren und eine größere Schrift verwenden, was sich nur auf den ausgewählten Text auswirkt) oder die Vorgaben für alle Überschriften anpassen. Diese werden in Formatvorlagen zusammen mit dem Aussehen für alle anderen Texte (z. B. Listen und Absätze) gespeichert. Wenn Sie Formate in einer Vorlage festlegen (zum Beispiel größere Schrift für die Überschrift erster Ordnung), werden alle Überschriften erster Ordnung größer. Durch die Verwendung von Formatvorlagen wird sichergestellt, dass gleichartige Texte identisch aussehen.

Ganz ähnlich funktionieren Cascading Style Sheets (CSS): Sie schreiben CSS-Formatvorlagen für Textauszeichnungssprachen wie HTML oder XML. In diesem Buch wird auf die Formatierung von Webseiten, also HTML-Dokumenten, eingegangen.

CSS - Die Einführung

Fortune plango vulnera *stilantibus ocellis*, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata. In Fortune solo sederam elatus, prosperatis vario flore coconatus; quiiquid enim florui felix et beatus, nunc a summo corru gloria privatus. Fortune rota volvitur descendere minoratus, alter in altum tollitur, nimis exaltatus rex sedet in vertice; caveat ruinam! Nam sub axe legimus hecubam redinam.

Qui autem auscultare nolet, exsurgat foras, ut sit, ubi sedat ile qui auscultare vult. Id nos Latine gloriosum dicimus. Quia adsedistis causa in festivo loco, comoedial quam nos actuari sumus et argumentum et nomen vobis eloquar. Alazon Graece huic nomen est comoediae: Id nos Latine gloriosum dicimus.

Omnia Sol temperat purus et subtilis, novo mundo [reserat faciem Aprilis](#); ad Amorem prosperat animus herilis et iocundis imperat deus puerilis. Rerum tanta novitas in solemni vere et veris auctoritas iubet nos gaudere, vias prebet solitas, et in tue vere [res est et probitas tuum retinere](#). Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.

Mit Stylesheets formatierte Webseite (*kap01\start.htm*)

In diesem Beispiel (*kap01\start.htm*) wurden der Überschrift, den beiden Absätzen und dem Text im Kasten per CSS Formate zugewiesen.

Webseiten können und sollten ausschließlich mit CSS formatiert werden.

1.2 Vorteile von CSS

✓ **Trennung von Inhalt und Präsentation**

Formatvorlagen (Stylesheets) dienen der Gestaltung logisch strukturierter Dokumente wie HTML oder XML. Wird die Trennung von Inhalt (HTML) und Präsentation (CSS) konsequent eingehalten, kann die Darstellung von Webseiten im Browser weitgehend beeinflusst werden, ohne die Inhalte zu manipulieren.

✓ **Öffentliche Dokumentation**

Die Standards, die CSS zugrunde liegen, sind öffentlich dokumentiert. Alle Browserhersteller können somit diese Standards unterstützen. Das zugrunde liegende Betriebssystem spielt keine Rolle mehr, wodurch Webseiten im Idealfall identisch aussehen – egal ob Windows, Linux oder macOS verwendet wird.

✓ **Unterstützung unterschiedlicher Medien**

CSS erlaubt es, Formatierungsanweisungen ganz gezielt für bestimmte Geräte anzulegen. Das bedeutet, dass die Inhalte zum Beispiel für den Druck anders aufbereitet werden können, als sie auf einem Bildschirm angezeigt werden.

✓ **Unterstützung verschiedener großer Bildschirme**

Sie können unterschiedlich große Bildschirme gezielt ansprechen. Dadurch können Sie für Smartphones, Tablets, Laptops und ultrahochauflösende Bildschirme angepasste Layouts ausliefern.

✓ **Effizienz**

Bei konsequenter Anwendung wird ein einheitliches Layout sichergestellt. Änderungen, die sich auf alle einzelnen Seiten auswirken, müssen nur an einer Stelle vorgenommen werden.

✓ **Verständlichkeit**

CSS ist eine modulare Sprache, die für Menschen gut lesbar ist.

Durch die stetige Weiterentwicklung kommt es dazu, dass nicht alle Browser gleichzeitig alle Neuerungen korrekt umsetzen. Das kann vor allem bei der Verwendung von neuen CSS3-Features zu Unterschieden in den Browser-Darstellungen führen. Welche Formatierungen die Browser unterstützen, finden Sie unter <https://caniuse.com>.

1.3 Dokumententyp und Zeichensatz

Der HTML5-DOCTYPE wird wie folgt angegeben (Groß- und Kleinschreibung spielt keine Rolle):

```
<!DOCTYPE html>
```

Obwohl in HTML5 fast alles erlaubt ist, was auch in HTML 4.01 und XHTML 1.0 erlaubt war, wurden doch einige Elemente und Attribute aus dem Standard entfernt. Wieder andere Elemente haben eine neue semantische Bedeutung erhalten. Eine Übersicht über die Unterschiede zu früheren Versionen finden Sie unter <https://www.w3.org/TR/html5-diff/>.

Unicode-Zeichensatz UTF-8

Als Zeichenkodierung für Ihre HTML- und CSS-Dateien sollten Sie UTF-8 verwenden, um möglichst flexibel zu sein.

Dazu notieren Sie Folgendes nach dem öffnenden Tag <head>:

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

! Im folgenden ersten Beispiel sind Dokumententyp und Zeichenkodierung angegeben. Im weiteren Verlauf dieses Buches wird zugunsten der Übersichtlichkeit bei den **abgedruckten Quelltexten** auf die Angabe des DOCTYPE und des Zeichensatzes verzichtet. **Diese sollten Sie in realen Webseiten aber immer notieren!**

Die Beispieldateien zu diesem Buch sollen valide sein und funktionieren. Daher werden Sie in den eigentlichen Dateien alle notwendigen Angaben vorfinden.

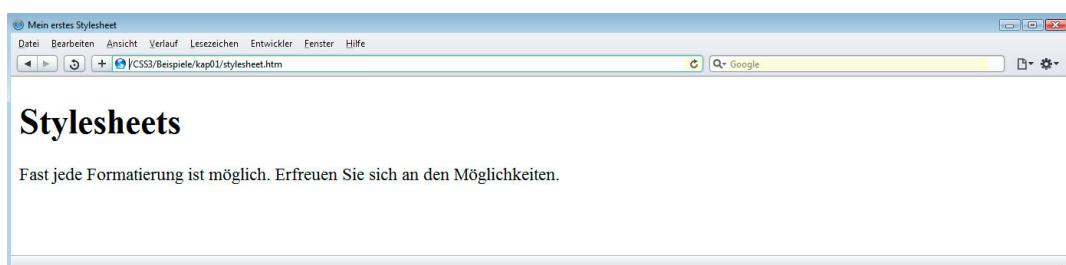
1.4 Formatvorlagen

Beispiel: *kap01\stylesheet.htm*

- ▶ Erstellen Sie in einem Texteditor das folgende HTML-Dokument.

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="utf-8">
    <title>Mein erstes Stylesheet</title>
</head>
<body>
    <h1>Stylesheets</h1>
    <p>Fast jede Formatierung ist möglich. Erfreuen Sie sich an den Möglichkeiten.</p>
</body>
</html>
```

- ▶ Speichern Sie den Quellcode als Datei *stylesheet.htm*.
- ▶ Öffnen Sie die Datei im Browser.



HTML-Dokument: Darstellung unter Verwendung der Browser-Formatvorlage

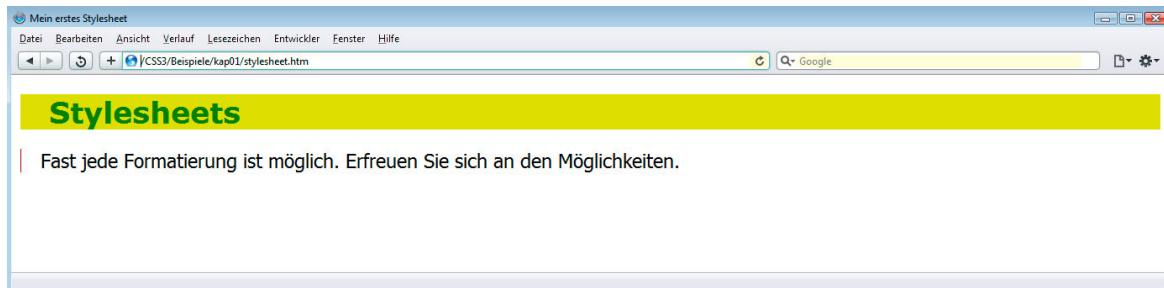
- ▶ Fügen Sie in die Datei nach dem Titel der Webseite die nachfolgend grau unterlegten Zeilen mit den Stilangaben ein.

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="utf-8">
    <title>Mein erstes Stylesheet</title>
    <style>
        h1 { font-family: Verdana, Arial, sans-serif;
              font-size: 1.5em;
              color: #00ff00;
              background-color: #dd0000;
              text-indent: 1em; }
        p { font-family: Tahoma, Arial, sans-serif;
              border-left: 1px solid #880000;
              text-indent: 1em; }
    </style>
</head>
<body>
    <h1>Stylesheets</h1>
    <p>Fast jede Formatierung ist möglich. Erfreuen Sie sich an den Möglichkeiten.</p>
</body>
</html>
```

Die eingefügten Zeilen beinhalten Formatierungsanweisungen. Die verwendete Sprache, in der sie notiert werden, heißt Cascading Style Sheet (CSS). So entstehen Formatvorlagen, die denen eines Textverarbeitungsprogramms entsprechen. Das Beispiel enthält Regeln zur Darstellung von zwei HTML-Elementen: p (Absätze) und h1 (Überschriften erster Ordnung). Wird eines dieser Elemente verwendet, wird es vom Browser den hier notierten Vorgaben entsprechend dargestellt.

Farben und Hintergrundfarben sind besonders deutliche Möglichkeiten der Hervorhebung. In den Beispielen ist es nicht möglich, ohne sie auszukommen. Erst durch Farben werden zum Beispiel die Dimensionen von Elementen sichtbar. In diesem Buch wird in der Regel die übliche hexadezimale Schreibweise für Farbwerte verwendet. Wie Farben notiert werden, wird in Kapitel 5 erklärt.

- ▶ Speichern Sie die Änderung und öffnen Sie die Datei in einem Browser.



HTML-Dokument: Darstellung unter Verwendung der Browser-Formatvorlage und einer Autoren-Formatvorlage

Das Beispiel beinhaltet Formatierungsregeln für zwei verschiedene Elemente, nämlich `h1` und `p`. Schauen Sie sich die Definition für `h1` etwas genauer an.

```
h1 { font-family: Verdana, Arial, sans-serif;  
    font-size: 1.5em;  
    color: #00ff00;  
    background-color: #dd0000;  
    text-indent: 1em; }
```

Diese Zeilen teilen dem Browser mit, dass eine Überschrift erster Ordnung folgendermaßen darzustellen ist:

- ✓ Schriftart (`font-family`): Verdana, Arial oder eine andere seriflose Standardschrift (`sans-serif`)
- ✓ Schriftgröße (`font-size`): 1.5em (`em` = Breite des Buchstabens „m“. 1.5em entspricht 150 % der normalen Schriftgröße)
- ✓ Textfarbe (`color`): grün
- ✓ Hintergrundfarbe (`background-color`): gelbliche Tönung
- ✓ Texteinzug (`text-indent`): 1em

1.5 Stylesheets einbinden

Stylesheet-Angaben können auf drei verschiedene Arten in ein HTML-Dokument eingebunden werden:

- ✓ per Attribut in einem öffnenden Tag (`style="Eigenschaft:Wert;"`),
- ✓ im Kopf des HTML-Dokuments als Inhalt des Elementes `style`,
- ✓ in einer externen Datei. Der Pfad dieser Datei sollte dem Browser mittels `link`-Element mitgeteilt werden.

Stylesheets im Dokumentenkopf festlegen

Diese Methode kennen Sie bereits aus dem vorherigen Abschnitt. Beim Erstellen des ersten Dokuments haben Sie die Stylesheets im Kopf des Dokuments definiert.

```
<style></style>
```

Dies ist das Grundgerüst zum Definieren von Stylesheets. Der HTML-Tag `<style>` leitet die Formatdefinitionen ein.

Stylesheet-Angaben, die auf diese Weise im Kopf der HTML-Datei definiert werden, sind nur für diese eine HTML-Datei gültig. Damit verlieren Sie einen Hauptvorteil von Stylesheets, nämlich die Möglichkeit, das Aussehen vieler Seiten über eine einzige externe Datei zentral zu steuern (zentrale Formate). Möchten Sie mehreren Dateien dieselben Eigenschaften zuweisen, müssen Sie die Stylesheets in einer externen Datei definieren und in die HTML-Dateien einbinden.

Die Beispiele in diesem Buch zeigen, wann es sinnvoll ist, CSS im HTML-Head eines Dokumentes zu notieren: wenn Sie mit nur einer einzigen Datei Inhalte formatiert weitergeben oder präsentieren wollen. Da ein Webauftritt in aller Regel aus vielen Seiten besteht, sollten Sie die Formate in eigenständigen Dateien notieren. Der Browser wird diese Dateien laden und abspeichern. Alle Seiten können auf diese Datei zugreifen. So können alle Formate zentral gepflegt und verwendet werden. Sie müssen nur ein einziges Mal und nicht mit jeder Einzelseite neu übertragen werden.



CSS ist eine eigene Sprache. Innerhalb des Style-Elementes (also zwischen `<style>` und `</style>`) dürfen keine HTML-Tags benutzt werden.

Stylesheets in eine externe Datei auslagern

Statt das Layout des Dokumentes immer wieder in allen HTML-Dokumenten festzulegen, können Sie eine zentrale Datei mit den Formatierungsanweisungen anlegen und von jeder einzelnen Webseite darauf zugreifen. Diese externe CSS-Datei enthält die Definitionen für alle Seiten Ihrer Website. Ändern Sie beispielsweise in dieser Datei die Angaben zur Schriftart, wird die Änderung in allen Dokumenten berücksichtigt, welche diese Vorlage verwenden.

Bei der CSS-Datei handelt es sich um eine **reine Textdatei**, die Sie in jedem Texteditor bearbeiten können. Verwenden Sie als Zeichensatz UTF-8. Die Endung des Dateinamens sollte `.css` lauten (nicht zwingend erforderlich).

```
<link rel="stylesheet" href="dateiname.css">
```

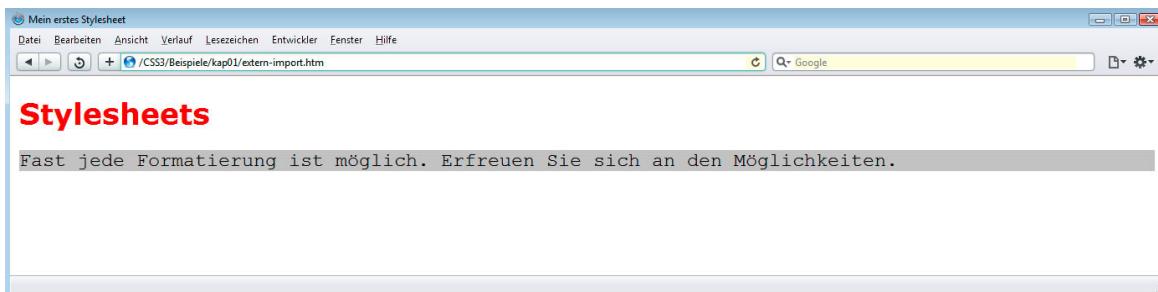
Diese Zeile fügen Sie in den Kopfbereich (`head`) der Webseite ein. Damit teilen Sie dem Browser mit, dass er die hier genannte Datei zur Darstellung des aktuellen HTML-Dokumentes verwenden soll. Der HTML-Tag `<link>` verweist auf eine externe Datei und `rel="stylesheet"` stellt den Bezug zu einem Stylesheet (Formatvorlage) dar. Das Attribut `href` enthält den Pfad zur gewünschten Datei.

- ▶ Erstellen Sie die nachfolgende CSS-Datei mit den Formatdefinitionen:

```
h1 { font-family: Verdana, Arial, sans-serif;  
     font-size: 1.5em;  
     color: red; }  
p { font-family: "Courier New", monospace;  
    background: silver; }
```

- ▶ Speichern Sie die Datei unter dem Namen `layout.css`.
- ▶ Öffnen Sie in einem Texteditor das HTML-Dokument, das Sie bereits mit Stylesheets formatiert haben.
- ▶ Löschen Sie alle Zeilen, die Formatdefinitionen zwischen `<style>` und `</style>` enthalten.
- ▶ Fügen Sie folgende Zeile ein:
`<link rel="stylesheet" href="layout.css">`

- ▶ Speichern Sie das HTML-Dokument *extern-import.htm* im selben Verzeichnis wie die Datei *layout.css*.
- ▶ Öffnen Sie das HTML-Dokument in Ihrem Browser.



HTML-Dokument mit externer Formatvorlage (*kap01\extern-import.htm*)

Die erstellte Formatdatei *layout.css* können Sie auch in andere HTML-Dokumente einbinden. Die Informationen innerhalb der Elemente *h1* und *p* erhalten dort ebenfalls dieselbe Formatierung.

Ändern Sie die Schriftgröße von Absätzen *p* auf *font-size: 75%*, ändert sich die Darstellung automatisch in allen Webseiten, in denen die CSS-Datei eingebunden ist. Ohne zentrale Vorlage müssten Sie jedes einzelne HTML-Dokument öffnen und die Schriftgröße für jeden Absatz oder wenigstens einmal pro Dokument anpassen. Mit einer zentralen Vorlage sind umfangreiche Änderungen am Gesamtlayout Ihrer Webseite effizient und konsistent umsetzbar.

Stylesheets importieren

Eine weitere Möglichkeit ist das Importieren von Stylesheets. Dies funktioniert grundsätzlich genauso wie das Laden aus einer Datei, jedoch über die CSS-spezifische Syntax. Dieser Import muss vor allen anderen CSS-Definitionen erfolgen, ansonsten werden die importierten Stylesheets nicht in das HTML-Dokument implementiert.

```
<style>
  @import url(dateiname.css);
</style>
```

Die Importdirektive zum Laden der Formatvorlage wird im HTML-Dokument im Style-Element notiert. Die Funktion *@import url()* lädt die angegebene CSS-Datei und stellt die Formatanweisungen für die Webseite zur Verfügung. Diese Methode ist weniger performant als das Einbinden per Link-Element. Es handelt sich hierbei um CSS-Syntax, daher muss die Angabe im Style-Element notiert werden. Als CSS-Angabe kann diese auch **am Anfang** einer CSS-Datei verwendet werden. So können Sie in eine Datei Formate importieren. Einige Entwickler trennen lange Formatvorlagen in mehrere Dateien auf, z. B.: *header.css*, *navigation.css*, *footer.css* usw.

! Für jede weitere Datei ist eine zusätzliche Anfrage an den Webserver nötig (http-Request). Jede Anfrage kostet Zeit!

Elemente mittels Attribut formatieren

Sie können jedem Element auch direkt im HTML-Code eine Formatierung zuweisen. Dazu geben Sie die Stylesheet-Deklarationen als Wert eines Attributes für das zu formatierende Element an.

```
<Element style="Eigenschaft:Wert;"> ... </Element>
```

Die Formatierung wird als Attribut `style` dem entsprechenden Element hinzugefügt. Zu beachten ist hierbei, dass Sie die Anführungszeichen setzen müssen.

Beispiel: *kap01\style-inline.htm*

```
<p style="font-family:Verdana,Helvetica,Arial,sans-serif;  
color:#0000ff">Text</p>
```



Diese Methode hat gravierende Nachteile und sollte möglichst vermieden werden: Denn die konzeptionelle Trennung von Layout und logischer Strukturierung der Inhalte wird damit unterlaufen. Auf Englisch nennt man diese Art der Formatierung „presentational markup“.

Alle Elemente, die formatiert werden sollen, müssen das Style-Attribut mit entsprechenden CSS-Regeln erhalten und bei jeder Änderung müssen alle Elemente wieder angefasst werden. Weil Sie dabei leicht ein Element vergessen können, ist dieses Vorgehen fehleranfällig und erschwert die Wartbarkeit einer Website.

Die vielen unnötigen Wiederholungen aller Angaben müssen bei jedem Dateiabruf mit übertragen werden. Das kostet Performanz und Effizienz und damit Geld und Energie.

1.6 Rangfolge der Stylesheets

CSS bietet die Möglichkeit, mehrere Formatvorlagen zu nutzen. Unabhängig davon, ob Sie Formate im Kopfbereich der Webseite definieren, direkt am Element angeben oder von einer externen Datei einbinden: Die einzelnen Formatierungen können sich gegenseitig überschreiben. Diesem Prinzip der **Kaskadierung** verdankt CSS seinen Namen. Damit Sie das Überschreiben beeinflussen können, existiert eine Rangfolge.

Grundsätzlich gilt: Die zuletzt vom Browser bearbeitete Formatierung überschreibt frühere Formatangaben. Von der Reihenfolge unabhängig überschreiben genauere (spezifischere) Angaben ungenauere Angaben (vgl. Kapitel 2).

Im head-Bereich einer Webseite können Sie eine externe CSS-Datei einbinden und zusätzlich lokale Definitionen festlegen. Hier ist die Reihenfolge der Einbindung wichtig. Nutzen Sie z. B. eine extern bereitgestellte CSS-Datei und möchten Sie Änderungen nur für die Einzelseite vornehmen, müssen Sie zuerst die externe Datei einbinden und dann die lokale Anpassung vornehmen. Der Browser geht bei der Bearbeitung linear vor, das heißt, er beginnt am Anfang der HTML-Datei und arbeitet sich Zeichen für Zeichen weiter durch alle Angaben.

Wenn Sie zuerst die externe Datei einbinden und erst danach im Style-Element Angaben für die Einzelseite machen, überschreiben Sie damit **für diese eine Seite** die zuvor geladenen Angaben aus der externen Datei.

```
<link rel="stylesheet" href="dateiname.css">
<style>
/* Notieren Sie hier Ihre Formatierungsanweisungen */
</style>
```

Wenn Sie die externe Datei erst nach der lokalen Definition einbinden, können Sie lokale Änderungen vornehmen und werden womöglich keine Änderung sehen. Im ungünstigsten Fall würde Ihnen die externe Formatierung immer Ihre lokale Festlegung überschreiben.

Die Angabe von Formatierungen über das **Attribut style** überschreiben immer die Formatierungen der externen Formatierungen und die der Festlegungen im Kopfbereich der Webseite, weil diese Angaben sehr spezifisch sind (sie gelten genau für ein einziges Element).

Folgende Reihenfolge sollten Sie daher konsequent einhalten:

Externe Datei	<link rel="stylesheet" href="dateiname.css">
Lokale Definition	<style></style>
Definition am Element	<Element style="">

Beispiel: kap01\extern.css

In einer externen CSS-Datei legen Sie fest, dass die Überschrift h3 rot darzustellen ist.

```
h3 { color: red; }
```

Beispiel: kap01\rangfolge1.htm

Es wird eine externe Datei eingebunden. Danach machen Sie Angaben, die nur für die aktuelle Seite gelten. Zuletzt überschreiben Sie diese im folgenden Beispiel mit Formatangaben direkt am Element.

```
<html lang="de">
<head>
    <title>Rangfolge von Stylesheets</title>
    ①   <link rel="stylesheet" href="extern.css">
    ②   <style>
        h3 { color: green; }
    </style>
</head>
<body>
    ③   <h3>Externe vor interner Definition: Farbe ist grün</h3>
    ④   <h3 style="color:blue;">Style überschreibt die vorherigen
        Definitionen: Farbe wird blau</h3>
</body>
</html>
```

- ① Die externe CSS-Datei mit der Definition der roten Überschrift `h3` wird eingebunden.
- ② Im Kopfbereich legen Sie die lokale Definition für diese Webseite fest. In diesem Fall wäre `h3` in grüner Schriftfarbe darzustellen.
- ③ Die Ausgabe der Überschrift erfolgt in grüner Schrift.
- ④ In dieser Überschrift, die eigentlich grün sein soll, überschreiben Sie die Farbe mit dem Attribut `style`. Die Überschrift wird blau angezeigt.

Wenn Sie die externe Datei nach der lokalen Definition einbinden, also Punkt ① und ② vertauschen, wird die Überschrift ③ rot dargestellt. Die Überschrift ④ bleibt blau, da `style` für das Element `h3` zuletzt angegeben wird.

Spezifität einer Angabe erhöhen

Die Angabe von

`!important`

setzt die Spezifität einer Angabe sehr hoch. Sie legen damit fest, dass eine Formatierung nicht von nachfolgenden Angaben überschrieben werden soll. Dieses Schlüsselwort darf jedoch nicht für eine komplette Regel angegeben werden, sondern muss für jede CSS-Deklaration einzeln notiert werden, z. B.

```
h3 { color: green !important; font-size: 1em !important; }.
```

Setzen Sie `!important` bewusst und sparsam ein. Wenn Sie ein Format wie `color: green` notieren und der Browser das nicht wie gewünscht umsetzt, versuchen Sie, den Grund zu verstehen. Vielleicht haben Sie an späterer Stelle `color: black` notiert? Verwenden Sie `!important` nicht, weil es einfacher ist – Ihr Code wird schlechter verständlich und die Probleme mehren sich, wenn Sie nicht verstehen, warum der Browser nicht wie gewünscht reagiert.

CSS wurde entwickelt, um Layout und Inhalt voneinander zu trennen. Webseiten sind sehr viel einfacher zu gestalten und zu warten, wenn Sie von Anfang an konsequent alle Formate in einer zentralen Datei ablegen.

1.7 Ausgabemedien

Sie können Stylesheets für bestimmte Ausgabemedien definieren. In der Praxis werden beispielsweise oft unterschiedliche Angaben gemacht, je nachdem ob ein Dokument auf einem herkömmlichen Computer mit großem Bildschirm oder einem Smartphone angezeigt wird. Auch für den Druck werden meist Anpassungen vorgenommen.

Sinnvolle Angaben für Druckformate werden in einem gesonderten Kapitel am Ende des Buches besprochen.

Unterschiede der Stylesheets für Bildschirmausgabe und Druck

Der Sinn von verschiedenen Stylesheets ist, eine für jedes Medium optimal angepasste Darstellung bereitzustellen.

Es gibt aber noch mehr Unterschiede. Beispielsweise benötigt eine ausgedruckte Seite keine Navigation. Diese können Sie aus der Druckdarstellung entfernen. Andere Angaben, wie zum Beispiel Informationen über den Ort, an dem sich die Seite innerhalb des gesamten Internetangebotes befindet, sollten auch in der ausgedruckten Version vorhanden sein. So kann man einen ausgedruckten Artikel später leichter wiederfinden.

Mittels CSS ist es sogar möglich, die Zieladresse von Links sichtbar zu machen. Das ist für den Druck immer dann sinnvoll, wenn auf einer Webseite Links zu weiterführender Lektüre vorhanden sind. Wenn die Adressen zu den Literaturangaben ausgedruckt werden, können diese zu einem späteren Zeitpunkt eingetippt und so erreicht werden.

Da in Büros Schwarz-Weiß-Drucker noch sehr verbreitet sind, ist für die Druckdarstellung schwarze Schrift auf weißem Grund empfehlenswert. Hintergrundgrafiken und -farben oder schwache Hell-Dunkel-Kontraste sind hier nicht mehr erkennbar bzw. verbrauchen unnötig Toner bzw. Tinte und können deshalb für die Druckdarstellung entfallen. Zudem dauert der Druck von grauen Zeichen länger als von schwarzen.

Um dem Browser mitzuteilen, welche Formatvorlage für welches Ausgabemedium gedacht ist, geben Sie beim Element `<link>` zusätzlich das Attribut `media` an.

```
<link rel="stylesheet" href="dateiname.css" media="Medientyp">
```

Folgende Medientypen sind verfügbar:

<code>screen</code>	Darstellung am Bildschirm (Monitor, TFT, Smartphone usw.)
<code>print</code>	Ausgabe auf einem Drucker
<code>all</code>	Ausgabe für alle Medientypen

`all` verwenden Sie, wenn Sie möchten, dass die enthaltenen Formate sowohl für den Druck als auch für die Bildschirmdarstellung verwendet werden – da dies die Voreinstellung ist, muss diese Angabe nicht explizit gemacht werden. `screen` dient der Darstellung auf einem Bildschirm und `print` ist für die Druckausgabe.

Sämtliche anderen Medientypen, wie `speech` für die Sprachausgabe über Lautsprecher, `braille` für Ausgabegeräte für Blindenschrift oder `tv` für die Ausgabe auf einem fernsehähnlichen Bildschirm, sind veraltet und sollten laut aktuellem Standard nicht mehr benutzt werden.

Für ein HTML-Dokument können Sie mehrere Stylesheets für die unterschiedlichen Medientypen einbinden. Je nachdem, welches Ausgabegerät für den Abruf der Webseite benutzt wird, werden die jeweiligen Formatierungen umgesetzt. Soll eine Formatvorlage für mehrere Ausgabegeräte benutzt werden, geben Sie mehrere Medientypen durch Kommata voneinander getrennt an.

```
<link rel="stylesheet" href="screen.css" media="screen">
<link rel="stylesheet" href="print.css" media="print">
```

Binden Sie beispielsweise beide Anweisungen zusammen in ein HTML-Dokument ein, werden für die Bildschirmschirmdarstellung die Formatdefinitionen der CSS-Datei *screen.css* und für die Ausgabe auf einem Drucker die der Datei *print.css* verwendet.

Da heutzutage fast jeder dank Smartphone überall auf das Internet zugreifen kann, verliert der Druck von Webseiten an Bedeutung. Aus diesem Grund enthält diese Version des Buches kein eigenes Kapitel mehr über Druckstylesheets.

@media-Regel anwenden

Die Verwendung der @media-Regel gestattet den Gebrauch von Stylesheets für verschiedene Medien in einem einzigen Dokument. So werden mehrfache http-Requests vermieden. Sie geben dabei durch Kommas voneinander getrennt die Medientypen an ([kap01\@media-regel.htm](#)).

```
@media print {
    body { color: green }
}
@media screen {
    body { color: blue }
}
@media screen, print {
    h1 { color: red }
}
```

In diesem Beispiel wird festgelegt, dass für den Ausdruck eine grüne Textfarbe und für die Ausgabe am Bildschirm eine blaue Textfarbe verwendet werden soll. Die Überschrift *h1* erscheint sowohl im Ausdruck als auch am Bildschirm in roter Farbe.

Sie sind nicht auf Medientypen beschränkt, sondern können stattdessen oder zusätzlich explizit Medienmerkmale (Media Features) angeben. Diese beschreiben eine spezifische Eigenschaft des Mediums, auf dem die Ausgabe erfolgt. Dies kann beispielsweise die Breite/Mindestbreite (*width/min-width*) oder Höhe/Mindesthöhe (*height/min-height*) oder eine andere charakteristische Eigenschaft des Mediums sein. Jede Abfrage eines Medienmerkmals wird in runden Klammern platziert.

```
@media screen and (width >= 1024px) {
    h3 { font-size: 125% }
}
```

In diesem Beispiel wird für einen Viewport mit einer Breite von mindestens 1024 Pixeln eine andere Schriftgröße festgelegt.

Häufig verwendete Medienmerkmale:

Medienmerkmal	Erläuterung	Beispiel
<code>width</code>	Die Breite des Viewports bzw. einer Seite, oft in Kombination mit <code>min</code> (bzw. der moderneren Größer-gleich-Schreibweise <code>>=</code> , siehe obiges Beispiel) oder <code>max</code> (bzw. der moderneren Kleiner-gleich-Schreibweise <code><=</code>)	<code>@media (max-width: 1250px)</code>
<code>height</code>	Die Höhe des Viewports bzw. einer Seite, oft in Kombination mit <code>min</code> (bzw. <code>>=</code>) oder <code>max</code> (bzw. <code><=</code>)	<code>@media (min-height: 360px)</code>
<code>aspect-ratio</code>	Das Verhältnis des Medienmerkmals <code>width</code> zum Merkmal <code>height</code>	<code>@media (aspect-ratio: 4/3)</code>
<code>orientation</code>	Seitenformat: Hochformat (<code>portrait</code>) oder Querformat (<code>landscape</code>)	<code>@media (orientation: portrait)</code>

Eine Erläuterung aller Medienmerkmale finden Sie beispielsweise hier: https://wiki.selfhtml.org/wiki/CSS/Media_Queries/Medienmerkmale.

Formate für unterschiedliche Browsergrößen festlegen

Die `@media`-Regel hat eine zentrale Bedeutung bei der Gestaltung einer Webseite für unterschiedlich große Ausgabegeräte. In der Regel legen Sie zunächst grundlegende Formate fest, die für alle Browser gelten (vom Smartphone bis zum ultrahochauflösenden 60"-Monitor). Das sind Angaben zu Farben oder Schriftarten, die auf jedem Gerät verwendet werden sollen. Dann arbeiten Sie das Layout für größere Bildschirme weiter aus, wenn ein Browser mehr Platz bereitstellt, im oben gezeigten Beispiel mindestens eine Breite von 1024 Pixeln.

In den obigen Beispielen wird lediglich die Schriftgröße angepasst, es sind aber auch komplette Änderungen des Layouts denkbar, also zum Beispiel die Verwendung mehrerer nebeneinander liegender Spalten auf großen Bildschirmen, die dafür ausreichend Platz haben. Der Ansatz heißt **Responsive Website Design**. Dieses Design reagiert auf veränderte Größenverhältnisse, um den Nutzern auf allen Geräten eine optimale Darstellung zu bieten.

Hierzu lässt sich die Eigenschaft `(min-)width`, also die tatsächlich nutzbare Größe des Browserfensters, abfragen.

Progressive enhancement und mobile first

Grundsätzlich sollten Sie sich angewöhnen, sich zunächst um die Darstellung auf Geräten mit wenigen Fähigkeiten, alten Browsern und kleinen Bildschirmen zu kümmern. Für Geräte und Software mit mehr Fähigkeiten fügen Sie Verbesserungen schrittweise hinzu, **nachdem Sie durch ausgiebige Tests sichergestellt haben, dass für die weniger potenteren Geräte die Webseite vollständig bedienbar und lesbar ist**.

Das Prinzip, Grundfunktionalitäten für alle und verbesserten Komfort für Hardware und Software bereit zu stellen, die diese nutzen kann, nennt man **progressive enhancement** (fortschreitende Verbesserung).

Daher kümmern Sie sich auch zunächst ausschließlich um die Darstellung auf kleinen Bildschirmen, wie zum Beispiel Smartphones – erst in den nächsten Schritten erweitern Sie Ihr Layout um mehrere Spalten und komplexere Anordnungen. Diese Vorgehensweise beschreibt man mit dem Schlagwort **mobile first** (mobile Geräte zuerst).

Smartphones sollten in zweierlei Hinsicht an erster Stelle stehen: erstens bei Ihrer Arbeitsweise wie oben beschrieben, zweitens in Ihrer Aufmerksamkeit. Inzwischen sind Smartphones die beliebtesten Geräte zur Nutzung des Webs. Daher sollte Ihre Webseite auf den kleinen Geräten auch besonders gut funktionieren.

Wenn Sie progressive enhancement und mobile first als Grundlage für Ihren Arbeitsablauf etablieren, profitieren Sie doppelt: Ihre Arbeit wird erleichtert, denn es ist mühsam und fehlerträchtig, ein Layout, das nur für große Bildschirme und modernste Browser konzipiert ist, nachträglich aufgrund von Beschwerden für ältere und kleinere Geräte zugänglich zu machen.

Außerdem sind die Nutzer zufriedener und belohnen Sie durch wiederkehrende Besuche. Zu schwer zu reparierenden Beschwerden kommt es entsprechend selten.

1.8 Alternative Stylesheets

Die CSS-Spezifikation sieht eine Möglichkeit vor, den Benutzern mehrere unterschiedliche Stylesheets zur Auswahl anzubieten. Der Besucher kann dann das Stylesheet auswählen, mit dem er die Webseite betrachten möchte.

Die meisten Nutzer kennen diese Funktion nicht und wissen nicht, wie man alternative Stylesheets aktiviert, und die Browserunterstützung für den Wechsel der Stylesheets durch die Nutzer ist eher mangelhaft. Das bedeutet: Es lohnt sich kaum, ein alternatives Stylesheet anzubieten, wenn Sie es nicht auch auf anderem Weg zugänglich machen. Unter https://wiki.selfhtml.org/wiki/JavaScript_und_CSS/Stylesheets_dynamisch_wechseln finden Sie hierzu eine ausführliche Anleitung.

Zur Einbindung externer Stylesheets in eine HTML-Datei ist folgende Zeile nötig:

```
<link rel="alternate stylesheet" href="alternate-screen.css"  
title="Auswahl1">
```

Das Wichtigste an dieser Angabe ist der Wert `alternate stylesheet`, mit dem Sie das alternative Stylesheet kenntlich machen. Bei der Angabe von mehreren Formatvorgaben ermöglicht das Attribut `title` dem Anwender, das entsprechende Format aus einer generierten Liste (meistens über einen Menüpunkt) auszuwählen.

Das Stylesheet, das standardmäßig beim Laden der Webseite genutzt werden soll, binden Sie über die Standardeinbindung ein. Dies ist das bevorzugte (preferred) Stylesheet. Um dem Nutzer mitzuteilen, welches Stylesheet standardmäßig verwendet wird, geben Sie über das Attribut `title` eine aussagekräftige Bezeichnung wie z. B. `Standard` an.

```
<link rel="stylesheet" href="screen.css" title="Standard">
```

Beispiel: *kap01\alternativ.htm*

Erstellen Sie eine Webseite, bei der Sie eine externe CSS-Datei mit Formatierungen einbinden und zusätzlich zwei alternative Formatierungen ermöglichen.

```
<html lang="de">
<head>
    <title>Alternative Stylesheets</title>
    ①   <link rel="stylesheet" href="style1.css" title="Standard">
    ②   <link rel="alternate stylesheet" href="style2.css"
        title="Serifenschrift - grüner Text">
    ③   <link rel="alternate stylesheet" href="style3.css"
        title="Serifenlose Schrift - blauer Text">
</head>
<body>
    ④   <h2>Alternative Stylesheets</h2>
        <p>Dieser Text wird in verschiedenen Schriftarten und -farben
        angezeigt, wenn Sie im Browser auf die alternativen Stylesheets
        umschalten.</p>
</body>
</html>
```

- ① Standardmäßig wird die CSS-Datei *style1.css* zur Formatierung der Elemente genutzt. Damit dies auch im Browser ersichtlich wird, vergeben Sie den Titel Standard.
- ② Die erste alternative Formatierung verbirgt sich in der Datei *style2.css*. In der Auswahl wird diese über die Bezeichnung Serifenschrift – grüner Text angezeigt.
- ③ Die zweite Alternative versteckt sich hinter der Bezeichnung Serifenlose Schrift – blauer Text. Wird diese im Browser ausgewählt, werden die Elemente nach den Festlegungen der CSS-Datei *style3.css* formatiert.
- ④ Die HTML-Elemente werden eingefügt, um das Umschalten der Formate sichtbar zu machen.

Damit die einzelnen Formatierungen eingebunden und über ein Menü gewechselt werden können, erstellen Sie zusätzlich die drei verschiedenen externen Stylesheet-Dateien *style1.css*, *style2.css* und *style3.css*.

style1.css

```
body { font-family: Georgia, Verdana, Arial, Helvetica, sans-serif; }
h2 { color: red }
p { color: black }
```

Der gesamte Inhalt der Webseite wird in der Schriftart Georgia oder in einer der anderen Schriftarten dargestellt. Überschriften der zweiten Ordnung *h2* werden rot und Absätze *p* werden schwarz eingefärbt.

style2.css

```
body { font-family: "Times New Roman", Times, sans; }
h2 { color: black }
p { color: green }
```

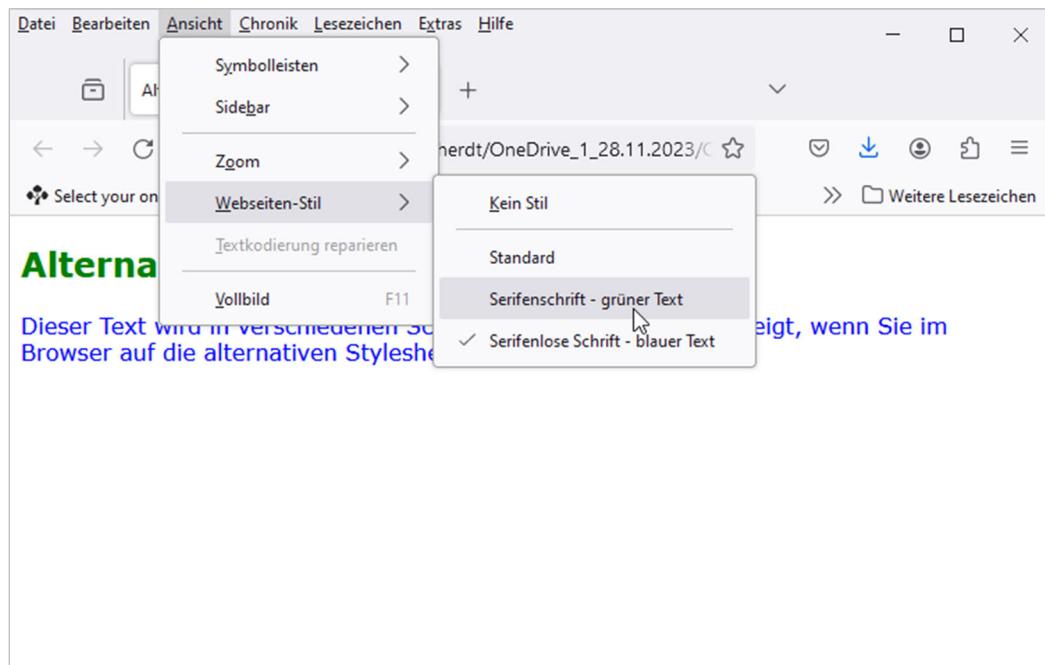
Die Webseite wird in der Schriftart Times New Roman oder in einer der anderen Schriftarten dargestellt. Überschriften `h2` werden schwarz und Absätze `p` grün eingefärbt.

style3.css

```
body { font-family: Verdana, Arial, helvetica, sans-serif; }
h2 { color: green }
p { color: blue }
```

Neben der Schriftart Verdana als Hauptschriftart werden die Überschriften `h2` grün und Inhalte der Absätze `p` blau dargestellt.

Nach dem Laden der Webseite `kap01\alternativ.htm` in Firefox wählen Sie *Ansicht - Webseiten-Stil*, um das gewünschte Stylesheet auszuwählen. (Falls die Menüleiste nicht angezeigt wird, können Sie diese mit der Taste `Alt` einblenden lassen.)



Auswahl in Mozilla Firefox

1.9 Übungen

Übung 1: Theoriefragen zu Stylesheets

Level		Zeit	ca. 30 min
Ergebnisdatei	<i>kap01\uebung1-7.doc</i>		

1. Erklären Sie, was Cascading Style Sheets sind.
2. Geben Sie an, welche Möglichkeiten zur Einbindung von Stylesheets bestehen.
3. Nennen Sie die Vor- und Nachteile.
4. Erklären Sie die Rangfolge, mit der die Stylesheets angewendet werden.
5. Wie können Sie die Rangfolge der Stylesheets ändern?
6. Mit welchen Medientypen legen Sie zum einen die Formatierungen für den Computermonitor und zum anderen für den Ausdruck fest?
7. Erklären Sie, mit welchen Festlegungen Sie dem Besucher Ihrer Webseite ermöglichen, die Formatierung der Webseite zu wechseln.

Übung 2: Externe Stylesheets einbinden

Level		Zeit	ca. 7 min
Übungsdatei	<i>stylesheet.htm</i>		
Ergebnisdateien	<i>kap01\uebung8.htm, kap01\uebung8.css</i>		

1. Übertragen Sie die Formate in der Beispieldatei *stylesheet.htm* in eine externe Datei.
2. Binden Sie diese Datei in das Dokument ein.

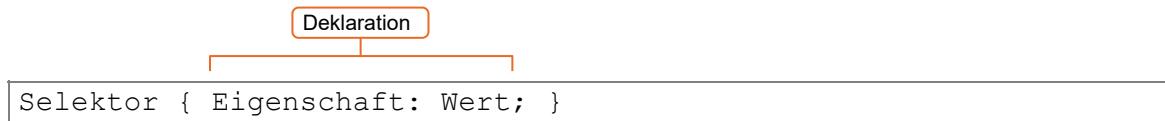
2

CSS-Syntax und Selektoren

2.1 Aufbau von Stylesheets

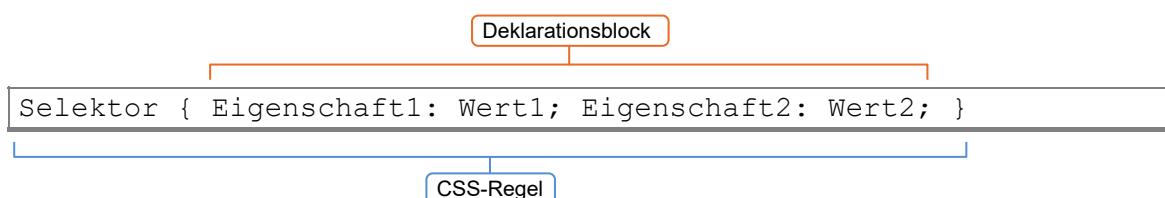
Ein Stylesheet besteht aus Regeln (engl.: style rules oder rules). Eine Regel wiederum besteht aus dem **Selektor** (engl.: selector), mit dem ausgewählt wird, **was** formatiert werden soll, und **Deklarationen** (engl.: declarations), die beschreiben, **wie** formatiert werden soll. Diese Deklarationen werden in geschweiften Klammern notiert.

Jede Deklaration besteht wiederum aus einer **Eigenschaft** (engl.: property), gefolgt von einem Doppelpunkt, gefolgt von einem **Wert** (engl.: property value).



Wenn Sie mehrere Eigenschaften angeben möchten (zum Beispiel Schriftfarbe und Hintergrundfarbe), werden die Deklarationen durch ein **Semikolon** (einen Strichpunkt) voneinander getrennt.

Wie bei HTML benötigen die Browser auch in CSS-Dateien keine Leerzeichen oder Zeilenumbrüche. Diese fügen Menschen ein, um die Formate besser lesen zu können. **Browser sind auf das Semikolon angewiesen**. Damit werden mehrere Deklarationen für den Browser voneinander getrennt. Gewöhnen Sie sich an, auch das für Browser nicht nötige Semikolon hinter der letzten Deklaration stets zu setzen. Man vergisst es sonst leicht, wenn eine neue Deklaration hinzugefügt wird, und die Suche nach einem vergessenen Semikolon kann sehr lange dauern.



Die Gesamtheit aller Deklarationen innerhalb der geschweiften Klammern wird **Deklarationsblock** (engl.: declaration block) genannt.

Beispiel

Mittels eines Selektors wählen Sie Elemente zur Formatierung aus. Wenn Sie allen Bildern Formate zuweisen möchten, notieren Sie als Selektor `img`. Im folgenden Beispiel sollen Höhe und Breite der Bilder festgelegt werden.

```
img {width: 5em; height: 4em; }
```

2.2 Wie Browser Ihre Webseite untergliedern

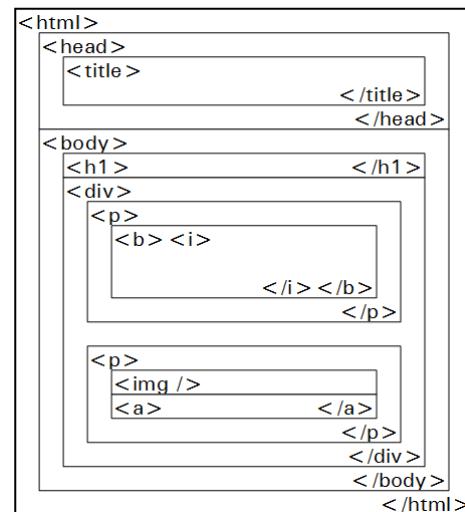
Für den richtigen Einsatz von Selektoren müssen Sie verstehen, wie Browser Ihre Webseiten wahrnehmen. HTML ist streng hierarchisch aufgebaut. Für Browser gibt es innere und äußere Elemente. **Die äußeren Elemente sind die Vorfahren der inneren Elemente. Dementsprechend sind die inneren Elemente die Nachfahren der äußeren.**

Bei direkten Nachfahren spricht man von Kind-elementen. `body` ist ein Kindelement von `html`.

Bei direkten Vorfahren spricht man von Eltern-elementen. `html` ist ein Elternelement von `body`.

Weitere Beispiele:

- ✓ `div` ist ein Vorfahre von `p`, `b` und `i` sowie von `p`, `img` und `a`.
- ✓ `h1` und `div` sind Kinder von `body`.
- ✓ `h1`, `div`, `p`, `b`, `i`, `img` und `a` sind Nachfahren von `body`.
- ✓ `div` ist ein Nachbar von `h1` und `img` ist ein Nachbar von `a`.
- ✓ `b` und `i` sind Nachfahren vom ersten `p`.
- ✓ `b` ist ein Kind vom ersten `p`.
- ✓ `i` ist ein Kind von `b`.



2.3 Vererbung

Vererbung (engl.: inheritance) ist ein wichtiger Bestandteil des CSS-Konzeptes. Vererbung bedeutet, dass Formate von einem HTML-Element an seine Unterelemente weitergegeben werden. Anschaulich wird dieses System anhand des Elemente-Baums einer Seite.

Der sichtbare Teil einer Webseite befindet sich zwischen den Tags <body> und </body>. Definieren Sie z. B. für den body eine schwarze Schriftfarbe. Jetzt werden alle textlichen Elemente der Webseite schwarz dargestellt. Legen Sie dann für die geordnete Liste ol eine rote Schrift fest, werden auch die Unterelemente li rot dargestellt. Sie haben die Eigenschaft von dem übergeordneten Element geerbt.

```
<html>
  <head>
    <title></title>
  </head>
  <body> (schwarz)
    <h1>(schwarz)</h1>
    <p>(schwarz)
      <b>(schwarz)</b>
    </p>
    <ol> (rot)
      <li>(rot)</li>
      <li>(rot)</li>
    </ol>
    <p>(schwarz)</p>
  </body>
</html>
```

Einige Eigenschaften werden vererbt. Das gilt beispielsweise für Schriften und Schriftfarben.

So können Sie mit der folgenden Regel allen Elementen eine rote Schriftfarbe zuweisen:

```
body { color: red; }
```

Andere Eigenschaften werden nicht vererbt. Wenn Sie beispielsweise dem Rumpf (engl.: body) Ihrer Seite einen Rahmen mitgeben, bedeutet das nicht, dass alle enthaltenen Elemente ebenfalls umrahmt werden (stellen Sie sich mal vor, wie das aussähe).

CSS wurde so entwickelt, dass Eigenschaften vererbt werden, wenn es Sinn macht. Für Sie bedeutet das, dass in der Regel genau das passiert, was Sie erwarten und möchten (z. B., dass alle Schriften in body rot eingefärbt werden können, Rahmen aber für jedes Element einzeln vergeben werden müssen).

Vererbung beeinflussen

Das Verhalten von CSS bezüglich der zu vererbenden Eigenschaften können Sie ändern. Wenn Sie beispielsweise möchten, dass eine Überschrift erster Ordnung (h1) in Ihrem body denselben Rahmen (engl.: border) wie sein Elternelement bekommt, können Sie diese Eigenschaft ausdrücklich übernehmen. Dazu stellt CSS das Schlüsselwort `inherit` bereit.

```
body {
  border: 3px solid red;
}
h1 {
  border: inherit;
```

Überschrift einer Anleitung

Kapitel-Überschrift

Überschrift h1 erhält denselben Rahmen wie das Elternelement.

Nihil quisquam fugit ab accusamus ratione ipsum et! Blanditiis consequuntur enim quod voluptate numquam in illum, placeat. Porro aliquam iure sunt vitae, perferendis facere sint accusamus est unde amet sapiente!

<kap02\inherit-schluesselwort.htm>

Durch geschickte Nutzung der Vererbung können Sie mit recht wenigen Formatierungen das Aussehen Ihrer Website effizient verändern.

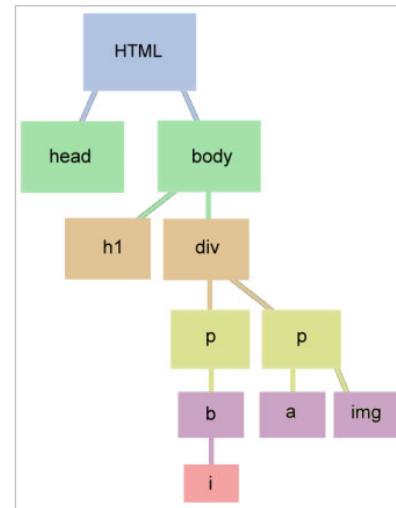
Vererbung von den Vorfahren zu den Nachfahren

Die nebenstehende Baumansicht eines HTML-Dokumentes erleichtert das Verständnis von Vererbung. Es ist dasselbe Dokument, das auch in der schematischen Darstellung innerer und äußerer Elemente weiter oben in diesem Kapitel dargestellt wurde.

Oben stehende Elemente sind die Vorfahren, die ihre Eigenschaften an ihre unten stehenden Nachfahren weitergeben.

Dabei werden nicht alle Eigenschaften an die Nachfahren weitergegeben. Vererbt werden zum Beispiel Schrifteigenschaften wie Farbe oder Schriftart.

Haben mehrere Elemente dasselbe Elternelement (in der Abbildung sind dies zum Beispiel `a` und `img` als Kinder von `p`), dann spricht man von **Geschwistern**.



Die Elemente `b` und `i` dienen dazu, Text auszuzeichnen, der nicht wichtiger ist als umgebender Text, der aber dennoch hervorgehoben werden soll. In der Regel wird mit `i` auf Fachbegriffe hingewiesen. Botanische Bezeichnungen können so ausgezeichnet werden, zum Beispiel: "Das Maiglöckchen (<i>Convallaria majalis</i>) ist eine Pflanze [...]".

Mehr Informationen finden Sie unter <https://html.spec.whatwg.org/#the-i-element> (engl.).

2.4 Universal-Selektor

```
* { Eigenschaft: Wert; }
```

Mit dem Sternchen * wählen Sie alle Elemente aus, die sich in Ihrem Dokument befinden. Wenn Sie das folgende Beispiel notieren, erhalten alle Elemente einen roten Hintergrund.

```
* { background-color: #f00; }
```

Sinnvoll verwenden lässt sich der Universal-Selektor vor allem in Kombination mit einem der folgenden Selektoren.

Wenn Selektoren miteinander kombiniert werden, spricht man von **Kombinatoren**.

2.5 Element-Selektor

Die einfachste Art, ein Format zuzuweisen, besteht darin, den Namen eines Elementes als Selektor zu notieren. Beispielsweise werden mit dem Selektor `p` alle Absätze im HTML-Dokument angesprochen und formatiert.

```
p { width: 75%; }
```

Damit legen Sie fest, dass alle Absätze 75 % der Breite ihres Elternelementes erhalten sollen. Möchten Sie das Format gleichzeitig mehreren HTML-Tags zuweisen, geben Sie alle entsprechenden Elemente an und trennen Sie diese durch Kommas voneinander. Die folgenden Beispiele führen zu demselben Ergebnis:

Beispiel 1

```
h1 { width: 75%; }
h2 { width: 85%; }
h3 { width: 85%; }
p { width: 85%; }
```

Beispiel 2

```
h1 { width: 75%; }
h2, h3, p { width: 85%; }
```

Auf diese Weise können Sie alle Elemente Ihrer Seiten formatieren. Möchten Sie Links in der Navigation anders präsentieren als im Fließtext oder soll der erste Absatz auf jeder Seite hervorgehoben werden, benötigen Sie feinere Möglichkeiten, Elemente auszuwählen. Dazu mehr in den folgenden Abschnitten.

2.6 Nachfahren-Selektor

```
Selektor1 Selektor2 { Eigenschaft: Wert; }
```

Mit dem Nachfahren-Selektor haben Sie die Möglichkeit, Elemente abhängig von ihren Vorfahren auszuwählen und zu formatieren. Wenn Sie zum Beispiel möchten, dass Texte, die mit `` ausgezeichnet sind, nur dann rot dargestellt werden, wenn diese sich in Absätzen befinden, sollten Sie den Nachfahren-Selektor verwenden. Da der Nachfahren-Selektor nur durch die Kombination von zwei Elementen möglich wird, gehört er zu den **Kombinatoren**.

Beispiel: *kap02\selektor-nachfahren.htm*

Ein Text wird mittels HTML-Tags strukturiert. Die Elemente `b` und `i` werden im Folgenden formatiert, und zwar abhängig davon, in welchem Element sie sich befinden.

```

<html lang="de">
<head>
    <title>Nachfahren-Selektor</title>
    ① <style type="text/css">
        body { color: #000; }
        ② p b { color: #a00; }
        ③ p i { color: #0a0; }
        ④ span { text-decoration: underline; }
    </style>
</head>
<body>
    <h1>Die Nachfahren</h1>
    <p>Es folgt ein Text, der die Formatierungen verdeutlichen soll.</p>
    ⑤ <p><span><b>Dieser fett formatierte Text wird rot eingefärbt</b></span>
    ⑥ <i>und dieser kursive Text wird automatisch grün dargestellt</i>.
    ⑦ Dieser Text ist wieder schwarz.</p>
</body>
</html>

```

- ① Mit `<style>` wird das Styleelement geöffnet.
- ② Wird in einem Absatz das Element `b` verwendet, ist dieser Text in roter Farbe darzustellen.
- ③ Befindet sich `i` in einem Absatz, soll der Text grün dargestellt werden.
- ④ Text im Element `span` soll unterstrichen werden.
- ⑤ Text wird von den Tags `` und `` umschlossen. Einer der Vorfahren von `b` ist ein Absatz, deshalb erhält `b` eine rote Schrift.
- ⑥ Text wird von den Tags `<p>` und `<i>` umschlossen und wird grün hervorgehoben.
- ⑦ Die letzte Zeile wird nur von `<p>` umschlossen. Damit wird dieser Teil des Textes wie der Rest der Webseite dargestellt.

Die Nachfahren

Es folgt ein Text, der die Formatierungen verdeutlichen soll.

Dieser fett formatierte Text wird rot eingefärbt und dieser kursive Text wird automatisch grün dargestellt. Dieser Text ist wieder schwarz.

kap02\selektor-nachfahren.htm

Sie können beliebig viele Elemente als Vorfahren im Selektor angeben. Die Formatierung wird erst dann angewendet, wenn alle Elemente in der angegebenen Reihenfolge aufgetreten sind:

Beispiel

```
ul p a { color: #f00; }
```

Hier wird Text nur rot, wenn es sich um einen Link `a` in einem Absatz `p` handelt, der sich wiederum in einer unsortierten Liste `ul` befindet.

Beachten Sie: in HTML dürfen Listen nur Listeneinträge `li` beinhalten. Die Listeneinträge selbst dürfen auch andere Elemente beinhalten. `p` kann also niemals ein Kind von `ul` sein – es ist aber sein Nachfahre. Entsprechend muss auch der Link kein Kind von `p` sein. Die Formate werden auch angewendet, wenn der Link nochmals in anderen Elementen wie `span` oder `em` verschachtelt ist.

2.7 Kind-Selektor

```
Selektor1 > Selektor2 { Eigenschaft: Wert; }
```

Der Kind-Selektor ist ein enger Verwandter des Nachfahren-Selektors. Der Unterschied besteht darin, dass `Selektor2` ein **direkter** Nachfolger von `Selektor1` sein muss.

Beispiel

```
p > b { background-color: #eee; }
```

Ist `b` ein Kind von `p`, wird der Hintergrund farbig unterlegt.

```
<!-Das Wort "fett" wird farbig unterlegt, weil b ein Kind von p  
ist. -->  
<p>Der Text ist <b>fett</b> formatiert.</p>  
  
<!-Hier nicht, weil b zwar ein Nachfahre, aber kein Kind von p  
ist. span ist ein Kind von p, b ist ein Kind von span, beide sind  
Nachfahren von p -->  
<p>Der <span>Text ist <b>fett</b> formatiert.</span></p>
```

Beispiel: `kap02\selektor-kind.htm`

Sie wählen `b` per Kind-Selektor aus. Die zwei Absätze machen dabei die Funktionsweise des Selektors deutlich.

```
① <html lang="de">  
  <head>  
    <title>Kind-Selektor</title>  
    <style>  
      p > b { color: #a00; }  
      i { text-decoration: underline; }  
    </style>  
  </head>  
  <body>  
    <h1>Der Kind-Selektor</h1>
```

```

② <p>Auch dieser <b><i>unterstrichene Teil</i> wird rot  

    gefärbt</b>, da das Element &lt;b&ampgt ein direktes Kind von  

    &lt;p&ampgt ist.  

</p>  

③ <p><i><b>Diese Formatierung hingegen ruft keine Färbung  

    hervor</b>. Der Tag &lt;b&ampgt ist kein direktes Kind von  

    &lt;p&ampgt.</i></p>  

</body>  

</html>

```

- ① Mit **** und **** umschlossener Text wird rot dargestellt, wenn das Tag sich direkt in einem Absatz befindet.
- ② An dieser Stelle ist **b** ein Kind von **p**. Der von **** umschlossene Text wird rot dargestellt.
- ③ Hier ist **b** ein Kind von **i**, nicht von **p**. Die Bedingung im Selektor trifft also nicht zu, sodass dieses Element seine schwarze Schriftfarbe behält.

Der Kind-Selektor

Auch dieser unterstrichene Teil wird rot gefärbt, da das Element **** ein direktes Kind von **<p>** ist.

Diese Formatierung hingegen ruft keine Färbung hervor. Der Tag ist kein direktes Kind von <p>.

kap02\selektor-kind.htm

2.8 Selektor für Geschwisterelemente

```
Selektor1~Selektor2 { Eigenschaft: Wert; }
```

Dieser Selektor wählt Selektor2, wenn Selektor1 und Selektor2 dasselbe Eltern-element besitzen, wenn die beiden Elemente also Geschwister sind **und** Selektor2 im HTML-Dokument **nach** Selektor1 notiert wird.

Browser lesen HTML-Dokumente nur in einer Richtung (vom Anfang zum Ende) – **ein Zugriff auf vorherige oder übergeordnete Elemente ist grundsätzlich nicht möglich**.

Aus diesem Grund gibt es auch keine Eltern- oder Vorfahrenelektoren.

2.9 Selektor für benachbarte Elemente

```
Selektor1+Selektor2 { Eigenschaft: Wert; }
```

Dieser Selektor funktioniert ähnlich wie der Geschwister-Selektor, verhält sich aber strenger. Hier wird Selektor2 nur formatiert, wenn das entsprechende Element direkt auf Selektor1 im HTML-Dokument folgt.

Beispiel: *kap02\selektor-nachbar.htm*

Ihre Webseite beinhaltet eine Reihe von Absätzen. Den ersten Absatz nach einer Überschrift heben Sie besonders hervor. Außerdem möchten Sie jedes **b**, das auf ein *i* folgt, rot färben.

```

① <html lang="de">
  <head>
    <title>Selektor für benachbarte Elemente</title>
    <style>
      h1+p { background-color: #ccc; }
      i+b { color: #f00; }
    </style>
  </head>
  <body>
    <h1>Selektor für benachbarte Elemente</h1>
    <p>Fortune plango vulnera stulantibus ocellis, ...
    <p>Omnia Sol <i>temperat</i> purus et <b>subtilis</b>, ...
    <p>Qui autem auscultare nolet, exsurgat foras, ...
  </body>
</html>

```

- ① Mit dem Selektor **h1+p** legen Sie fest, dass das zugehörige Format auf den Absatz angewendet werden soll, der direkt nach der Überschrift **h1** folgt. Außerdem soll ein **b** nach einem *i* rot hervorgehoben werden.
- ② Dieser Absatz folgt einer **h1** und erhält deshalb die festgelegte Hintergrundfarbe.
- ③ Das Wort **subtilis** wird rot hervorgehoben, da **b** einen Nachbarn *i* besitzt.

Selektor für benachbarte Elemente

Fortune plango vulnera stulantibus ocellis, quod sua michi munera subtrahit rebellis.
Zephyrus nectareo spirans in odore, certiam pro bravia curramus in amore.
Omnia Sol *temperat* purus et **subtilis**, novo mundo reserat faciem Aprilis, ad Amorem
prosperat animus herilis et iocundis imperat deus puerilis.
Qui autem auscultare nolet, exsurgat foras, ut sit, ubi sedat ile qui auscultare vult. Id nos
Latine gloriosum dicimus.

Formatierung des ersten Absatzes nach der Überschrift (kap02\selektor-nachbar.htm)

2.10 Klassen-Selektor

```
.klassename { Eigenschaft: Wert; }
```

Klassen sind Elemente mit Gemeinsamkeiten. Genau wie eine Klasse von Autos hat auch eine Klasse von Elementen gleiche Eigenschaften. Doch genauso wenig wie alle Mittelklasse-Autos identisch sind, sind alle Elemente, die zu einer Klasse gehören, identisch.

Im Abschnitt über den Element-Selektor wurde bereits darauf hingewiesen, dass es nötig sein kann, nicht pauschal jedes Auftauchen eines Elementes zu gestalten. Ein typisches Beispiel hierfür sind Links. Oft möchte man interne Links anders darstellen als externe Links, die von der Webseite wegführen. Es macht also Sinn, alle externen Links zu markieren, um diese gesondert gestalten zu können. Um ein Element einer bestimmten Klasse zuzuordnen, müssen Sie ihm im HTML-Dokument das Attribut **class** mit einem entsprechenden Namen mitgeben. Den Namen können Sie selber festlegen.

Bei den Klassennamen müssen Sie folgende Namenskonventionen beachten:

- ✓ Erlaubt sind die Zeichen A–Z, a–z, 0–9 sowie Mittestrich – und Unterstrich.
- ✓ Zusätzlich sind die Zeichen ab dem ASCII-Wert 161 erlaubt, also z. B. die deutschen Umlaute ä, ö, ü und auch die Zeichen £, ♂, §, ©, ®.
- ✓ Die Namen dürfen jedoch nicht mit einem Mittestrich – beginnen.
- ✓ Leerzeichen sind nicht erlaubt.
- ✓ Groß- und Kleinschreibung müssen im HTML und im CSS identisch notiert werden.

Im folgenden Beispiel erhält ein externer Link eine Klasse mit dem Namen `extern`:

Klasse erstellen und nutzen

```
<a class="extern" href="http://www.example.com">Link zu  
externeSite.de</a>
```

Nachdem Sie für alle Links, die auf eine externe Site verweisen, in Ihrem HTML-Dokument die Klasse `extern` vergeben haben, können Sie mit dem Klassen-Selektor gezielt alle Links auswählen, die von Ihrer Website wegführen.

Der Klassen-Selektor wird mit einem Punkt `.` gekennzeichnet, gefolgt vom Namen der Klasse.

```
.extern { color: #0a0; }
```

So wählen Sie alle Elemente mit der Klasse `extern` aus – egal ob es sich dabei um Überschriften, Absätze oder Links handelt.

Sie können dem Klassen-Selektor noch eine weitere Bedingung hinzufügen. Es ist möglich, nur bestimmte Elemente auszuwählen, denen die Klasse mitgegeben wurde, also beispielsweise nur Links. Dazu geben Sie den Elementnamen vor dem Punkt an:

```
Elementname.Klassename { Eigenschaft: Wert; }
```

So wählen Sie nur Links aus, denen die Klasse `extern` mitgegeben wurde:

```
a.extern { color: #0a0; }
```

Natürlich dürfen Sie auch weitere Elemente mit dieser Klasse formatieren:

```
a.extern { color: #0a0; }  
span.extern { color: #0aa; }
```

Beispiel: *kap02\selektor-class.htm*

Sie erstellen drei verschiedene Klassen. Zwei Formatzuweisungen sollen nur angewendet werden, wenn die entsprechende Klasse Absätzen `p` zugeordnet ist. Die Klasse `klein` wird ohne Angabe eines Tags selektiert. Dadurch wird die Schriftgröße von jedem Element mit dieser Klasse auf 75 %¹ verkleinert.

```

<html lang="de">
<head>
    <title>Klassen von Stylesheets</title>
    <style>
        ① body           { color: black; }
        ② h1            { font-size: 125%; background-color: #eee; }
        ③ p.hintergrund { background-color: #ddd; }
        ④ p.rot         { color: #800; }
        ⑤ .klein        { font-size: 75%; }

    </style>
</head>
<body>
    ⑥ <h1>Stylesheets: Verwenden von Klassen</h1>
    ⑦ <p class="hintergrund">Absatz &lt;p&gt; mit Klasse
        .hintergrund</p>
    ⑧ <p class="rot">Absatz &lt;p&gt; mit Klasse .rot</p>
    ⑨ <p class="klein">Absatz &lt;p&gt; mit Klasse .klein</p>
    ⑩ <h1 class="klein">Überschrift h1 und zusätzlich Klasse
        .klein</h1>
</body>
</html>
```

- ① Alle Texte sollen schwarz sein.
- ② Die Überschrift `h1` wird 125% groß und erhält einen hellgrau eingefärbten Hintergrund.
- ③ Wird einem `p` die Klasse `hintergrund` zugewiesen, wird der Absatz hellgrau unterlegt.
- ④ Der Text eines Absatzes mit der Klasse `rot` wird in roter Schrift angezeigt.
- ⑤ Die Klasse `klein`, für welche die Schriftgröße auf 75 % festgelegt wird, kann bei jedem HTML-Element verwendet werden.
- ⑥ Diese Überschrift wird über den Selektor `h1` angesprochen.

Stylesheets: Verwenden von Klassen

Absatz `<p>` mit Klasse `.hintergrund`

Absatz `<p>` mit Klasse `.rot`

Absatz `<p>` mit Klasse `.klein`

Überschrift `h1` und zusätzlich Klasse `.klein`

Verschieden formatierte Absätze und
Überschriften (*kap02\selektor-class.htm*)

¹ Prozentangaben beziehen sich auf die Schriftgröße des Elterelementes. Kommt im Eltern-element eine Schriftgröße von 16 Pixeln zum Einsatz, hätte die Schrift im Kindelement mit einer Angabe von 75 % also eine Größe von 12 Pixeln.

- ⑦ Der Absatz wird hellgrau unterlegt, da die Bedingung "Absatz mit der Klasse `hintergrund`" zutrifft.
- ⑧ Dank `class="rot"` innerhalb des Tags `<p>` greift der Selektor `p.rot` – der Text wird rot.
- ⑨ Der Inhalt des Absatzes wird durch Zuweisen der Klasse `klein` formatiert.
- ⑩ Die Überschrift `<h1>` wird automatisch über den Selektor `h1` formatiert. Da sie jedoch zusätzlich die Angabe der Klasse `klein` besitzt, bleibt zwar die Hintergrundfarbe erhalten, die Schriftgröße wird jedoch auf 75 % der normalen Schriftgröße verkleinert.

Einem Element mehrere Klassen zuweisen

```
<style>
  body { color: #000; }
  li.rot { color: #c00; }
  li.grün { color: #0c0; }
  li.groß { font-size: 120%; }
  li.klein { font-size: 80%; }
</style>
```

Sie können einem Element auch mehr als nur eine Klasse zuweisen. Dazu geben Sie als Attributwert von `class` die anzuwendenden Klassen an. Jede Klasse ist durch ein Leerzeichen von den anderen zu trennen:

```
<ul>
  <li class="rot groß">rot und groß</li>
  <li class="grün groß">grün und groß</li>
  <li class="rot klein">rot und klein</li>
  <li class="grün klein">grün und klein</li>
</ul>
```

! Damit dieses Beispiel leicht nachzuvollziehen ist, wurden den Klassen Namen gegeben, die den geänderten Eigenschaften entsprechen, also `rot` für rote Schrift und `groß` für große Schrift. In der Praxis sollten die Bezeichnungen keinen Bezug zum Design haben, damit das Design geändert werden kann, ohne dass Änderungen am HTML-Dokument vorgenommen werden müssen. Es empfehlen sich also logische Bezeichnungen wie `wichtig` oder `fussnote` statt Namen von Farben oder Schriften.

2.11 ID-Selektor

```
#ID { Eigenschaft: Wert; }
Element#ID { Eigenschaft: Wert; }
```

Der ID-Selektor ist dem Klassen-Selektor sehr ähnlich. Im HTML-Dokument haben Sie die Möglichkeit, einem Element über das Attribut `id` eine eindeutige Kennung zuzuweisen. Der Unterschied zur Klasse: Eine ID darf innerhalb eines Dokumentes nur genau einmal vorkommen. Eine Klasse können Sie mehrfach auf einer Seite einsetzen.

Theoretisch könnten Sie für die **Formatierung** komplett auf Ids verzichten. Klassen würden völlig ausreichen. Allerdings hat die Eindeutigkeit von Ids auch Vorteile. Sie eignen sich als Ziel von Zuordnungen und Links und wahrscheinlich sind bestimmte Ids in Ihrem Dokument gerade deswegen bereits vorhanden.

Da Ids nur einmal pro HTML-Dokument vorkommen, ist die Verwendung einer ID als Selektor spezifischer als die Verwendung einer Klasse. Wenn Sie einem Element eine Klasse und eine ID mitgeben, können Sie mittels ID-Selektor Eigenschaften überschreiben, die in der Klasse festgelegt sind.

Beispiel

Im folgenden Beispiel hat ein Link, der auf eine bestimmte Seite innerhalb Ihres Webauftrittes verweisen soll (zur Startseite) die Klasse `intern` und die ID `zurStartseite`.

```
<a class="intern" id="zurStartseite" href="index.htm">
```

Im Stylesheet haben Sie für alle internen Links festgelegt, dass diese rot dargestellt werden sollen. Der Link mit der ID `zurStartseite` soll grün sein. Obwohl die ID im folgenden Beispiel vor der Klasse `intern` definiert wird, funktioniert das, denn eine ID hat eine höhere Spezifität als eine Klasse und setzt sich in der Darstellung durch.

```
#zurStartseite { color: green; }
.intern { color: red; }
```



Die hohe Spezifität von Ids macht es schwer, Formate, die in einer Regel mit einem ID-Selektor erstellt werden, an anderer Stelle wieder zu überschreiben. Daher sollten Sie den ID-Selektor möglichst vermeiden.

Beispiel: `kap02\selektor-id.htm`

Erstellen Sie ein HTML-Dokument mit diesen Ids:

```
<html lang="de">
<head>
<title>ID-Selektoren</title>
<style>
①   #bild1 { position: absolute; left: 200px; top: 75px; }
②   #element13 { font-size: 150%; }
③   ul#main_navigation { color: #800; background-color: #eee; }
</style>
</head>
<body>
<h3>ID-Selektoren</h3>
```

```

④ <ul id="main_navigation">
    <li>Aktuelle Seite</li>
    <li><a href="link.htm">Link zu einer Seite</a></li>
    <li><a href="link.htm">Link zu einer Seite</a></li>
    <li><a href="link.htm">Link zu einer Seite</a></li>
</ul>
⑤ <p id="element13">Fortune plango vulnera stilantibus ocellis,
    quod sua michi munera subtrahit rebellis.</p>
⑥ <p id="nichtvorhanden">Omnia Sol temperat purus et subtilis,
    novo mundo reserat faciem Aprilis; ad Amorem prosperat animus.
    </p>
</body>
</html>

```

- ① Über den ID-Selektor #bild1 legen Sie fest, dass das entsprechende Bild 200 Pixel vom linken und 75 Pixel vom oberen Seitenrand positioniert werden soll. Achtung: Es verdeckt unter Umständen andere Elemente, die sich an dieser Stelle befinden (mehr zu absolut positionierten Elementen in Kapitel 10).

ID-Selektoren

- Aktuelle Seite
- Link zu einer Seite
- Link zu einer Seite
- Link zu einer Seite



Fortune plango vulnera stilantibus ocellis,
quod sua michi munera subtrahit rebellis.

Omnia Sol temperat purus et subtilis,
novo mundo reserat faciem Aprilis; ad Amorem prosperat animus.

kap02\selektor-id.htm

- ② Dem ID-Selektor element13 wird eine Schriftgröße von 150 % zugewiesen.
- ③ Sollte im HTML-Dokument eine unsortierte Liste als main_navigation gekennzeichnet sein, so wird diese mit roter Schrift auf grauem Hintergrund dargestellt werden (in der Praxis macht die Auswahl mittels elementname#ID-name selten Sinn, da die ID bereits eine eindeutige Bezeichnung ist – es ist aber möglich, Element- und ID-Selektor zu kombinieren, was hier gezeigt werden soll).
- ④ Die Liste mit der ID main_navigation wird rot auf grauem Grund dargestellt (die enthaltenen Links haben keine Formatierungen und erscheinen standardmäßig blau).
- ⑤ element13 wurde diesem Absatz mitgegeben – die Schrift dieses Absatzes wird 150 % groß sein.
- ⑥ Im Stylesheet wurde der ID nichtvorhanden kein Format zugewiesen. Somit wird der Inhalt wie vom Browser vorgegeben formatiert.
- ⑦ Die Angabe einer ID muss sich nicht auf einen Absatz beziehen. Hier wird eine eingebundene Grafik gekennzeichnet. Daher greift der Selektor #bild1 und das Bild wird an die festgelegte Stelle gesetzt. Dabei verdeckt es den Text.

2.12 Attribut-Selektor

Selektor [Attribut]

Über diesen Selektor können Sie Elemente mit bestimmten Attributen und Attributwerten auswählen. Zum Beispiel können Sie einen Hyperlink mit einem bestimmten `href`-Wert oder eine Grafik mit einem speziellen `alt`-Text selektieren und formatieren. Sie können auch festlegen, dass Sie generell Tabellen mit der Angabe `width` auswählen möchten.

Der Attribut-Selektor unterteilt sich in zwei Bereiche. Der erste Teil identifiziert ein Element. Der zweite Teil gibt die Bedingungen für das Attribut an. Sieben verschiedene Angaben sind möglich:

[Attribut]	Jedes Element mit dem angegebenen Attribut wird ausgewählt. Beispiel: <code>[class]</code> Wählt: <code><div class="foo"></code>
[Attribut="Wert"]	Der Attributname und der Wert müssen genau die angegebenen Werte aufweisen. Beispiel: <code>[class="foo"]</code> Wählt: <code><div class="foo"></code>
[Attribut~="Wert"]	Ein Attribut eines Elements kann auch mehrere Werte beinhalten. Die Werte müssen dabei durch ein Leerzeichen voneinander getrennt sein. Mit dem Tilde-Zeichen (~) können Sie überprüfen, ob mindestens ein Wert übereinstimmt. Beispiel: <code>[class~="foo"]</code> Wählt: <code><div class="bar foo"></code>
[Attribut = "Wert"]	Mit dem Pipe-Zeichen () können Sie überprüfen, ob ein Teil des Attributwertes übereinstimmt. Die Einschränkung ist, dass sich der gesuchte Teil am Anfang des Wertes befinden muss und durch ein Minuszeichen – von dem restlichen Werte-Teil getrennt ist. Beispiel: <code>p[class = "foo"]</code> Wählt: <code><p class="foo-01"></code>
[Attribut^="Wert"]	Ein Attribut, dessen Wert mit dem hier angegebenen Wert beginnt. Beispiel: <code>p[class^=foo]</code> Wählt: <code><p class="fooend"></code>
[Attribut\$="Wert"]	Ein Attribut, dessen Wert mit dem hier angegebenen Wert endet. Beispiel: <code>p[class\$=jpg]</code> Wählt: <code></code>
[Attribut*="Wert"]	Ein Attribut, dessen Wert den hier angegebenen Wert an beliebiger Stelle enthält. Beispiel: <code>p[class*=foo]</code> Wählt: <code><p class="barfoo-01"></code>

Beispiel: *kap02\selektor-attribute.htm*

Das Beispiel soll einige Möglichkeiten der Auswahl mittels Attribut-Selektor aufzeigen.

```

<html lang="de">
<head>
  <title>Attribut-Selektoren</title>
  <style>
    ① p[class]           { width: 80%; }
    ② p[class="falsch"]   { text-decoration: line-through; }
    ③ p[class~="korrigiert"] { background-color: #080; color: #fff; }
    ④ p[class|= "weitere"] { background-color: #008; color: #fff; }
  </style>
</head>
<body>
  <h2>Attribut-Selektoren</h2>
  ⑤ <p>Qui autem auscultare nolet, exsurgat foras, ut sit, ubi sedat
  ile...</p>
  ⑥ <p class="falsch">Qui autem auscultare nolet, exsurgat foras,
  ut sit, ubi sedat ile qui auscultare vult. Id nos Latine gloriosum
  dicimus...</p>
  ⑦ <p class="korrigiert wert2 wert3">Qui autem auscultare nolet,
  exsurgat foras, ut sit, ubi sedat ile qui auscultare vult...</p>
  ⑧ <p class="weitere-informationen">Zephyrus nectareo spirans in
  odore, certiam pro bravia curramus in amore...</p>
  ⑨ <p class="weitere-lektuere">Zephyrus nectareo spirans in odore,
  certiam pro bravia curramus in amore...</p>
</body>
</html>
```

- ① Die Breite aller Absätze mit irgendeiner Klasse wird auf 80 % festgelegt.
- ② Absätze mit der Klasse falsch werden durchgestrichen dargestellt.
- ③ Wenn Absätze einen oder mehrere Werte enthalten und einer davon korrigiert heißt, bekommen diese einen grünen Hintergrund, Text wird weiß.
- ④ Sollte ein Attributwert von `<p>` mit weitere- beginnen, wird der Absatz blau unterlegt.
- ⑤ Diesem Absatz wurde keine Klasse zugewiesen – hier verwendet der Browser die Standardformatierung.
- ⑥ Dem Absatz wird wie allen folgenden die Formatierung ① zugewiesen. Außerdem wird der Text durchgestrichen, wie in ② festgelegt.

Attribut-Selektoren

Qui autem auscultare nolet, exsurgat foras, ut sit, ubi sedat ile Zephyrus nectareo spirans in odore, certiam pro bravia curramus in amore. Zephyrus nectareo spirans in odore, certiam pro bravia curramus in amore...

Qui autem auscultare nolet, exsurgat foras, ut sit, ubi sedat ile qui auscultare vult. Id nos Latine gloriosum dicimus. Zephyrus nectareo spirans in odore, certiam pro bravia curramus in amore...

Zephyrus nectareo spirans in odore, certiam pro bravia curramus in amore. Zephyrus nectareo spirans in odore, certiam pro bravia curramus in amore...

Zephyrus nectareo spirans in odore, certiam pro bravia curramus in amore. Zephyrus nectareo spirans in odore, certiam pro bravia curramus in amore...

kap02\selektor-attribute.htm

- ⑦ Hier wurden dem Attribut `class` mehrere Werte zugewiesen. Da einer davon korrigiert lautet, wird dieser Text grün unterlegt, wie unter ③ festgelegt.
- ⑧ Hier lautet der Wert des `class`-Attributs `weitere-informationen`. Er beginnt also mit `weitere-` und deshalb wird der Hintergrund blau gefüllt. Das wurde in ④ festgelegt.
- ⑨ Hier lautet der Wert des `class`-Attributs `weitere-lektuere`. Auch hier trifft die Bedingung von ④ zu – daher wird auch dieser Absatz blau unterlegt.

2.13 Pseudoklassen

Pseudoklassen dienen dazu, etwas auszuwählen, was Sie nicht explizit in Ihrem HTML-Dokument notiert haben, zum Beispiel die erste Zeile eines Absatzes oder das letzte Kindelement einer unsortierten Liste. Häufig werden sie auch verwendet, um besuchte Links anders zu formatieren als unbesuchte.

Abgrenzung zu Pseudoelementen (siehe nächster Abschnitt)

Als Pseudoelement werden Bereiche Ihrer Website bezeichnet, die Sie per CSS wie ein Element formatieren können, obwohl sie nur ein Teil eines tatsächlich vorhandenen Elementes sind, zum Beispiel die erste Zeile eines Absatzes.

Mit Pseudoklassen wählen Sie dagegen ein komplettes Element, so, als ob dieses eine Klasse hätte (z. B. den ersten Absatz in einem Artikel)

Pseudoklassen wird ein Doppelpunkt (:) vorangestellt, Pseudoelementen ab CSS3 ein doppelter Doppelpunkt (::).²

Die Bezeichnungen der Pseudoklassen und Pseudoelemente sind fest vorgegebene Schlüsselwörter. Diese stellen einen möglichen Zustand oder die Position eines Elements dar.

So ist es mit dem folgenden Selektor möglich, das erste Kind von mehreren Geschwistern auszuwählen.

Pseudoklasse `:root`

```
:root { Eigenschaft: Wert }
```

Die Angabe von `:root` entspricht in HTML-Dokumenten dem Element `html` – allerdings wählt es in XML-Dokumenten wie SVG ebenfalls das Wurzelement aus, ohne dass dieses explizit benannt werden müsste (in diesem Beispiel also `svg`).

² In CSS Level 2.1 wurde auch den Pseudoelementen nur ein einfacher Doppelpunkt vorangestellt.

Pseudoklasse :first-child

```
:first-child { Eigenschaft: Wert }
```

Die Angabe von `:first-child` wirkt wie `*:first-child` und selektiert jedes erste Kind eines übergeordneten Elementes (zum Beispiel den ersten Listeneintrag einer Liste).

Um die Selektion auf einen Absatz zu beschränken, geben Sie `p:first-child` an. Das wählt alle Absätze aus, die das erste Kind in einem Elternelemente sind.

Beispiel: [kap02\pseudo_first-child.htm](#)

Sehen Sie sich den folgenden Quelltext genau an, um die Theorie an einem praktischen Beispiel nachzuvollziehen.

```
<html lang="de">
<head>
<title>Pseudoklasse :first-child</title>
<style>
①   * :first-child { background-color: #eee; }
②   p:first-child { color: #800; }
③   div > p:first-child { font-size: 18px; }
</style>
</head>
<body>
④   <p>Der erste Absatz p als Kind vom Element body.</p>
      <p>Der zweite Absatz p als Kind vom Element body.</p>
      <p>Der dritte Absatz p als Kind vom Element body.</p>
      <hr>
      <div>
⑤       <p>Der erste Absatz p als Kind vom Element div.</p>
          <p>Der zweite Absatz p als Kind vom Element div.</p>
          <p>Der dritte Absatz p als Kind vom Element div.</p>
      </div>
      <hr>
      <div>
⑥       <h3>Überschrift h3 als Kind vom Element div</h3>
          <p>Der erste Absatz p als Nicht-Kind vom Element div.</p>
          <p>Der zweite Absatz p als Nicht-Kind vom Element div.</p>
          <p>Der dritte Absatz p als Nicht-Kind vom Element div.</p>
      </div>
</body>
</html>
```

- ① Die Angabe der Pseudoklasse `* :first-child` legt fest, dass alle ersten Kindelemente zur Verdeutlichung einen grauen Hintergrund erhalten sollen.
- ② Die rote Schriftfarbe erhalten nur Elemente `p`, die das erste Kind eines übergeordneten Elementes sind.
- ③ Der Absatz `p`, der das erste Kind des Elements `div` ist, soll in der Schriftgröße 18 Pixel dargestellt werden.
- ④ Dieser Absatz ist das erste Kind von `body`. Er wird farbig unterlegt. Da er zusätzlich ein Absatz `p` ist, wird der Inhalt in roter Schriftfarbe eingefärbt.
- ⑤ Dieser Absatz ist das erste Kind von `div` und erhält den grauen Hintergrund. Neben der roten Schriftfarbe wird der Inhalt auch noch in der Schriftgröße 18 Pixel dargestellt. Dies kommt daher, weil der Selektor ③ zutrifft. Das erste Kind ist ein Absatz `p` und gleichzeitig ein Kind von `div`.
- ⑥ Die HTML-Überschrift `h3` ist das erste Kind von `div` und wird somit grau unterlegt. Weitere Formatierungen werden nicht vorgenommen, da für das Element `h3` keine angegeben wurden.
- ⑦ Im Gegensatz zu den anderen beiden bisher formatierten Absätzen trifft für diesen Absatz keine Selektion zu. Er ist nicht das erste Kind des Elements `div`.

Der erste Absatz `p` als Kind vom Element `body`.

Der zweite Absatz `p` als Kind vom Element `body`.

Der dritte Absatz `p` als Kind vom Element `body`.

Der erste Absatz `p` als Kind vom Element `div`.

Der zweite Absatz `p` als Kind vom Element `div`.

Der dritte Absatz `p` als Kind vom Element `div`.

Überschrift `h3` als Kind vom Element `div`

Der erste Absatz `p` als Nicht-Kind vom Element `div`.

Der zweite Absatz `p` als Nicht-Kind vom Element `div`.

Der dritte Absatz `p` als Nicht-Kind vom Element `div`.

Auswirkungen der Pseudoklasse

Wie Sie den ersten Absatz in einem Elternelement auswählen können, wenn ein anderes Element (z. B. `h3`) noch davorsteht, erfahren Sie später im Abschnitt „Einige mächtige CSS3-Selektoren“.

Link-Pseudoklassen `:link` und `:visited`

Browser zeigen standardmäßig nicht besuchte Hyperlinks andersfarbig an als bereits besuchte Hyperlinks. Deshalb unterstützt auch CSS die unterschiedliche Formatierung, indem es die Pseudoklassen `:link` und `:visited` zum Selektieren der beiden Arten von Hyperlinks anbietet.

Daher beziehen sich diese Pseudoklassen auf den Tag `a` mit dem Attribut `href`.

```
a:link { Eigenschaft: Wert; }
:link { Eigenschaft: Wert; }
a:visited { Eigenschaft: Wert; }
:visited { Eigenschaft: Wert; }
```

Die Pseudoklasse `:link` steht für die Links, die noch nicht besucht wurden, `:visited` für die besuchten Links. Beide Klassen schließen sich wechselseitig aus. Ein Element kann also nicht gleichzeitig beide Bedingungen erfüllen.



Die Eigenschaft `visited` kann missbraucht werden, um das Surfverhalten von Nutzern auszuspähen. Gibt man einem Link z. B. Hintergrundbilder mit, ist es anhand des Aufrufs von Bildern möglich, festzustellen, wer welche Seiten abgerufen hat, und das sogar abzuspeichern, um Bewegungsprofile zu erstellen. In aktuellen Browsern ist es nicht möglich, mit `:visited` Links so umfangreich wie andere Elemente zu gestalten. Oft ist nur noch ein Wechsel der Farbe möglich. Eine recht ausführliche Beschreibung des Problems finden Sie in englischer Sprache auf der Webseite <https://developer.mozilla.org/en-US/docs/Web/CSS/:visited>.

Pseudoklassen `:hover`, `:active` und `:focus`

Die Browser können auf Aktionen des Anwenders reagieren, indem die Darstellung von Elementen geändert wird, wenn ein Benutzer diese zum Beispiel mit der Maus überfährt. Am weitesten verbreitet sind Hyperlinks, die beim Überfahren mit der Maus (`hover`), beim Aktivieren (`active`) und beim Setzen des Fokus (`focus`) die Hintergrundfarbe wechseln oder ein Icon vorangestellt bekommen.

Die entsprechenden Pseudoklassen lauten:

```
:hover { Eigenschaft: Wert; }
:active { Eigenschaft: Wert; }
:focus { Eigenschaft: Wert; }
```

Im Gegensatz zu den Link-Pseudoklassen schließen sich diese drei Klassen nicht gegenseitig aus. Dies bedeutet, ein Element kann möglicherweise gleichzeitig alle drei Zustände einnehmen.

Beispiel: `kap02\pseudo_dynamisch.htm`

In dem nachfolgenden Beispiel werden über Hyperlinks die verschiedenen Pseudoklassen aktiviert. Zur besseren Unterscheidung werden die Hyperlinks jeweils unterschiedlich eingefärbt.

```
<html lang="de">
<head>
<title>Dynamische und Link-Pseudoklassen</title>
<style>
①   a:link      { color: red; }
    a:visited    { color: blue; }
    a:hover      { color: yellow; }
    a:focus      { color: black; }
    a:active     { color: lime; }
</style>
</head>
<body>
②   <p><a href="#">Link auf aktive Seite</a>; gilt somit als
        besucht (<b>:visited</b>).</p>
```

```

③ <br><a href="nichtvorhanden.htm">Link auf nicht vorhandene  

Seite</a>; gilt somit als nicht besucht (<b>:link</b>).</p>  

<ol>  

④ <li><b>:hover</b> wird beim Überfahren der Hyperlinks mit  

der Maus aktiv. Die Links werden gelb dargestellt.</li>  

<li><b>:focus</b> wird beim Auswählen der Hyperlinks mit  

der TABULATOR-Taste aktiv. Die Links werden schwarz  

dargestellt.</li>  

<li><b>:active</b> wird beim Aktivieren (Klicken und  

Festhalten) der Hyperlinks aktiv. Die Links werden grün  

dargestellt.</li>  

</ol>  

</body>  

</html>

```

- ① Hyperlinks `a` werden über die verschiedenen Pseudoklassen formatiert. Dazu werden unterschiedliche Schriftfarben angegeben.
- ② Der erste Link verweist auf die bereits geöffnete Webseite und wird nach dem ersten Klick auf den Hyperlink als besuchter Hyperlink markiert (`a:visited`).
- ③ Die verlinkte Webseite `nichtvorhanden.htm` existiert nicht. Somit bleibt der Link immer als unbesucht markiert und erhält die Formatierung der Pseudoklasse `a:link`.
- ④ Die farblichen Veränderungen durch die drei dynamischen Pseudoklassen können Sie sichtbar machen, wenn Sie den Anweisungen in der Beispieldatei folgen.

Link auf aktive Seite; gilt somit als besucht (`:visited`).
Link auf nicht vorhandene Seite; gilt somit als nicht besucht (`:link`). 

1. `:hover` wird beim Überfahren der Hyperlinks mit der Maus aktiv. Die Links werden gelb dargestellt.
2. `:focus` wird beim Auswählen der Hyperlinks mit der TABULATOR-Taste aktiv. Die Links werden schwarz dargestellt.
3. `:active` wird beim Aktivieren (Klicken und Festhalten) der Hyperlinks aktiv. Die Links werden grün dargestellt.

Wirkung der dynamischen Pseudoklassen

Sie haben bereits gelesen, dass die Reihenfolge für den Browser ausschlaggebend ist, wenn Sie für eine Eigenschaft unterschiedliche Werte angeben. Der Browser wird im Konfliktfall den zuletzt notierten Wert anwenden. Im vorliegenden Beispiel provozieren Sie einen solchen Konflikt. Das können Sie folgendermaßen nachvollziehen:

- ▶ Drücken Sie die Tabulator-Taste , bis ein Link ausgewählt (fokussiert) ist.
Die Schrift ist nun schwarz.
- ▶ Fahren Sie mit der Maus über denselben Link.

Die Schriftfarbe wird nun nicht mehr gelb, weil zwei Bedingungen zutreffen (`:hover` und `:focus`). Der Browser entscheidet zugunsten der zuletzt notierten Angaben (`:focus`).

! Bedenken Sie das unbedingt beim Verfassen Ihrer CSS-Datei, denn wie der Link letztendlich aussieht, hängt von der Reihenfolge ab, in der Sie die Regeln notieren!

2.14 Pseudoelemente

::first-line und **::first-letter**

```
::first-line { Eigenschaft: Wert; }
::first-letter { Eigenschaft: Wert; }
```

Das Pseudoelement **::first-line** wählt die erste Zeile eines Elements aus. So können Sie z. B. die Einleitungszeile eines Absatzes andersfarbig unterlegen oder in Kapitälchen setzen. Auch wenn sich die Breite des Browserfensters und somit die Zeilenbreite ändert, wird immer nur die erste Zeile des Elements formatiert.

Mit dem Selektor **::first-letter** können Sie den ersten Buchstaben eines Elements formatieren. Angewendet wird dies beispielsweise beim Erzeugen eines Initials, also des Anfangsbuchstabens eines Absatzes, der sich deutlich vom Rest des Textes abhebt.

Beispiel: *kap02\pseudo_firstline.htm*

Zur Verdeutlichung der Wirkungsweise werden die Pseudoelemente bei zwei unterschiedlichen Absätzen angewendet.

```
<html lang="de">
<head>
    <title>Pseudoelemente: ::first-line & ::first-letter</title>
    <style>
        ① #absatz1::first-line { background-color: #ddd; }
        ② #absatz2::first-letter {
            font-size: 400%; font-style: italic;
            font-weight: bold; float: left; margin-right:10px;
        }
    </style>
</head>
<body>
    <h1>::first-line</h1>
    ③ <p id="absatz1">Fortune plango vulnera stilantibus ocellis,
        quod sua ...</p>
    <h1>::first-letter</h1>
    ④ <p id="absatz2">Qui autem auscultare nolet, exsurgat foras,
        ut sit, ...</p>
</body>
</html>
```

- ① Der Selektor mit dem Pseudoelement `::first-line` wird so notiert, dass er nur bei einem Absatz mit der ID `absatz1` zum Einsatz kommt. Die erste Zeile des Absatzes `p` wird grau unterlegt.
- ② Das Pseudoelement `::first-letter` wird wegen der Angabe `p#absatz2` nur eine Änderung des Absatzes mit der ID `absatz2` bewirken. Die Erläuterung der verwendeten Formate folgt in den weiteren Kapiteln des Buches. Diese Formatierungssangaben erzeugen ein Initial, bei dem der erste Buchstabe vergrößert neben dem eingerückten Text steht.
- ③ In diesem Beispiel sind zwei Absätze unterschiedlich zu formatieren. Deshalb werden sie über eine eindeutige ID angesprochen. In diesem Fall erhält dieser Absatz die Formatierung von `p#absatz1`: Die erste Zeile wird farbig hervorgehoben.
- ④ Entsprechend erhält der zweite Absatz die Formatierungssangaben, die unter ② angegeben wurden, und bekommt damit ein Initial.

Geben Sie die Pseudoelemente ohne einen vorangestellten Elementnamen an, werden von allen Elementen die erste Zeile und der erste Buchstabe angesprochen und formatiert.

::before und ::after

```
::before { Eigenschaft: Wert; }
::after   { Eigenschaft: Wert; }
```

Mit den beiden Elementen können Sie vor (`::before`) und nach (`::after`) selektierten Elementen Texte oder Bilder hinzufügen. Zum Beispiel könnten Sie jeden Absatz mit einer kleinen Grafik abschließen.

! Beachten Sie bei der Nutzung dieser Pseudoelemente, dass CSS ausschließlich für Darstellungszwecke verwendet werden soll. Inhalte gehören in das HTML-Dokument! Denkbar sind zum Beispiel Anführungszeichen um alle Zitate (`blockquote` und `q`) oder die Ausgabe des Linkziels in der Druckausgabe.

Beispiel: kap02\pseudo_before-after.htm

Für das Beispiel, bei dem am Anfang und am Ende des Absatzes automatisch Grafiken eingeblendet werden, verwenden Sie die CSS-Eigenschaft `content`, mit der Sie Inhalte hinzufügen können. Möglich sind dabei URL-Angaben von Grafiken sowie normale Textangaben. Mehr Informationen dazu erhalten Sie in einem der nachfolgenden Kapitel.

```
<html lang="de">
<head>
  <title>Pseudoelemente: ::before & ::after</title>
  <style>
```

::first-line

Fortune plango vulnera stolidibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronte capillata, sed plerunque sequitur occasio calvata. In Fortune solio sedera m elatus, prosperitatis vario flore cononatus; quicquid enim florui felix et beatus, nunc a summo corrui gloria privatus. Fortune rota volvitur descendendo minoratus, alter in altum tollitur, nimis exaltatus rex sedet in vertice; caveat ruinam! Nam sub axe legimus hecubam redinam.

::first-letter

Q uia autem auscultare nolet, exsurgat foras, ut sit, ubi sedat ille qui auscultare vult. Id nos Latine gloriosum dicimus. Quia adsedis causa in festivo loco, comoedial quam nos actuari sumus et argumentum et nomen vobis eloquar. Alazon Graece huic nomen est comoediae: Id nos Latine gloriosum dicimus. Flora veris leta facies mundo propinatur, hemialis acies victa iam fugatur. In vestitu vario Flora principiatur, nemorum dulcissimo que cantu celebratur. Flore fusu gremio Phebus novo more risum dat, huc vario iam stipite flore.

Veränderte Darstellung mit Pseudoelementen

```

①    p::before { content:"... "; }
②    p::after  { content:url(images/after.gif); }
</style>
</head>
<body>
  <h2>::before und ::after</h2>
  <p>Qui autem auscultare nolet, exsurgat foras, ut ...</p>
</body>
</html>

```

- ① Absätzen `p` wird das Pseudoelement `::before` zugewiesen, mit dem Inhalte vorangestellt werden können. Über die Eigenschaft `content` legen Sie als Ausgabe die Zeichenfolge "..." fest.
- ② Außerdem erzeugen Sie das Pseudo-element `::after`. Die Eigenschaft `content` erhält jetzt über die Angabe von `url()` eine Pfadangabe zu einer Grafik. Diese soll am Ende des Absatzes `p` eingefügt werden.
- ③ Dem Absatz werden die Zeichen "..." vorangestellt und dessen Ende durch die Grafik `after.gif` (■) visualisiert.

:before und :after

... Qui autem auscultare nolet, exsurgat foras, ut sit, ubi sedat ile qui auscultare vult. Id nos Latine gloriosum dicimus. Qua adseditis causa in festivo loco, comoedial quam nos actuari sumus et argumentum et nomen vobis eloquar. Alazon Graece huic nomen est comoediae: Id nos Latine gloriosum dicimus. Flora veris leta facies mundo propinatur, hemalis acies victa iam fugatur. In vestitu vario Flora principatur, nemorum dulcisono que cantu celebratur. Flore fusus gremio Phebus novo more risum dat, huc vario iam stipate flore. ■

*Einfügen von zusätzlichen Inhalten
(kap02\pseudo_before-after.htm)*

! Wenn Sie einem Absatz Texte wie zum Beispiel die drei Punkte mittels `::before` hinzufügen, sehen die Standards vor, dass diese vor dem im HTML-Dokument befindlichen Text, aber noch innerhalb des Absatzes einzufügen sind. Da leere Elemente keine Inhalte haben, stellen standardkonforme Browser wie der Firefox keine Texte oder Bilder dar, die Sie zum Beispiel Bildern (`img`) oder horizontalen Linien (`hr`) als Pseudoelemente mitgeben.

2.15 Weitere Selektoren

HTML-Klassen dienen nicht vorrangig der Formatierung

Die Möglichkeiten, die CSS bietet, um bestimmte Elemente auszuwählen, sind so zahlreich, dass es immer seltener nötig wird, HTML mit Klassen oder Ids anzureichern, um Seiten zu gestalten. Desto mehr Augenmerk sollten Sie bei der Erstellung Ihrer HTML-Dokumente auf eine rein logische Auszeichnung der Inhalte legen. Für die meisten Selektionswünsche sind Sie damit bereits gut gerüstet.

Dennoch macht es Sinn, im HTML **Elementen mit gleichen Eigenschaften** logische Klassen zu vergeben – ob Sie diese mal für die Formatierung benötigen, sollte bei der Auszeichnung des HTMLs keine Rolle spielen.³

Die folgenden Selektoren nehmen dem Entwickler viel Arbeit ab. So ist es leicht möglich, in langen Listen jeden zweiten Listeneintrag mit einer Hintergrundfarbe zu versehen. Dennoch sollten diese Selektoren mit Bedacht eingesetzt werden, da sie vom Browser und damit auch vom verwendeten Computer relativ viel Rechenleistung verlangen. Ein großzügiger Einsatz kann sich vor allem auf mobilen Endgeräten spürbar auswirken.

:lang()

```
:lang() { Eigenschaft: Wert; }
```

Hiermit werden Elemente in der angegebenen Sprache ausgewählt. Die Klammer ist verbindlich, in ihr wird die Sprache notiert, für die Sie eine Formatierung mitgeben wollen.

Beispiel 1

```
:lang(en) { color: #f00; }
```

Diese Angabe stellt alle Texte rot dar, die der Browser aufgrund Ihrer Sprachangaben als englischen Text erkennt.

Beispiel 2

```
:lang(de) quote::before,  
quote:lang(de)::before {  
    content: ''';  
}  
:lang(de) quote::after,  
quote:lang(de)::after {  
    content: ''';  
}
```

Diese Angabe fügt allen Zitaten, die in einem Bereich mit deutscher Sprache vorkommen oder selbst als deutschsprachig ausgezeichnet sind, die typischen deutschen Anführungszeichen hinzu.

Wenn Sie so etwas für mehrere Sprachen festlegen, sorgt Ihre Formatvorlage automatisch für Zitate, mit sprachtypischen Anführungszeichen – alles, was Sie dafür tun müssen, ist, die Sprachen korrekt anzugeben.

³ Noch mehr Bedeutung können Sie Ihrem HTML-Dokument mittels Mikroformaten hinzufügen. Da Sie hierfür vorgegebene Attribute verwenden, haben Sie auf so ausgezeichnete Elemente mit dem Attribut-Selektor eine sehr fein granulierte Selektionsmöglichkeit.

:target()

```
:target() { Eigenschaft: Wert }
```

Hiermit können Elemente hervorgehoben werden, die das Ziel eines geklickten Links sind.

Beispiel

```
#foo:target { color: #f00; }
```

Diese Angabe sorgt für roten Text, wenn sie sich in einem Element mit der ID `foo` befindet, das per Link angesprungen wurde.

Das ist hilfreich, um anzusehen, auf welchen Bereich einer Seite ein Link zielt – insbesondere, wenn sich Link und Ziel beide im sichtbaren Bereich befinden und der Browser nicht an die betreffende Stelle springt. Dann ist für Nutzer oft weder ersichtlich, ob sich überhaupt etwas getan hat, noch, wofür der Link gut ist.

:not()

```
:not() { Eigenschaft: Wert; }
```

Hiermit wählen Sie alle Elemente aus, für welche die angegebene Bedingung nicht zutrifft.

```
:not(p) { color: #f00; }
```

Dies führt dazu, dass alle Texte, die sich **nicht** in Absätzen befinden, rot dargestellt werden.

Sie können den not-Selektor auch mit anderen Selektoren kombinieren:

```
a:not(:hover) { color: #f00; }
```

Auf diese Weise färben Sie alle Links rot, die **nicht** mit der Maus überfahren werden.



Innerhalb der Klammern des `:not()`-Selektors dürfen sich nur einfache Selektoren befinden. Das sind beispielsweise der ID-Selektor, der Element-Selektor oder der Klassen-Selektor.

Kombinatoren wie Nachfahren- oder Kind-Selektor sind nicht erlaubt. Eine Liste der erlaubten Selektoren finden Sie in der Spezifikation unter:

<https://www.w3.org/TR/css3-selectors/#negation>

:nth-child(n)

Dieser Selektor ermöglicht es, Elemente auszuwählen, die sich an einer bestimmten Stelle in einer Reihe von Kindelementen befinden.

```
:nth-child() { Eigenschaft: Wert }
```

Die Bezeichnung des Selektors ist an die englische Kurzschreibweise für Ordnungszahlen angelehnt, also fourth (vierter) oder fifth (fünfter). Diese finden sich in englischen Texten häufig in der Form 4th oder 5th.

Beispiel 1

```
ol li:nth-child(4) { color: #f00; }
```

Mit diesem Beispiel wird der vierte Listeneintrag von sortierten Listen rot gefärbt.

Sie können aber auch mehrere Elemente auswählen.

Beispiel 2

```
li:nth-child(2n) { background-color: #f00; }
```

Das n hinter dem Zahlenwert führt dazu, dass jedes zweite Element rot unterlegt wird. So lassen sich Listen oder Tabellen erstellen, deren Zeilen abwechselnd weiß und rot dargestellt werden. Statt 2n können Sie auch die Schlüsselwörter odd und even verwenden, um alle geraden oder ungeraden Elemente aus einer Reihe auszuwählen. Auch so lassen sich abwechselnde Darstellungen umsetzen.

Beispiel 3

```
li:nth-child(-n+5) { background-color: #f00; }
```

Auf die hier dargestellte Weise werden die ersten 5 Elemente ausgewählt.

Beispiel 4

```
tr:nth-child(n+2):nth-child(-n+8) { background-color: #f00; }
```

Die Selektoren lassen sich kombinieren. Hier werden die Tabellenzeilen 2 bis 8 ausgewählt.

:nth-last-child(n)

```
:nth-last-child() { Eigenschaft: Wert }
```

Der Name dieser Pseudoklasse verrät, wie dieser Selektor einzusetzen ist. Mit ihm werden Elemente ausgewählt, die an einer bestimmten Stelle stehen, wenn man beim letzten Element zu zählen beginnt. Also das vorletzte oder fünftletzte Element.

Alles zu :nth-child(n) Gesagte trifft auch für :nth-last-child() zu – nur eben in umgekehrter Reihenfolge.

Beispiel

```
li:nth-last-child(odd) { background-color: #f00; }
```

odd bewirkt, dass jedes ungerade Element rot unterlegt wird. Also das letzte, drittletzte, fünftletzte und so weiter. So lassen sich Listen oder Tabellen erstellen, deren Zeilen abwechselnd weiß und rot dargestellt werden.

:last-child

```
:last-child { Eigenschaft: Wert }
```

Damit wird lediglich das letzte Element aus einer Reihe von Elementen ausgewählt.

:nth-of-type (n)

```
:nth-of-type(n) { Eigenschaft: Wert }
```

Hiermit werden Elemente angesprochen, auf die zwei Bedingungen zutreffen müssen: Sie müssen vom gleichen Typ sein (also zum Beispiel Absätze) und sie müssen Geschwister sein. n kann wie bei :nth-child(n) durch Zahlen oder durch eine Zahl und n für jedes n-te Auftauchen eines Elementes oder durch ein Schlüsselwort wie odd oder even ersetzt werden.

Beispiel

```
div p:nth-of-type(odd) { background-color: #f00; }
```

Dieses Konstrukt sorgt dafür, dass der erste, dritte, fünfte usw. Absatz eine rote Unterlegung erhält, wenn er sich in einem div befindet.

:first-of-type

```
:first-of-type { Eigenschaft: Wert }
```

Hiermit wird das erste Element von gleichartigen Geschwistern ausgewählt, also zum Beispiel der erste Absatz in einem div – auch dann, wenn vor diesem Absatz andere Elemente wie eine Überschrift oder Liste vorkommen.

```
div p:first-of-type { background-color: #f00; }
```

:only-child

```
:only-child { Eigenschaft: Wert }
```

Hiermit wird ein Element ausgewählt, wenn es sich als einziges im Elternelement befindet.

Beispiel

```
div p:only-child { background-color: #f00; }
```

Im Beispiel bekommt ein Absatz nur dann einen roten Hintergrund, wenn er sich allein in einem div befindet.

2.16 Kommentare

```
/* Kommentar */
```

Kommentare werden vom Browser ignoriert und nicht angezeigt. Sie dienen dazu, Notizen anzulegen, die das Bearbeiten der Formatvorlagen zu einem späteren Zeitpunkt oder für andere erleichtern.

Mit `/*` leiten Sie den Kommentar ein, der sich auch über mehrere Zeilen erstrecken kann. Mit `*/` beenden Sie den Kommentar.

Auch wenn es unbequem oder trivial erscheinen mag, ist es sehr ratsam, mit Kommentaren seine Formatvorlagen verständlicher zu machen.

Beispiel

```
/* Schriftfarbe im ersten
   Absatz ist immer blau */
p:first-of-type { color: blue; }

li { font-family:"Courier New" } /* Listen immer in Courier */
```

2.17 Übung

Theoriefragen zu Selektoren

Level		Zeit	ca. 15 min
Ergebnisdatei	<i>kap02\uebung1-4.doc</i>		

1. Erläutern Sie den Zusammenhang zwischen einem HTML-Element und dem Element-Selektor.
2. Erklären Sie den Unterschied zwischen einem Nachfahren-Selektor und einem Kind-Selektor.
3. Wann verwenden Sie einen Klassen-Selektor, wann einen ID-Selektor?
4. Nennen Sie Selektoren, die Sie sinnvoll einsetzen können, um den folgenden Elementen Formate hinzuzufügen:
 - a. Sie möchten ein einzelnes Element der Webseite formatieren.
 - b. Sie möchten mehreren Absätzen, die einem gemeinsamen Zweck dienen, das gleiche Format zuweisen.
 - c. Sie möchten den ersten Absatz nach einer Überschrift andersfarbig formatieren.
 - d. Sie möchten alle ungeraden Elemente einer Liste formatieren.
 - e. Sie möchten den letzten Link in einem Elternelement formatieren – auch wenn danach noch andere Elemente wie `span` oder `strong` folgen.

3

Eigenschaften von Schrift

3.1 Schriftfamilie

Durch das Ändern der Schriftfamilie beeinflussen Sie das Aussehen von Texten am stärksten. Schriftfamilien werden umgangssprachlich meist als „Schriftart“ bezeichnet. Der Begriff Familie führt daher, dass eine Schriftart aus mehreren Schriftschnitten besteht, in der Regel eine normale, eine fette, eine kursive und eine fett-kursive Variante.

```
font-family: Schriftart [, 'Schriftart 2', ...];
```

Werte	<ul style="list-style-type: none">✓ Name der Schriftart✓ serif für Serifenschriften wie z. B. Georgia oder Times New Roman✓ sans-serif für serifenlose Schriften wie z. B. Arial oder Helvetica✓ cursive für schreibschriftähnliche Schriften wie z. B. Zapf Chancery✓ fantasy für Zierschriften✓ monospace für Schriften, deren Buchstaben alle die gleiche Breite besitzen, wie z. B. Courier (New)
Standardwert	Eine Serifenschriftart, abhängig vom verwendeten Betriebssystem
Vererbbar	Ja
Anwendbar auf	Alle Elemente

Verschiedene Computersysteme verwenden unterschiedliche Schriftfamilien. So besitzt beispielsweise Windows die Schrift Arial, das Betriebssystem UNIX hingegen nicht. UNIX besitzt dagegen die Schrift Geneva, die der Schriftart Arial ähnlich ist. Damit Ihr Dokument in jedem Browser korrekt angezeigt werden kann, sollten Sie mehrere Schriftarten angeben. Die Namen der Schriftarten trennen Sie durch Kommata. Der Browser versucht, die erste angegebene Schriftart zu laden. Ist diese nicht vorhanden, sucht er die nächste im System usw. Ist keine der angegebenen Schriftarten auf dem Computersystem vorhanden, stellt der Browser den formatierten Textabschnitt in seiner Standardschrift dar.

Um dem Problem aus dem Weg zu gehen, dass bestimmte Schriftarten beim Anwender nicht installiert sein könnten, sollten Sie am Ende Ihrer Liste eine generische Schrift angeben, wie `serif` oder `sans-serif`. Der Browser entscheidet in diesem Fall selbstständig, welche Schriftart er zum Anzeigen benutzt, wenn die anderen Schriften nicht vorhanden sind.

Das W3-Konsortium empfiehlt, Namen von Schriften, die Leerzeichen enthalten, in Anführungszeichen zu setzen. Dabei ist es gleich, ob Sie einfache oder doppelte Anführungszeichen verwenden. Erlaubt sind die Schreibweisen `font-family:"Times New Roman"` und `font-family:'Times New Roman'`.

Beispiel: *kap03\font-family.htm*

Im Folgenden werden verschiedene Schriftarten festgelegt. Es werden Fantasie-Schriften (z. B. Bolognese) verwendet, damit der Browser generische oder vom Browser vorgegebene Schriftarten verwendet.

```
<html lang="de">
<head>
<title>Schriftfamilie</title>
<style>
①   p.georgia   { font-family: Georgia, "Times New Roman", Times,
    serif; }
②   p.bolognese { font-family: Bolognese, Arial, Helvetica, sans-
    serif; }
③   p.margarita { font-family: Bolognese, Margarita; }
</style>
</head>
<body>
    <h1>Schrift auswählen mit font-family</h1>
④    <p class="georgia">In dieser Zeile wird der Text in der Schriftart
        Georgia angezeigt.</p>
⑤    <p class="bolognese">Da es keine Schriftart Bolognese gibt, wird
        die nächste Schriftart, Arial, verwendet.</p>
⑥    <p class="margarita">Da es weder die Schrift Bolognese noch
        Margarita gibt, erscheint dieser Text in der Standardschriftart
        des Browsers.</p>
⑦    <p style="font-family:monospace">Direkte Formatierung am Element.
        Durch die Angabe von monospace wird eine Schrift mit fester
        Zeichenbreite verwendet.</p>
</body>
</html>
```

- ① Absätzen mit der Klasse `georgia` wird die Schriftart Georgia zugewiesen. Alternativ sollen die Schriften Times New Roman oder Times genutzt werden. Ist keine der angegebenen Schriften verfügbar, soll der Browser irgendeine Serifenschrift auswählen.
- ② Mit der Klasse `bolognese` soll eine Schriftart verwendet werden, die es keinesfalls gibt.
- ③ Mit `margarita` gehen Sie einen Schritt weiter, indem Sie nur Schriften angeben, die nicht existieren.
- ④ Für die Darstellung dieses Absatzes wird die Schriftart Georgia verwendet – falls sie installiert ist.
- ⑤ Dieser Absatz wird in der Schriftart Arial formatiert.
- ⑥ Da es die zugewiesenen Schriftarten nicht gibt, stellt der Browser den Inhalt in der Standardschriftart dar.
- ⑦ Mit dem Wert `monospace` geben Sie an, dass der Browser eine Schriftart nutzen soll, bei der alle Buchstaben die gleiche Breite aufweisen.

Schrift auswählen mit `font-family`

In dieser Zeile wird der Text in der Schriftart Georgia angezeigt.

Da es keine Schriftart Bolognese gibt, wird die nächste Schriftart, Arial, verwendet.

Da es weder die Schrift Bolognese noch Margarita gibt, erscheint dieser Text in der Standardschriftart des Browsers.

Direkte Formatierung am Element. Durch die Angabe von monospace wird eine gleich breite Schrift verwendet.

Unterschiedliche Schriften (kap03\font-family.htm)

Da es nicht auf jedem Computersystem alle Schriften gibt, sollten die Alternativen Ihrem Favoriten ähneln. Die folgenden Gruppen von Schriftarten haben sich durchgesetzt und werden deshalb recht häufig im World Wide Web eingesetzt:

- ✓ `font-family: Georgia, "Times New Roman", Times, serif;`
- ✓ `font-family: Verdana, Arial, Helvetica, sans-serif;`
- ✓ `font-family: "Courier New", Courier, monospace;`

Schriftarten bereitstellen

```
@font-face { font-family: src; }
```

Wenn Sie Schriftarten verwenden möchten, die auf einem System (möglicherweise) nicht vorhanden sind, aber dennoch (unbedingt) verwendet werden sollen, können Sie diese mit `@font-face` bereitstellen.

Hunderte kostenlose Schriften stellt Google bereit unter <https://fonts.google.com/>.



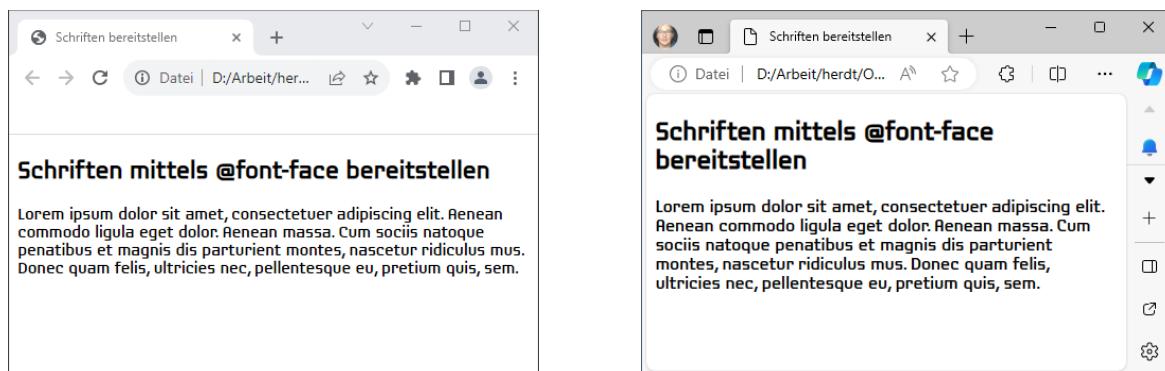
Auch für Schriften gilt das **Urheberrecht!** Achten Sie daher auf die Lizenzen. Viele kommerzielle Schriften dürfen nicht zum Download im WWW bereitgestellt werden.

Beispiel: kap03\font-face.htm

Im Beispiel wird die frei verfügbare Schrift Laconic_Regular im OpenType-Format verwendet. Das Beispiel zeigt die reguläre (also weder fette noch kursive) Version der Schrift.

```
<html lang="de">
<head>
<title>Schriften bereitstellen</title>
<style>
@font-face {
    font-family: 'Laconic_Regular';
    src: url('Laconic-Regular.otf');
}
body { font-family: 'Laconic_Regular', Arial, sans-serif; }
</style>
</head>
<body>
<h1>Schriften mittels @font-face bereitstellen</h1>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem.</p>
</body>
</html>
```

Alle Browser stellen die Schriftart korrekt dar.



Laconic Regular in Google Chrome (links) und Microsoft Edge (rechts) (kap03\font-face.htm)

Wenn möglich, verwenden Sie eine Schrift im woff-Format oder dem moderneren woff2-Format, da diese von allen gängigen Browsern unterstützt wird (außer Opera mini). Außerdem sind die Dateigrößen dieser Schriften vergleichsweise gering:

<https://wiki.selfhtml.org/wiki/Typografie/Schriftformatierung#.40font-face.>

3.2 Schnitte einer Schrift

Bei einer **Schriftfamilie** werden die formal zusammengehörenden Schriften mit demselben Namen in einer Gruppe zusammengefasst, wie z. B. Arial. Diese Gruppe unterscheidet sich dann im **Schriftschnitt**, wie normal, fett, halbfett, kursiv. Arial ist z. B. eine Schriftfamilie von Microsoft mit elf verschiedenen Schriftschnitten. Die bekanntesten davon heißen Arial, Arial Italic, Arial Black, Arial Bold. Informationen zu diesen und anderen Microsoft-Schriften finden Sie unter <https://learn.microsoft.com/de-de/typography/font-list/>. Dort finden Sie ebenfalls die Information, welche Schriftarten standardmäßig mit Windows zusammen installiert werden.

Stärke der Schrift

Die Stylesheet-Eigenschaft `font-weight` ermöglicht eine abgestufte Angabe für das Gewicht von Zeichen, also wie fett genau die Buchstaben sein sollen.

font-weight: Wert;

Werte	<ul style="list-style-type: none">✓ <code>normal</code> (normal)✓ <code>bold</code> (fett)✓ 100–900 in 100er-Schritten (100 = extra dünn; 900 = extra fett)✓ <code>bolder</code> (stufenweise Erhöhung)✓ <code>lighter</code> (stufenweise Verringerung)
Standardwert	Normal
Vererbbar	Ja
Anwendbar auf	Alle Elemente

Voraussetzung für die unterschiedliche Darstellung der Schriftstärke ist, dass die Schriftart eine solche Abstufung unterstützt.

Zusätzlich sind die Angaben `bolder` (dicker) und `lighter` (dünn) möglich. Sie bewirken jedoch erst eine Änderung, wenn einem Textabschnitt bereits eine bestimmte Stärke zugewiesen wurde.

Stil der Schrift

Den Stil ändern Sie, indem Sie die Schriftlage verändern. Um einen Textabschnitt kursiv darzustellen, verwenden Sie die folgende Eigenschaft:

font-style: Wert;

Werte	✓ normal (normal) ✓ italic (kursiv) ✓ oblique (kursiv)
Standardwert	Normal
Vererbbar	Ja
Anwendbar auf	Alle Elemente

Wenn Text vom Browser kursiv dargestellt werden soll, versucht dieser, die Schriftart mit dem entsprechenden Schriftschnitt zu laden, beispielsweise Arial Italic. Besitzt eine Schriftart nur den Schnitt Oblique, so ist der Wert **oblique** zu übergeben.

Kapitälchen

Kapitälchen sind ein Schriftschnitt, bei dem die Kleinbuchstaben (in der Typografie oft „Gemeine“ genannt) die Form der entsprechenden Großbuchstaben (Versalien) besitzen (z. B. BEISPIEL).

font-variant: Wert;

Werte	✓ normal (normal) ✓ small-caps (Kapitälchen)
Standardwert	Normal
Vererbbar	Ja
Anwendbar auf	Alle Elemente

Beispiel: *kap03\font-schnitt.htm*

Jeder Schriftschnitt wird in einem separaten Klassen-Selektor definiert. Den Absätzen werden diese Klassen in verschiedenen Kombinationen zugewiesen, um die verschiedenen Formatierungen darzustellen.

```

...
<head>
<style>
①   body      { font-family: Arial,Helvetica,sans-serif; }
②   .fett      { font-weight: bold; }
     .kursiv    { font-style: italic; }
     .kapitaelchen { font-variant: small-caps; }
</style>
</head>
...

```

- ① Dem Element **body** wird die Schriftart Arial zugewiesen. Aufgrund der Vererbung werden die Inhalte der untergeordneten Elemente ebenfalls in der Schrift Arial ausgegeben.
- ② Je nach Schriftschnitt wird ein entsprechender Klassen-Selektor festgelegt.

Die Ausgabe der Abbildung erfolgt über den HTML-Quelltext mit der mehrfachen Klassenzuweisung.

```
<p>normale Formatierung</p>
<p class="fett">Fette Formatierung</p>
<p class="fett kursiv">Fette und kursive
Formatierung</p>
<p class="fett kapitaelchen">Fette
Formatierung mit Kapitaelchen</p>
<p class="fett kursiv kapitaelchen">Fette
und kursive Formatierung mit
Kapitälchen</p>
<p class="kursiv">Kursive Formatierung</p>
<p class="kursiv kapitaelchen">Kursive
Formatierung mit Kapitälchen</p>
<p class="kapitaelchen">Kapitälchen</p>
```

Angabe von verschiedenen Schriftschnitten

normale Formatierung
Fette Formatierung
Fette und kursive Formatierung
FETTE FORMATIERUNG MIT KAPITÄLCHEN
FETTE UND KURSIVE FORMATIERUNG MIT KAPITÄLCHEN
Kursive Formatierung
KURSIVE FORMATIERUNG MIT KAPITALCHEN
KAPITÄLCHEN

kap03\font-schnitt.htm

Mehrfache Klassenzuweisung

Auch für dieses Beispiel gilt: Die Klassennamen **fett**, **kursiv** und **kapitaelchen** sollen das Beispiel besonders leicht nachvollziehbar machen. Für eine Webseite sind Bezeichnungen, die sich auf die Formate beziehen, nicht geeignet. **Vergeben Sie immer logische Namen.**

Auf den ersten Blick scheint es bequem zu sein, das Aussehen von Elementen mittels Klassennamen zu beeinflussen. Auf diesem Prinzip beruhen **CSS-Frameworks** wie Bootstrap oder Foundation. Der Nachteil ist, dass man so de facto wieder in die Zeit vor der Erfindung von CSS zurückgeworfen wird, weil Sie bei späteren Änderungen wieder sämtliche HTML-Dokumente anpassen müssen. Auch wenn einem ein Content-Management-System oder eine selbstprogrammierte Anwendung die Arbeit erleichtern können, weil dort eine einmal eingefügte Klasse an vielen Stellen erscheint und so weniger geändert werden muss, bleiben zwei wesentliche Probleme bestehen.

- ✓ Wenn Sie nicht alle relevanten Stellen im Kopf haben (und das hat niemand bei umfangreichen Webprojekten), sind Fehler vorprogrammiert – Sie müssten bei jeder Änderung Hunderte von Seiten kontrollieren, um die Fehler zu finden. Und wenn Sie welche finden, stehen Sie vor dem nächsten Problem.
- ✓ Da Sie alle Elemente aufgrund Ihres Aussehens und nicht aufgrund Ihrer logischen Bedeutung benannt haben, haben Sie sowohl allen Absende-Buttons als auch allen Abbrechen-Buttons möglicherweise denselben Rahmen mitgegeben. Es ist Ihnen nun nicht mehr möglich, nur allen Abbrechen-Buttons einen anderen Rahmen mitzugeben. Also müssen Sie im laufenden Betrieb diesen Fehler korrigieren und ein Unterscheidungsmerkmal einfügen, indem Sie den Abbrechen-Button beispielsweise eine Klasse `abbrechen` mitgeben – warum nicht gleich so?¹

¹ Hier können Präprozessoren wie SASS eine Hilfe sein, um einerseits wiederverwertbares CSS zu schreiben, das verständlich benannt ist (z. B. `border2pxBlack`) und andererseits diese Konstrukte Elementen mit logischen Klassen wie `abbrechen` zuordnen zu können. Eine einleuchtende und verständliche Erklärung liefert Gunnar Bittersmann in seinem Vortrag „The best from both worlds“ unter <https://www.youtube.com/watch?v=ot-LoU0MGb0>.

3.3 Maßeinheiten

In CSS-Dateien dürfen Sie zahlreiche Maßeinheiten verwenden. Die meisten machen nur in bestimmten Zusammenhängen Sinn. So lassen sich Zentimeter unmöglich in Pixel umrechnen (Displays haben unterschiedlich große Pixel). Im Druck können Zentimeter sinnvoll sein, denn die Einheit Punkt basiert auf Zoll, die sich wiederum in Zentimeter umrechnen lassen. Vorausgesetzt, es ist bekannt, mit wie viel Punkt per Zoll (dots per Inch, kurz dpi) ein Ausdruck erfolgt.

Grundsätzlich wird zwischen absoluten und relativen Angaben unterschieden.

Absolute Maßeinheiten

<code>px</code>	Pixel
<code>mm</code>	Millimeter
<code>cm</code>	Zentimeter
<code>in</code>	Inch (englisches Längenmaß) 25.4 mm
<code>pc</code>	Punkt (typografisches Maß) 1/72 in = 0.353 mm
<code>pt</code>	Pica (typografisches Maß) 12 pt = 1/6 in = 4.23 mm

Bei der Angabe der Werte sind Dezimalstellen erlaubt. **Als Dezimaltrennzeichen müssen Sie einen Punkt `.` verwenden.**

Relative Maßeinheiten

Relative Angaben benötigen stets einen Bezug. Der Bezug für Schriftgrößen ist entweder die vom Browser verwendete Standardgröße für normalen Fließtext oder eine Angabe, die Sie selbst in Ihrem Stylesheet gemacht haben. Beispiele und visuelle Vergleiche finden Sie in der Datei `kap03\relative_masseinheiten.htm`.

<code>em</code>	<code>1em</code> ist die Höhe des Buchstabens „m“ – dies entspricht in den Standardeinstellungen der Browser 16 Pixel. Mit <code>em</code> können Sie einen Vergrößerungsfaktor angeben (<code>1em = 100%</code>). Die Größenangabe bezieht sich auf die Schriftgröße des aktuellen Elementes. Ausnahme: Die Einheit wird in der Eigenschaft <code>font-size</code> selbst verwendet. Dann bezieht sich <code>em</code> auf die Schriftgröße im Elternelement.
<code>rem</code>	<code>1rem</code> (root em) bezieht sich auf die Schriftgröße des Wurzelementes <code>html</code> . Diese Größe ist immer dieselbe, egal wo Sie sie im Dokument verwenden. Sie hängt nur von den Schriftgrößeneinstellungen des Nutzers oder von Ihren eigenen Schriftgrößen-Angaben für das HTML-Element ab
<code>ex</code>	<code>1ex</code> ist die Höhe des Buchstabens „x“.

ch	1ch entspricht der größten Ausdehnung des Zeichens „0“ (NULL, U+0030), das heißt entweder der Höhe oder der Breite von Null, je nachdem, welches von beiden größer ist.
%	Wird die Schriftgröße mittels font-size in Prozent angegeben, entspricht 100 % der Schriftgröße im übergeordneten Element. Bei Breiten- oder Höhenangaben in Prozent entspricht 100 % der Größe des Elternelementes.

3.4 Schriftgröße

Mit der Eigenschaft font-size können Sie Schriftgrößen angeben.

font-size: WertMaßeinheit;

Werte	<ul style="list-style-type: none"> ✓ Zahlenangaben mit entsprechender Maßeinheit ✓ xx-small, x-small, small (winzig, sehr klein, klein) ✓ medium (normal) ✓ large, x-large, xx-large (groß, sehr groß, riesig) ✓ smaller (nächstniedrigeres Maß) ✓ larger (nächst höheres Maß)
Standardwert	1em; 12pt; 16px; 4.23mm; 0.423cm; 0.166in; 1pc; medium
Vererbbar	Ja
Anwendbar auf	Alle Elemente

Beispiel: *kap03\font-size.htm*

In diesem Beispiel werden für Schriftgrößen verschiedene Maßeinheiten verwendet.

	<pre><html lang="de"> <head> <title>Schriftgröße</title> <style> p { font-family: Georgia, "Times New Roman", Times, serif} p.georgia_px { font-size: 16px; } p.georgia_mm { font-size: 7.5mm; } p.georgia_pt { font-size: 15pt; } p.georgia_pc { font-size: 2pc; } </style> </head> <body> <h2>Schriftgröße verändern mit font-size</h2> </pre>
--	--

```

③ <p class="georgia">Der Text wird in der Standardgröße  

    angezeigt.</p>  

<p class="georgia_pt">Der Text wird in der Schriftgröße 15pt  

    angezeigt.</p>  

<p class="georgia_px">Der Text wird in der Schriftgröße 16px  

    angezeigt.</p>  

<p class="georgia_mm">Der Text wird in der Schriftgröße 7.5mm  

    angezeigt.</p>  

<p class="georgia_pc">Der Text wird in der Schriftgröße 2pc  

    angezeigt.</p>  

④ <p style="font-size: small">  

    Der Text wird klein (small) angezeigt.  

⑤ <span style="font-size: larger">Der Text wird größer  

    (larger) angezeigt.</span></p>  

</body>  

</html>

```

- ① Für alle Elemente `p` legen Sie die Schriftart fest. Da Sie keine Schriftgröße angeben, verwendet der Browser die Standardgröße.
- ② Den nächsten vier Klassen weisen Sie über die Eigenschaft `font-size` verschiedene Schriftgrößen zu. Dabei verwenden Sie unterschiedliche Maßeinheiten.
- ③ Den verschiedenen Absätzen werden über das Attribut `class` die unterschiedlichen Formatklassen zugewiesen, sodass deren Inhalte in verschiedenen Größen angezeigt werden.
- ④ Diesem Absatz weisen Sie per Inline-Style die Schriftgröße `small` zu.
Damit wird der Text in einer kleinen Schriftgröße angezeigt, die von den Browsereinstellungen abhängt.
- ⑤ Mit der Angabe `larger` erhöhen Sie die Schriftgröße ausgehend vom übergeordneten Element `p` um eine Stufe. Der Text wird damit in der Schriftgröße `medium` angezeigt.

Schriftgröße verändern mit font-size

Der Text wird in der Standardgröße angezeigt.

Der Text wird in der Schriftgröße 15pt angezeigt.

Der Text wird in der Schriftgröße 16px angezeigt.

Der Text wird in der Schriftgröße 7.5mm angezeigt.

Der Text wird in der Schriftgröße 2pc angezeigt.

Der Text wird klein (small) angezeigt. Der Text wird größer (larger) angezeigt.

Anzeige verschiedener Schriftgrößen (kap03\font-size.htm)



Wenn Sie nachmessen, werden Sie feststellen, dass die Angaben in `mm` nicht korrekt sind – aus wie vielen Pixeln sich ein `mm` zusammensetzt, hängt von Größe und Auflösung des verwendeten Bildschirmes ab. Da `pt` und `pc` ebenfalls auf Millimeter (beziehungsweise Inch) beruhen, gilt für diese Maßeinheiten dasselbe. **Die Maßeinheiten `pc`, `pt`, `mm` und `cm` lassen sich somit nicht in Pixel umrechnen.**

Angaben in Punkt und Pica dienen der Druckdarstellung, Angaben in Pixel sind für die Bildschirmdarstellung geeignet. Weil `em` und `%` relative Maßeinheiten sind, können diese für beide Medien sinnvoll verwendet werden.

Für Schriften macht die Verwendung von Pixelangaben wenig Sinn. Schriften sollen in aller Regel proportional zueinander sein. Daher sind die Einheiten `em` und `rem` in aller Regel am besten geeignet, um Schriftgrößen anzugeben. Schlüsselwörter wie `small` oder `larger` erlauben auch eine proportionale Anpassung der Schriftgröße, allerdings können Sie mit den Kombinationen aus Zahlenwerten und Einheiten die Schriftgrößen detaillierter angeben.

3.5 Eigenschaften vererben oder zurücksetzen

Schriftformate werden in der Regel **vererbt**, dies bedeutet, dass untergeordnete Elemente Eigenschaften der übergeordneten Elemente übernehmen.

Beispiel: *kap03\vererbung.htm*

Einige Vererbungsmöglichkeiten werden im folgenden Dokument aufgezeigt.

```

<html lang="de">
<head>
<title>Vererbung</title>
<style>
① body { font-family: Georgia, "Times New Roman", serif; }
② div { font-size: .75rem; }
③ p { color: green; }
</style>
</head>
④ <body>
    <h1>Normale Vererbung</h1>
    <p>Dieser Text wird in Georgia, Standardgröße und grüner
       Schriftfarbe dargestellt.</p>
    ⑤ <div>Dieser Text wird in Georgia, 12px dargestellt,
        ⑥ <p>wohingegen dieser Absatz grün wird.</p></div>
</body>
</html>
```

- ① Alle Texte innerhalb des Elementes `body` und in allen untergeordneten Elementen sollen in der Schriftart Georgia dargestellt werden.
- ② Texte innerhalb von `div` werden auf $\frac{3}{4}$ der Größe anderer Schriften verkleinert (in diesem Fall $\frac{3}{4}$ von 16 Pixel Ausgangsgröße, also 12 Pixel).
- ③ Die Inhalte der Absätze erhalten die Schriftfarbe grün.
- ④ Das Element `h3` wurde nicht ausdrücklich formatiert. Da das Element `h3` jedoch ein untergeordnetes Element von `body` ist, erbt es die Schriftart Georgia.
- ⑤ Auch Absätze sind Nachfahren von `body`. Sie erben deshalb ebenfalls die Schriftart. Zusätzlich werden sie in grüner Schriftfarbe dargestellt.
- ⑥ Auch `div` erbt die Schriftart von `body`. Außerdem wurde für `div` die Schriftgröße auf $.75em$ festgelegt. In das `div` wird ein `p` eingesetzt. Der Absatz erbt die Formate von `body` und `div`.

Normale Vererbung

Dieser Text wird in Georgia, Standardgröße und grüner Schriftfarbe dargestellt.

Dieser Text wird in Georgia, 12px dargestellt, wohingegen dieser Absatz grün wird.

Der Wert **inherit**

Manchmal möchte man, dass ein Kind eine Eigenschaft vom Elterelement übernimmt, die nicht automatisch vererbt wird. Sie können explizit zur Übernahme solcher Formate auffordern.

Beispiel

```
<style>
  div { font-family: Georgia, "Times New Roman", Times, serif;
}
  p { font-family: Verdana, Arial, Helvetica, sans-serif; }
  div p { font-size: 0.75em; }
</style>
```

In diesem Beispiel würde der Inhalt des Absatzes `p` innerhalb eines `div`-Elementes aufgrund der Vererbung in der Schriftart Georgia dargestellt. Da für Absätze explizit die Schriftart Verdana angegeben ist, wird diese auch verwendet. Sie können dem Browser mitteilen, dass er für Absätze innerhalb eines `div` die Schriftart des Elterelements verwenden soll. Das erreichen Sie mit dem Wert `inherit`.

```
<style type="text/css">
  div { font-family: Georgia, "Times New Roman", Times, serif;
}
  p { font-family: Verdana, Arial, Helvetica, sans-serif; }
  div p { font-family: inherit; }
</style>
```

Der Wert `inherit` lässt sich jeder Stylesheet-Eigenschaft zuweisen und besagt, dass der entsprechende Wert aus dem Elterelement übernommen werden soll.

Normale Vererbung

Schrift ist Verdana;
Schrift ist Georgia;
Schrift ist Georgia;
Schrift ist Verdana, wie für `p` angegeben

Normale Vererbung

Vererbung mit `inherit`

Schrift ist Verdana;
Schrift ist Georgia;
Schrift ist Georgia;
auch diese Schrift ist Georgia, da vom Eltern-Element `div` geerbt;

Vererbung mit Wert `inherit`
([kap03/vererbung-inherit.htm](#))

Der Wert **initial**

Das Schlüsselwort `initial` wird genau wie `inherit` verwendet. Dabei werden zuvor gemachte Änderungen auf den Standardwert zurückgesetzt.

Vererbung von Schriftgrößen aufheben

Für eine optimale Ausgabe auf unterschiedlichen Endgeräten wird empfohlen, Schriftgrößen in **relativen Einheiten** anzugeben. In der Praxis haben sich `em` und `rem` (root em) durchgesetzt. Hierbei ist wichtig, die Vererbbarkeit von `font-size` zu berücksichtigen.

Wenn Sie die Schriftgrößen für Fließtext in Absätzen auf 12 Pixel festlegen möchten, notieren Sie in Ihrem CSS die folgende Regel (vorausgesetzt die Schriftgröße für das HTML-Element ist auf seinem Standardwert von 16 Pixel):

```
p {  
    font-size: .75rem; /* 16px Standardgröße mal .75 sind 12px */  
}
```

Dadurch werden Texte in Absätzen kleiner als andere Texte dargestellt, zum Beispiel in Listen. Um für Texte in Listen ebenfalls eine Schriftgröße von 12 Pixeln zu erreichen, empfiehlt es sich, für Listeneinträge dieselbe Schriftgröße anzugeben:

```
p, li { /* Der Regel wird der Selektor li hinzugefügt, durch  
        Komma von p getrennt */  
    font-size: 0.75em;  
}
```

Gilt in einer Liste eine Schriftgröße von `16px` (was `1em` entspricht) und gibt man für `li` eine Größe von `0.75em` an, so beträgt die vom Browser berechnete Schriftgröße in den Listeneinträgen `12px`. Enthält dieser Listeneintrag eine weitere Liste, gilt für diese ebenfalls die Schriftgröße des Elternelementes, also auch 12 Pixel. Die enthaltenen `li` werden – wie für alle `li` festgelegt – mit dem Faktor `0,75` multipliziert, sodass für die Texte in den untergeordneten Listen eine Schriftgröße von `9px` verwendet wird.



Mit jeder Verschachtelungsebene wird die Schrift kleiner. Dies ist in der Regel nicht gewünscht und kann zur völligen Unlesbarkeit der Texte führen. Daher sollte man die Vererbung unterbrechen. Fügen Sie dem Stylesheet eine weitere Regel hinzu:

```
p, li {  
    font-size: 0.75em;  
}  
  
li * { /* Alle Nachfahren von li */  
    font-size: inherit;  
}
```

Mit dieser Regel wird dem Browser mitgeteilt, dass jeder Nachfahre von `li` dieselbe Schriftgröße verwenden soll wie das `li` selbst. So stellen Sie sicher, dass Texte in verschachtelten Listen leserlich bleiben.

Alternativ führt ein `font-size: 100%` zu demselben Ergebnis.

Mit dieser Technik wird die Vererbung nicht wirklich unterbrochen. Damit werden andere Werte vererbt, nämlich 100 % oder die Schriftgröße des Elternelementes.

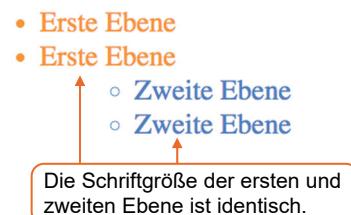
Beispiel: *kap03\vererbung_unterbrechen.htm*

```

<html lang="de">
<head>
<title>Vererbung unterbrechen</title>
<style>
① p, li {
    font-size: 0.75em;
}
② li * {
    font-size: 100%; /* Verhindert die weitere Verkleinerung */
}
</style>
</head>
<body>
    <ul>
        ③ <li>Erste Ebene</li>
        <li>
            Erste Ebene
            ④ <ul>
                <li>Zweite Ebene</li>
                <li>Zweite Ebene</li>
            </ul>
        </li>
    </ul>
</body>
</html>

```

- ① Die Schriftgröße in Absätzen und Listeneinträgen wird einheitlich auf 75% des Elterelementes festgelegt.
- ② Für alle Nachfahren (Kinder und tiefer verschachtelte Elemente) wird bestimmt, dass die Schriftgröße der des Elterelementes entsprechen soll. Die Schrift wird also nicht mit zunehmender Verschachtelungstiefe weiter verkleinert.
- ③ Elemente der ersten Ebene (orange), die auf 75% verkleinert werden sollen.
- ④ Nachfahren von `li` (werden nicht weiter verkleinert).



3.6 Angabe der Schrift in Kurzform

Anstatt jede Schrifteigenschaft einzeln anzugeben, können Sie die Kurzform verwenden.

```
font: [font-style font-variant font-weight] font-size font-family;
```

Die Reihenfolge der Angaben ist vorgeschrieben. Sie lautet: Schriftstil, Kapälchen, Stärke, Größe und Schriftart. Die Werte werden durch ein Leerzeichen voneinander getrennt. Angaben mit einem Leerzeichen im Wert sind in Anführungszeichen zu setzen.

Bei der Angabe der Eigenschaften müssen zwingend die Größe und die Schriftfamilie angegeben werden. Die drei anderen Eigenschaften `font-style`, `font-variant` und `font-weight` sind optionale Angaben, bei denen Standardwerte verwendet werden, wenn sie nicht angegeben sind. Bei diesen drei Eigenschaften ist die Reihenfolge nicht von Bedeutung. Die Browser weisen die Werte automatisch den entsprechenden CSS-Eigenschaften zu.

Beispiele

```
h1 { font: bold 1.25em Verdana, Arial, helvetica, sans-serif; }
```

Die Überschrift wird folgendermaßen dargestellt: `font-weight: bold`; `font-size: 1.25em`; `font-family: Verdana, Arial, helvetica, sans-serif`. Der Wert `normal` wird automatisch den nicht angegebenen Eigenschaften `font-style` und `font-variant` zugewiesen.

```
p { font: 0.75em Verdana, Arial, helvetica, sans-serif; }
```

Der Absatz besitzt folgende Einstellungen: `font-style: normal`, `font-variant: normal`, `font-weight: normal`, `font-size: 0.75em`, `font-family: Verdana, Arial, helvetica, sans-serif`.

```
li { font: oblique bold 0.75em }
```

Die Listeneinträge erhalten folgende Darstellung: `font-style: oblique`, `font-variant: normal`, `font-weight: bold`, `font-size: 0.75em`. Für die Schriftart wird die Standardschrift oder eine geerbte Schrift eingesetzt.

Sie können zusätzlich in der gekürzten Schriftangabe neben der Schriftgröße die Zeilenhöhe angeben.

```
font: font-style font-variant font-weight font-size/line-height  
font-family;
```

Die Höhe der Zeile, die standardmäßig 120 % der Schriftgröße beträgt, können Sie durch einen Schrägstrich getrennt direkt nach der Schriftgröße angeben. Die Zeilenhöhe wird in den bekannten Maßeinheiten angegeben werden.

Am einfachsten und sinnvollsten geben Sie die Zeilenhöhe ohne Angabe einer Maßeinheit in Relation zur Schriftgröße an:

```
p { font: 0.75em/1.5 Verdana, Arial, helvetica, sans-serif; }
```

Die Schriftgröße beträgt `0.75em`, die Zeilenhöhe das 1,5fache der Schriftgröße. Die vorzugsweise zu verwendende Schrift ist Verdana. Ist diese Schrift auf dem System des Besuchers nicht vorhanden, ist eine der nachfolgenden Schriften anzuwenden.

3.7 Übungen

Übung 1: Theoriefragen zu Schriftarten

Level		Zeit	ca. 15 min
Ergebnisdatei	<i>kap03\uebung1-2.doc</i>		

1. Erklären Sie, was Sie beim Einsatz von verschiedenen Schriftarten beachten müssen.
2. Erläutern Sie, welche Angaben über die Kurzform `font`: gemacht werden können. Was ist dabei zu beachten? Was passiert, wenn Sie bestimmte Eigenschaften nicht angeben?

Übung 2: Schriftformatierungen einer Webseite

Level		Zeit	ca. 30 min
Übungsdatei	<i>kap03\chalkidiki.txt</i>		
Ergebnisdatei	<i>kap03\uebung3.htm</i>		

1. Erstellen Sie aus der Textdatei *kap03\chalkidiki.txt* eine Webseite.
2. Verwenden Sie eine Serifen-Schriftart wie Georgia für den Fließtext. Die Größe der Schrift beträgt `12px` (entspricht `0.75em`). Die Überschrift soll hingegen in `24px` dargestellt werden. Bestimmte Wörter wie Eigennamen werden über einen Klassen-Selektor hervorgehoben. In diesem Selektor ist eine kursive und fette Formatierung hinterlegt.

Die Geschichte von Chalkidiki

Die **Chalkidiki** wurde im 7. und 8. Jh. v. Chr. kolonisiert. Die ersten Siedler, von denen die Chalkidiki ihren Namen erhielt, stammten aus der Stadt **Chalkis** auf der **Insel Eboa**. Zu jener Zeit entstehen überall in Griechenland voneinander unabhängige und oft untereinander verfeindete Stadtstaaten.

Nach zahlreichen leidvollen Kriegen wird im 5. Jh. v. Chr. der "Chalkidische Bund" geschlossen, um militärisch, politisch und wirtschaftlich eng zusammenzuarbeiten. Mitte des 4. Jh. v. Chr. wird die Chalkidiki dennoch von seinen starken Nachbarn, den staatlich geeinten und straff organisierten **Makedonen**, unterworfen. Unter König Philipp II. und später seinem Sohn Alexander dem Großen entsteht erstmals ein geeintes Griechenland. Allerdings ist Griechenland nur noch ein kleiner, relativ unbedeutender Teil im mächtigen Reich von Alexander dem Großen.

In der römischen Zeit wird Makedonien erobert und Chalkidiki wird durch Überfälle der **Goten** im Jahr 269 weitgehend zerstört und entvölkert. Ende des 4. Jh. fällt ganz Griechenland an **Ostrom**. In den folgenden Jahrhunderten der oströmisch-byzantinischen Zeit werden die Städte der Chalkidiki immer wieder Opfer von Plünderei und Zerstörung.

Mitte des 15. Jh. wird Griechenland, und somit auch die Chalkidiki, von den **Türken** erobert. Das Byzantinische Reich ist 1453 endgültig zerstört. Im 16. Jh. führen die Türken auf der Chalkidiki die Seidenraupenzucht und den Tabakanbau ein. Die Region erlebt einen gewaltigen wirtschaftlichen Aufschwung.

Nach Aufständen der Griechen gegen die türkische Fremdherrschaft wird Griechenland 1830 ein **unabhängiger Staat**. Aufstände auf der Chalkidiki werden aber niedergeschlagen. 1832 wird ein **Bayer** zum ersten König Griechenlands ernannt. Die Chalkidiki, viele Inseln und Nordgriechenland bleiben unter türkischer Herrschaft.

Im ersten Balkankrieg 1912-1913 verbünden sich Griechenland, Bulgarien, Serbien und Montenegro gegen das **Osmannische Reich**. Im Oktober 1912 wird die Chalkidiki von der türkischen Herrschaft befreit und an die Griechen übergeben.

1922 versuchte Griechenland, die Siedlungsgebiete an der östlichen Ägäisküste, in Ostthrakien und Konstantinopel zu besetzen. Mit Zustimmung Englands und Frankreichs marschierten griechische Truppen Richtung **Ankara**. Nachdem die Alliierten ihre Waffenlieferungen eingestellt hatten, werden die Griechen von den **Türken** unter Kemal Atatürk vernichtend geschlagen. Im anschließenden Friedensvertrag von Lausanne 1923 wird ein groß

*Text in einer angenehmen, lesbaren Form (*kap03\uebung3.htm*)*

4

Texte gestalten

4.1 CSS-Eigenschaften zur Textgestaltung

Die Gestaltung der Schrift beeinflusst maßgeblich die Lesbarkeit und damit die Verständlichkeit Ihrer Texte und unterstützt ein einheitliches und stimmiges Gesamtbild. Wenn Sie zu viele unterschiedliche Schriften einsetzen, kann eine Webseite unruhig und unordentlich wirken.

Die folgenden CSS-Eigenschaften dienen der Gestaltung von Texten:

- ✓ `text-align, vertical-align`
- ✓ `text-transform`
- ✓ `text-decoration`
- ✓ `line-height`
- ✓ `letter-spacing, word-spacing`
- ✓ `text-indent`
- ✓ `white-space`
- ✓ `text-shadow`

4.2 Text ausrichten

Horizontale Ausrichtung mit `text-align`

Hiermit steuern Sie die horizontale Ausrichtung innerhalb eines Elements.

`text-align: Wert;`

Werte	<ul style="list-style-type: none">✓ <code>left</code> für linksbündige Ausrichtung (Standard)✓ <code>right</code> für rechtsbündige Ausrichtung✓ <code>center</code> für zentrierte Ausrichtung✓ <code>justify</code> für Ausrichtung als Blocksatz
Standardwert	<code>left</code>
Vererbbar	Ja
Anwendbar auf	Alle Blockelemente

Diese Eigenschaft wirkt sich nur auf Blockelemente aus, beispielsweise `address, block-quote, dl, div, fieldset, form, h1 bis h6, hr, ol, p, pre, table und ul`.

Beispiel: *kap04\text-align.htm*

Drei aufeinanderfolgende Absätze werden rechtsbündig, linksbündig und im Blocksatz ausgerichtet. Zum leichteren Verständnis werden die Klassen entsprechend benannt.

```

<html lang="de">
<head>
    <title>text-align</title>
    <style>
        ① body      { font-family: Georgia,Times,
                        "Times New Roman", serif;
                        text-align: justify; }

        ② h1       { text-align: center; }

        ③ .rechts   { text-align: right; }
        .links     { text-align: left; }

    </style>
</head>
<body>
    <h1>Änderung der Eigenschaft "text-align" (Texte ausrichten)</h1>
    ④ <p class="rechts">Die Angabe von <b>text-align:right</b> bewirkt eine rechtsbündige Ausrichtung.</p>
    <p class="links">Standardmäßig wird Text linksbündig ausgerichtet: <b>text-align:left</b>.</p>
    ⑤ <p>Blocksatz wird folgendermaßen festgelegt: <b>text-align:justify</b>. Da Sie das Ergebnis am besten bei vielen Zeilen sehen können, folgt ein platzfüllender Blindtext ohne weitere Bedeutung: [...]</p>
</body>
</html>

```

- ① Für body geben Sie Blocksatz an. Dieser wird an alle eingeschlossenen Elemente (Nachfahren) weitervererbt.
- ② Überschriften h1 werden zentriert.
- ③ Elemente, denen die Klassen rechts und links zugewiesen werden, werden entsprechend horizontal ausgerichtet.
- ④ Nach der zentrierten Überschrift wird der Inhalt des ersten Absatzes rechtsbündig, der Inhalt des zweiten Absatzes linksbündig ausgerichtet.
- ⑤ Der letzte Absatz besitzt keine Klasse. Er wird so dargestellt wie für den gesamten body in ① festgelegt. Der Text ist dank text-align: justify links **und** rechts bündig (Blocksatz).

Änderung der Eigenschaft "text-align" (Texte ausrichten)

Die Angabe von **text-align:right** bewirkt eine rechtsbündige Ausrichtung.

Standardmäßig wird Text linksbündig ausgerichtet: **text-align:left**.

Blocksatz wird folgendermaßen festgelegt: **text-align:justify**. Da Sie das Ergebnis am besten bei vielen Zeilen sehen können, folgt ein platzfüllender Blindtext ohne weitere Bedeutung: Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.

Unterschiedliche Ausrichtungsarten

Nachteile von Blocksatz

Aufgrund folgender Nachteile verschlechtert Blocksatz die Lesbarkeit von Texten:

- ✓ Es fällt schwerer, die nächste Zeile zu finden, besonders bei langen Zeilen.
- ✓ Durch unterschiedliche Wortabstände können irritierende oder ablenkende Muster entstehen, besonders bei kurzen Zeilen.
- ✓ Der Versuch, durch Änderung der Zeilenlänge eines der Probleme zu lösen, erschwert das jeweils andere Problem. Oft wird dann auf (automatische) Silbentrennung zurückgegriffen. Auch diese erschwert die Lesbarkeit und erhöht die Ablenkung – besonders, wenn ein Wort wie Urin-stinkt (Ur-Instinkt) getrennt wird, das durch die Trennung an der falschen Stelle eine andere Bedeutung erhalten kann.

Vertikale Ausrichtung mit `vertical-align`

Hiermit legen Sie die vertikale Ausrichtung innerhalb eines Inline-Elementes fest, also wie sich kleiner Text in einer Zeile zu größerem Text in derselben Zeile verhalten soll oder wie Text neben einem Bild dargestellt wird.

Außerdem können tabellarische Daten innerhalb einer Tabellenzeile vertikal ausgerichtet werden.

Mit `vertical-align` sind keine Ausrichtungen von Block-Level-Elementen (z. B. `div`) möglich. Um Elemente innerhalb eines Elternelementes zueinander zu positionieren, benutzen Sie `display: flex` oder `display: grid` (siehe Kapitel 12).

<code>vertical-align: Wert;</code>

Mögliche Werte für diese Eigenschaft sind:

Werte	<ul style="list-style-type: none"> ✓ <code>top</code> = richtet alle Elemente am oberen Rand aus ✓ <code>bottom</code> = richtet alle Elemente am unteren Rand aus ✓ <code>text-top</code> = Ausrichtung am oberen Rand des Textes ✓ <code>text-bottom</code> = Ausrichtung am unteren Rand des Textes ✓ <code>baseline</code> = richtet alle Elemente an der Grundlinie aus ✓ <code>middle</code> = standardmäßige mittlere Ausrichtung ✓ <code>sub</code> = das Element wird tiefgestellt ✓ <code>super</code> = das Element wird hochgestellt ✓ Prozentangabe
Standardwert	<code>baseline</code>
Vererbbar	Nein
Anwendbar auf	Inline-Elemente, wie z. B. <code>a</code> , <code>acronym</code> , <code>br</code> , <code>em</code> , <code>img</code> , <code>input</code> , <code>small</code> , <code>span</code> , <code>strike</code> , <code>strong</code> , <code>textarea</code> ; zusätzlich auf <code>td</code>

Negative Angaben verschieben Elemente unter die Grundlinie.

Beispiel: *kap04\vertical-align.htm*

Für die Anzeige der Ausrichtungsarten müssen kleine Elemente an einem großen Element ausgerichtet werden. Dazu erstellen Sie einen Absatz, dessen Inhalt unterschiedliche Schriftgrößen aufweist.

```
<html lang="de">
<head>
    <title>vertical-align</title>
    <style>
        body, td { font: .75rem Georgia,Times,"Times New Roman",
            serif; }
        table {
            width: 100%;
            min-height: 100px;
            border-collapse: collapse;
        }
        td {
            border: 3px solid #333;
        }
    </style>
</head>
<body>
    <h1>Ausrichtungen mit vertical-align</h1>
    ① <p style="font-size: 2.25rem;">Baseline, normal
    ②   <span style="font-size: .75rem;">
        ③     <span style="vertical-align:top;">top</span>
        <span style="vertical-align:middle;">middle</span>
        <span style="vertical-align:bottom;">bottom</span>
        <span style="vertical-align:-10%;">-10%</span>
        <span style="vertical-align:75%;">75%</span>
    </span></p>
    <p>Es folgt ein hochgestellter Text:
    ④       <span style="vertical-align:super;">vertical-align:
            super.</span></p>
    <p>Es folgt ein tiefgestellter Text:
    ⑤       <span style="vertical-align:sub;">vertical-align:
            sub.</span></p>

    <table>
        <tr>
```

```
(6) <td style="vertical-align:top;">Dieser Text ist in der  
     Zelle oben ausgerichtet.</td>  
     <td style="vertical-align:bottom;">Dieser Text ist in  
     der Zelle unten ausgerichtet.</td>  
</tr>  
</table>  
</body>  
</html>
```

- ① Der Inhalt des Absatzes wird auf 225 % der Standardschriftgröße vergrößert.
- ② Die von span umfassten Inhalte werden auf ¼ der Standard-Schriftgröße verkleinert.
- ③ Durch die unterschiedlichen Schriftgrößen in einem Absatz ist es möglich, die Inhalte einzeln vertikal auszurichten. Die Ausrichtung erfolgt dabei am vorangegangenen großen Text mit einer entsprechenden Höhe. An dieser Höhe kann der kleiner formatierte Inhalt oben, mittig und unten ausgerichtet werden. Auch Prozentangaben sind möglich.
- ④ Dieser Text wird hochgestellt.
- ⑤ Hier wird Text tiefgestellt.
- ⑥ Durch die vorgegebene Mindesthöhe von 100 Pixeln werden die Zellen nicht voll ausgefüllt. Die Angabe der CSS-Eigenschaft ermöglicht somit das unterschiedliche Ausrichten des Textes in den Zellen.

Ausrichtungen mit vertical-align

Baseline, normal top middle bottom -10% 75%

Es folgt ein hochgestellter Text: vertical-align:super.

Es folgt ein tiefgestellter Text: vertical-align:sub.

Dieser Text ist in der Zelle oben ausgerichtet.	Dieser Text ist in der Zelle unten ausgerichtet.
---	--

Unterschiedliche vertikale Ausrichtungsarten

4.3 Groß- und Kleinschreibung beeinflussen

Mit der Eigenschaft `text-transform` machen Sie Angaben zur Groß- und Kleinschreibung von ganzen Wörtern oder den Wortanfängen.

text-transform: Wert;

Werte	<ul style="list-style-type: none"> ✓ uppercase = Großschreibung ✓ lowercase = Kleinschreibung ✓ capitalize = erster Buchstabe eines jeden Wortes ist groß ✓ none = ohne Umwandlung (wie im Quelltext)
Standardwert	None
Vererbbar	Ja
Anwendbar auf	Alle Elemente

Beispiel: kap04\text-transform.htm

Im Folgenden werden mehrere Absätze erstellt, denen über das HTML-Attribut style verschiedene Werte für die Groß- und Kleinschreibung zugewiesen werden.

```
<body>
    <h1>Änderung der Eigenschaft "text-transform" (Groß- und
    Kleinschreibung)</h1>
    ① <p style="text-transform:none">Normal formatierter Text mit
        „ß“.</p>
    ② <p style="text-transform:uppercase">Dieser Text besteht
        komplett aus Großbuchstaben mit „ß“.</p>
    ③ <p style="text-transform:lowercase">Im Gegensatz dazu ist
        dieser Text kleingeschrieben mit „ß“.</p>
    ④ <p style="text-transform:capitalize">Zum Schluss werden alle
        Wortanfänge großgeschrieben mit „ß“.</p>
</body>
```

- ① In diesem Absatz erhält die CSS-Eigenschaft den Wert none, damit keine Umwandlung vorgenommen wird. Die Angabe der Eigenschaft könnten Sie auch weglassen, da dies der Standardwert ist.
- ② Mit uppercase wird der Text in Großbuchstaben gesetzt.
- ③ Hier wird mit lowercase der Text kleingeschrieben.
- ④ Das Attribut capitalize erzwingt die Großschreibung des ersten Buchstabens eines jeden Wortes.

Änderung der Eigenschaft "text-transform" (Groß- und Kleinschreibung)

Normal formatierter Text.
DIESER TEXT BESTEHT KOMPLETT AUS GROSSBUCHSTABEN MIT „SS“.
im gegensatz dazu ist dieser text kleingeschrieben mit „ß“.
Zum Schluss Werden Alle Wortanfänge Großgeschrieben.

Groß- und Kleinschreibung mit „ss“

Seit dem 29. Juni 2017 ist das ß in der Großschreibung Bestandteil der amtlichen deutschen Rechtschreibung.

Dennoch wird der Buchstabe **ß** in allen modernen Browsern bei der Großschreibung automatisch in die Zeichenfolge **S S** umgewandelt.

Siehe auch https://de.wikipedia.org/wiki/Großes_ß

4.4 Text hervorheben mit `text-decoration` und `text-shadow`

Die Eigenschaft `text-decoration`

Mithilfe der Eigenschaft `text-decoration` können Sie Textpassagen besonders hervorheben. Sie können unterstrichen oder durchgestrichen dargestellt werden.

```
text-decoration: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ <code>underline</code> = unterstrichen ✓ <code>overline</code> = Linie über dem Element ✓ <code>line-through</code> = durchgestrichen ✓ <code>none</code> = normale Schreibweise
Standardwert	<code>none</code> (<code>underline</code>)
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Die Werte können Sie auch gleichzeitig angeben, sodass Sie ein Element unterstrichen und zusätzlich mit einer darüber befindlichen Linie formatieren können.

Vermeiden Sie es, die CSS-Eigenschaft `text-decoration` mit dem Wert `underline` zu nutzen. Da **standardmäßig Hyperlinks unterstrichen** dargestellt werden, erwartet der Besucher bei einem unterstrichenen Text eine Verlinkung. Setzen Sie `line-through` ein, wenn Sie z. B. auf veraltete Informationen hinweisen möchten, ohne diese endgültig zu löschen.

Beispiel: `kap04\text-decoration.htm`

Es werden mehrere Absätze erstellt, denen verschiedene Werte für die CSS-Eigenschaft `text-decoration` übergeben werden. Der Text wird dadurch unterschiedlich formatiert.

①	<pre><body> <h1>Elemente hervorheben: text-decoration</h1> <p style="text-decoration: none">Normal formatierter Text.</p></pre>
②	<pre><p style="text-decoration: underline">Dieser Text wird unterstrichen.</p></pre>
③	<pre><p style="text-decoration: overline underline">Der Text hat eine obere Linie und wird unterstrichen.</p></pre>
④	<pre><p style="text-decoration: line-through">Der Text wird durchgestrichen.</p></pre>
	<pre></body></pre>

- ① In diesem Absatz erhält die CSS-Eigenschaft den Wert `none`. Die Angabe könnten Sie auch weglassen, da dies automatisch der Standardwert ist.
- ② Hier wird der Wert `underline` zugewiesen, der so wenig wie möglich eingesetzt werden sollte.
- ③ Einer gleichzeitigen Anwendung mit einem anderen Wert steht dagegen nichts im Wege.
- ④ Der Inhalt dieses Absatzes wird durchgestrichen dargestellt.

Die CSS-Eigenschaft `text-decoration` wird beispielsweise eingesetzt, um die Unterstreichungen von Hyperlinks in Menüs abzuschalten. Beim Antabben (`:focus`) und Überfahren mit der Maus (`:hover`) kann der Link dann wieder unterstrichen werden.

Beispiel: `kap04\text-decoration-link.htm`

Eine mögliche Formatierung der Hyperlinks könnte beispielsweise so aussehen:

```
<style>
  a.anders:link    { text-decoration: none; }
  a.anders:visited { text-decoration: line-through; }
  a.anders:active, a.anders:hover , a.anders:focus {
    text-decoration: underline;
  }
</style>
```

Die Hyperlinks werden bei Angabe der entsprechenden Klasse `anders` von der normalen Darstellung abweichend formatiert. Bei der Formatierung über die CSS-Klasse `anders` werden die Hyperlinks beispielsweise ohne Unterstreichung dargestellt. Erst wenn Sie den Cursor über einen Verweis führen, wird der Hyperlink unterstrichen. Diese Formatierung kann z. B. wie folgt durchgeführt werden.

```
<p>Die normale Darstellung:<br>
  <a href="#">ein besuchter Link</a> -
  <a href="xyz.htm">ein unbesuchter Link</a></p>
<p>Die spezielle Darstellung:<br>
  <a class="anders" href="#">ein besuchter Link</a> -
  <a class="anders" href="xyz.htm">ein unbesuchter Link</a></p>
```

! Links heben sich standardmäßig durch zwei Eigenschaften von normalem Text ab: erstens ihre blaue Farbe und zweitens die Unterstreichung. Diese sogenannte Mehrfachkennzeichnung sorgt dafür, dass Links auch erkannt werden können, wenn ein Dokument auf einem Schwarz-Weiß-Drucker oder -Bildschirm ausgegeben wird, wenn hohe Kontraste oder eigene Farben vom Seitenbesucher verwendet werden oder wenn der Besucher farbenblind ist. Wenn Sie die Unterstreichung entfernen, sollten Sie also durch Hinzufügen eines anderen Merkmals (z. B. ein Icon) für die Erkennbarkeit von Links sorgen.

Effekte mit `text-shadow`

```
text-shadow: Wert Wert Wert Wert;
```

Die Eigenschaft `text-shadow` stellt eine Möglichkeit bereit, Texte mit Schatten auszustatten. Ihr können vier Werte mitgegeben werden. Diese stehen (in dieser Reihenfolge) für die horizontale und vertikale Verschiebung zum Text, für einen Übergang zur Hintergrundfarbe und für den Farbwert der Farbe des Schattens.

```
text-shadow: h-shadow v-shadow blur color;
```

Beispiel: [kap04\text-shadow.htm](#)

```
<style>
P { text-shadow: 1px 2px 3px #333; }
</style>
```

Dieser Code sorgt für einen Textschatten, der horizontal um 1 Pixel nach rechts und 2 Pixel nach unten verschoben ist. Es handelt sich um einen weichen Schatten. Der Übergang beträgt 3 Pixel. Die Farbe des Schattens ist ein dunkles Grau.

Text mit Schatten

Text mit weichem Schlagschatten

4.5 Textabstand

Zeilenhöhe

Die **Zeilenhöhe** ① ist die Distanz zwischen der Unterkante der Vorzeile und der Unterkante der Folgezeile. Mithilfe der Stylesheet-Eigenschaft `line-height` können Sie diesen Abstand in einem HTML-Dokument beeinflussen.

Der Browser verändert daraufhin den Abstand zwischen den beiden Zeilen. Bekommt die Zeilenhöhe einen Wert größer als 120 %, führt das optisch zu einem zusätzlichen Durchschuss ② (typografischer Ausdruck).

```
line-height: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ <code>normal</code> = in der Schrift definierte Zeilenhöhe (ca. 120 % Zeilenhöhe) ✓ Zahlenangaben mit entsprechender Maßeinheit (z. B. <code>em</code>) oder ohne Maßeinheit als Relation zur Textgröße ✓ Prozentuale Angaben
Standardwert	<code>normal</code>
Vererbbar	Auf alle Unterelemente
Anwendbar auf	Alle Elemente

Die Zeilenhöhe können Sie in den üblichen Maßeinheiten angeben. Dabei sollte der Wert mindestens 120 % der Schriftgröße besitzen, um das Lesen des Textes zu vereinfachen. Dies bedeutet, dass bei einer Schriftgröße von 12 Pixeln eine Zeilenhöhe von 15 Pixeln oder 120 % der Mindestanforderung genügt. Als ideal gilt das 1,4- bis 1,5-Fache der Zeilenhöhe, was aber auch von der Zeilenlänge abhängt. Die Angabe `line-height: normal` setzt den Wert wieder in den ursprünglichen Ausgangszustand.

- ✓ Je größer die Zeilenbreite ist, umso größer sollte die Zeilenhöhe werden. Das ermöglicht dem Leser ein leichteres Halten der Lesezeile.
- ✓ In den fetteren Versionen der Schriften sollten Sie die Zeilenhöhe gegenüber der normalen Schrift mindestens auf 130–140 % vergrößern.
- ✓ Ist die Zeilenhöhe kleiner als die Schrift, überschneiden sich die Zeilen und der Text wird unleserlich.

Beispiel: `kap04\line-height.htm`

Die Zeilenhöhe der verschiedenen Textabsätze wird über die CSS-Eigenschaft `line-height` und unterschiedliche Maßeinheiten variiert. Damit der Unterschied ersichtlich ist, enthält jeder Absatz den gleichen Text.

```
<html lang="de">
<head>
    <title>Zeilenhöhen beeinflussen</title>
    <style>
        ① body { font: 1em Georgia,Times,"Times New Roman",serif; }
        ② .weit { line-height: 150%; }
        ③ .eng { line-height: 80%; }

        ...
    </style>
</head>
<body>
    <h1>Zeilenhöhen, variabler geht's kaum noch.</h1>
    ④ <p>Fortune plango vulnera stilantibus ocellis, quod
        sua...</p>
    ⑤ <p class="weit">Fortune plango vulnera stilantibus ocellis,
        quod sua...</p>
    ⑥ <p class="eng">Fortune plango vulnera stilantibus ocellis,
        quod sua...</p>
</body>
</html>
```

- ① Grundsätzlich werden alle Inhalte der Webseite in der Standardgröße und der Schriftart Georgia dargestellt. Die Zeilenhöhe wird über den Wert `normal` auf den Standardwert festgelegt.
- ② Absätze, die über die Klasse `weit` formatiert werden, sollen eine Zeilenhöhe von 150 % erhalten. Sie passen sich damit auch automatisch an die Schriftgröße an, wenn Sie diese einmal ändern sollten.

- ③ Über die Klasse `eng` legen Sie die Zeilenhöhe auf nur 15 Pixel fest, was auch bei veränderten Schriftgrößen beibehalten wird.
- ④ Dadurch, dass kein Klassen-Selektor angegeben ist, wird die Einstellung von `body` übernommen, sodass der Text mit der normalen Zeilenhöhe dargestellt wird.
- ⑤ Im zweiten Absatz ist der Text recht gut lesbar. Durch die größere Zeilenhöhe kann das menschliche Auge den Wechsel vom Ende einer Zeile zum Anfang der nächsten Zeile besser durchführen. Besonders bei breiten Absätzen machen sich die höheren Zeilen positiv bemerkbar.
- ⑥ Die Zeilenhöhe wurde so weit verringert, dass der Text nur noch schwer lesbar ist.

Zeilenhöhen, variabler geht's kaum.

1

Fortune plango vulnera stilantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata. In Fortune solo sederam elatus, prosperitatis vario flore coconatus; quicquid enim florui felix et beatus, nunc a summo corru gloria privatus. Fortune rota volvitur descendendo minoratus, alter in altum tollitur, nimis exaltatus rex sedet in vertice; caveat ruinam! Nam sub axe legimus hecubam redinam.

2

Fortune plango vulnera stilantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata. In Fortune solo sederam elatus, prosperitatis vario flore coconatus; quicquid enim florui felix et beatus, nunc a summo corru gloria privatus. Fortune rota volvitur descendendo minoratus, alter in altum tollitur, nimis exaltatus rex sedet in vertice; caveat ruinam! Nam sub axe legimus hecubam redinam.

3

Fortune plango vulnera stilantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata. In Fortune solo sederam elatus, prosperitatis vario flore coconatus; quicquid enim florui felix et beatus, nunc a summo corru gloria privatus. Fortune rota volvitur descendendo minoratus, alter in altum tollitur, nimis exaltatus rex sedet in vertice; caveat ruinam! Nam sub axe legimus hecubam redinam.

Unterschiedliche Zeilenhöhen in einem Dokument (kap04\line-height.htm)

Wortzwischenraum

Neben der Zeilenhöhe können Sie auch den Abstand zwischen einzelnen Wörtern verändern. Der Wortzwischenraum ist der nicht gedruckte Teil zwischen zwei Wörtern und wird auch Wortabstand bzw. in der Typografie Ausschluss genannt. Der optimale Abstand zwischen zwei Wörtern dient der Harmonie und der Lesbarkeit. Laut typografischen Vorgaben ist der ideale Wortzwischenraum die Breite, die der Großbuchstabe 1 einer Serifenschrift einnehmen würde.

`word-spacing: Wert;`

Werte	<ul style="list-style-type: none"> ✓ Normal ✓ Zahlenangaben mit entsprechender Maßeinheit; auch negative Werte sind möglich. ✓ Keine prozentualen Angaben!
Standardwert	Normal
Vererbbar	Ja
Anwendbar auf	Alle Elemente

Positive Werte vergrößern und negative Werte verkleinern den Abstand zwischen den einzelnen Wörtern. Die Angabe von `word-spacing: normal` setzt den Abstand auf den Standardwert zurück.

Beispiel: *kap04\word-spacing.htm*

In diesem Beispiel wird neben dem standardmäßigen Wortzwischenraum über die Klasse `plus` ein größerer und über die Klasse `minus` ein kleinerer Wortzwischenraum erzwungen.

```
<style>
  body { font: .75rem Georgia,Times,"Times New Roman",serif; }
  .plus { word-spacing: .25rem; }
  .minus { word-spacing: -.25rem; }
</style>
```

Wortzwischenraum

Dieser Absatz wird mit dem standardmäßigen Wortabstand angezeigt.

Der Abstand zwischen den einzelnen Wörtern ist immer gleich und beträgt in diesem Fall 5px.

Der Abstand zwischen den einzelnen Wörtern wurde über einen negativen Wert verkleinert.

*CSS-Eigenschaft `word-spacing` wird angewendet (*kap04\word-spacing.htm*)*

Zeichenabstand

Der freie Raum zwischen den einzelnen Zeichen wird auch als Buchstabenweite, Buchstabenabstand oder Laufweite bezeichnet. Dies ist eine flexible Größe, die je nach Schriftart, Schriftgröße und Schriftschnitt unterschiedlich ist und angepasst werden kann. Ein fetter Schriftschnitt benötigt, genau wie eine größere Schriftart, einen größeren Abstand zwischen den Buchstaben. Ansonsten würden die Buchstaben ab einer bestimmten Schriftgröße ineinanderfließen.

Mit der CSS-Eigenschaft `letter-spacing` können Sie die Abstände einzelner Buchstaben vergrößern (typografisch: sperren) oder verkleinern (typografisch: stauchen).

`letter-spacing:` Wert;

Werte	<ul style="list-style-type: none"> ✓ <code>normal</code> ✓ Zahlenangaben mit entsprechender Maßeinheit; standardmäßig wird die Maßeinheit Pixel (px) verwendet; auch negative Werte sind möglich. ✓ Keine prozentualen Angaben!
Standardwert	<code>normal</code>
Vererbbar	Ja
Anwendbar auf	Alle Elemente

Die positiven Werte vergrößern und die negativen Werte verkleinern den Zeichenabstand. Die Angabe von `letter-spacing: normal` setzt den Wert wieder auf den ursprünglichen Wert von 0 Pixel zurück.

Beispiel: *kap04\letter-spacing.htm*

Mit der folgenden Stylesheet-Angabe geben Sie die Buchstabenzwischenräume über verschiedene Maßeinheiten an. Dem jeweiligen Absatz weisen Sie das Format über den Selektor zu.

```
<style>
    body      { font: .75rem Helvetica,Arial,sans-serif; }
    .stauchen { letter-spacing: -0.0625rem; }
    .sperren1 { letter-spacing: 5px; }
    .sperren2 { letter-spacing: 1em; }
</style>
```

Zeichenabstand

`letter-spacing: normal;` - Dies ist der normale Zeichenabstand.

`letter-spacing: -0.2mm;` - Gestrauchter Text

`letter-spacing: 5px;` - Gesperrter Text

`letter-spacing: 1em;` - da passst ein
m d a z w i s c h e n |

Verschiedene Zeichenabstände (*kap04\letter-spacing.htm*)

4.6 Text einrücken

In manchen Büchern wird die erste Zeile eines Absatzes oder Kapitels eingerückt, was der besseren Lesbarkeit dient. In CSS ist dieses typografische Element als Eigenschaft implementiert.

<code>text-indent: Wert;</code>

Werte	Zahlen- und Prozentangaben mit entsprechender Maßeinheit; auch negative Werte sind möglich.
Standardwert	0
Vererbbar	Ja
Anwendbar auf	Alle Blockelemente

Pzentangaben beziehen sich auf die Gesamtbreite des jeweiligen Absatzes.

Beispiel: *kap04\text-indent.htm*

Drei Absätzen werden unterschiedliche Texteinzüge zugewiesen. Damit ein negativer Texteinzug möglich ist, wird der Seitenrand der Webseite erhöht.

```

<html lang="de">
<head>
    <title>Text einrücken</title>
    <style>
        body { font: .75rem Georgia, Times, "Times New Roman",
        serif;
            line-height: normal; }
        .rein { text-indent: 10%; }
        .raus { text-indent: -2em; }
    </style>
</head>
<body style="margin-left: 4em">
    <h1>Texteinzug (text-indent)</h1>
    <p style="text-indent: 15%;">Fortune plango vulnera
    stilantibus...</p>
    <p class="rein">Fortune plango vulnera stilantibus ocellis,
    quod sua...</p>
    <p class="raus">Fortune plango vulnera stilantibus ocellis,
    quod sua...</p>
</body>
</html>

```

- ① Der prozentuale Wert bewirkt eine Veränderung der Einrückung, sobald Sie die Breite des Browserfensters verändern. Der Einzug soll in diesem Fall immer 10 % der Fensterbreite betragen.
- ② Einen festen Wert geben Sie über eine entsprechende Maßeinheit an. Aufgrund des negativen Vorzeichens wird ein Einzug in die andere Richtung erzwungen.
- ③ Damit die negative Einrückung aus der Formatierung ② sichtbar wird, müssen Sie den Rand der Webseite vergrößern. Dies realisieren Sie für body über die CSS-Eigenschaft margin-left. Der linke Seitenrand beträgt somit 4em.
- ④ Die Angabe der Eigenschaft text-indent kann genauso direkt am Element erfolgen wie die Klassenzuweisung bei den beiden nachfolgenden Absätzen.

Texteinzug (text-indent)

Fortune plango vulnera stilantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata. Fortune solo sederam elatus, prosperitatis vario flore coconatus; quicquid enim florui felix et beatus, nunc a summo corruui gloria privatus.

Fortune plango vulnera stilantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata. In Fortune solo sederam elatus, prosperitatis vario flore coconatus; quicquid enim florui felix et beatus, nunc a summo corruui gloria privatus.

Eingerückter Text (kap04\text-indent.htm)

4.7 Bekannte Schriften neu entdecken

Obwohl die Schriftart Times New Roman als Standardschriftart eines jeden Browsers unter dem Betriebssystem Windows verfügbar ist, wird sie selten eingesetzt. Mithilfe der schon kennengelernten CSS-Eigenschaften können Sie die Wirkung der Schriftart stark verändern.

Beispiel: *kap04\time.htm*

Das Erscheinungsbild der Schrift verändern Sie, indem Sie den kursiven Schriftschnitt verwenden und einen erweiterten Zeichenabstand einsetzen.

```
<style>
①  h1.eins { font: italic 2rem 'Times New Roman', Times,serif;
      letter-spacing: 0.3333em; text-transform:
      lowercase; }
②  h1.zwei { font: italic 2rem 'Times New Roman', Times,serif;
      letter-spacing: 0.3333em; text-align: right; }
③  p      { line-height: 150%; text-indent: 2em; text-align:
      justify; }
</style>
```

- ① Die Schriftart wird auf Times New Roman oder eine ähnliche Serifenschrift eingestellt und in der Größe verdoppelt. Das Interessante an dieser Formatierung ist die kursive Darstellung der Schrift. Die einzelnen Zeichen werden mit einem Abstand von 0.3333em gesetzt, also einem Drittel der Breite des Buchstabens m. Dieser Text soll in Kleinbuchstaben erscheinen.
- ② Die zweite Definition unterscheidet sich darin, dass der Inhalt nicht kleingeschrieben, aber rechtsbündig ausgerichtet wird.
- ③ Die Absätze, die den Überschriften folgen sollen, werden im Blocksatz mit einer vergrößerten Zeilenhöhe und einem 2em großen Texteinzug dargestellt.

Die Formatierungen weisen Sie den beiden Überschriften über die Klassenangabe `class="eins"` bzw. `class="zwei"` zu. Die Absätze mit dem eigentlichen Inhalt werden automatisch durch den definierten Typ-Selektor `p` formatiert.

o m n i a s o l t e m p e r a t p u r u s

Et subtilis, novo mundo reserat faciem Aprilis; ad Amorem prosperat animus herilis et iocundis imperat deus puerilis. Rerum tanta novitas in solemni vere et veris auctoritas iubet nos gaudere, vias prebet solitas, et in tue vere fides est et probitas tuum retinere. Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentaliter absens in remota.

O m n i a s o l t e m p e r a t p u r u s

Et subtilis, novo mundo reserat faciem Aprilis; ad Amorem prosperat animus herilis et iocundis imperat deus puerilis. Rerum tanta novitas in solemni vere et veris auctoritas iubet nos gaudere, vias prebet solitas, et in tue vere fides est et probitas tuum retinere. Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentaliter absens in remota.

Kursive Überschriften in der Schriftart Times New Roman (kap04\time.htm)

4.8 Zeilenumbruch

Mit CSS können Sie Text vor einem Zeilenumbruch schützen. Hierfür gibt es die CSS-Eigenschaft `white-space` mit verschiedenen Werten.

```
white-space: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ <code>normal</code> = automatischer Zeilenumbruch ✓ <code>pre</code> = Umbrüche, wie im Quelltext ✓ <code>pre-wrap</code> = wie <code>pre</code>, außer dass bei Platzmangel ein Umbruch gesetzt wird ✓ <code>pre-line</code> = wie <code>pre</code>, aber mehrere Leerzeichen werden zusammengefasst ✓ <code>nowrap</code> = Zeilenumbruch unterbinden
Standardwert	Normal
Vererbbar	Ja
Anwendbar auf	Alle Blockelemente

Der automatische Zeilenumbruch erfolgt, wenn eine Textzeile länger ist, als das beinhaltende Element breit ist. Mit dem Wert `pre` werden die Zeilenumbrüche und Leerzeichen so angezeigt, wie sie auch im Quellcode stehen. Bei der Verwendung von `nowrap` wird zwar ein Zeilenumbruch unterbunden, mit dem Element `br` können Sie ihn jedoch jederzeit erzwingen. Mehrere Leerzeichen werden zu einem zusammengefasst.

Beispiel: *kap04\white-space.htm*

Die nachfolgenden Tabellen werden mit derselben Breite angegeben. Durch die verschiedenen Werte von `white-space` werden die Auswirkungen auf die Breite der Tabelle sichtbar. In der Tabelle wird nur der sichtbare Bereich der Webseite abgedruckt.

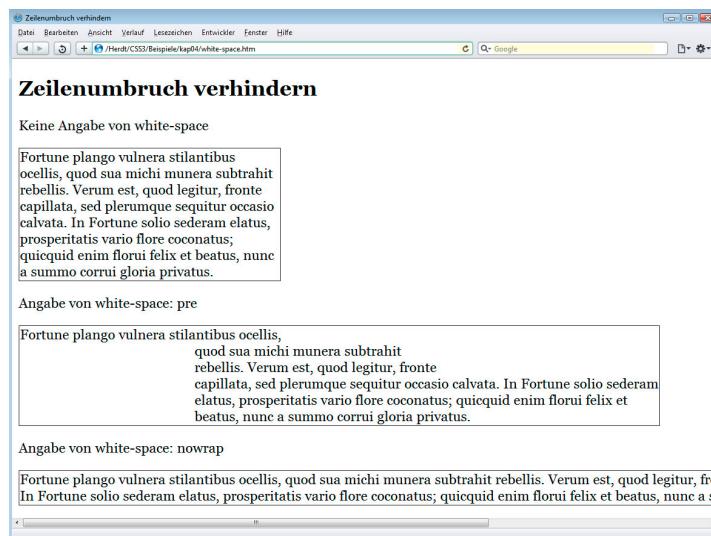
①	<pre> <style> table { width:300px; } [...] </style> [...] <h1>Zeilenumbruch verhindern</h1> <p>Keine Angabe von white-space</p> </pre>
---	--

```

② <table summary="Keine Angabe von white-space">
    <tr>
        <td>Fortune plango vulnera stilantibus ocellis, quod sua
            michi munera subtrahit rebellis. Verum est, quod
            legitur, fronte capillata...</td>
    </tr>
</table>
<p>Angabe von white-space: pre</p>
<table summary="Angabe von white-space: pre">
    <tr>
        <td style="white-space: pre;">Fortune plango vulnera
            stilantibus ocellis, quod sua michi munera subtrahit
            rebellis. Verum est, quod legitur, fronte...</td>
    </tr>
</table>
<p>Angabe von white-space: nowrap</p>
④ <table summary="Angabe von white-space: nowrap">
    <tr>
        <td style="white-space: nowrap">Fortune plango vulnera
            stilantibus ocellis, quod sua michi munera subtrahit
            rebellis<br>...</td>
    </tr>
</table>
...

```

- ① Die Breite der Tabellen wird auf 300 Pixel festgelegt.
- ② Der Inhalt der ersten Tabelle wird ohne die Angabe der Eigenschaft `white-space` angegeben. Der Browser nutzt den Standardwert `normal`. Daher wird der Text wie üblich bei Erreichen des rechten Zellrandes umgebrochen.
- ③ Der Inhalt der zweiten Tabelle wird mit `white-space: pre` so formatiert, dass die Zeilenumbrüche des Quellcodes angezeigt werden. Überschreitet der Text die Tabellenbreite, wird sie automatisch bis zum nächsten Zeilenumbruch vergrößert.
- ④ Bei Angabe von `white-space: nowrap` findet kein Zeilenumbruch statt, es sei denn, Sie erzwingen ihn per `
`. Die Zelle wird so breit, dass sie den gesamten Text umschließt – auch über die Fenstergröße des Browsers hinaus. Inhalte sind dann nur noch durch Scrollen erreichbar. Mehrere Leerzeichen im Quelltext werden zu einem zusammengefasst.



Automatische Zeilenumbrüche (kap04\white-space.htm)

Das HTML-Element `br` steht für ein line break und dient dazu, dem Browser mitzuteilen, dass eine Zeile zu Ende ist, und kann darum korrekt nur am Ende einer Zeile eingesetzt werden. Es sollte nicht dazu benutzt werden, Abstände einzufügen (z. B. nach Überschriften). Dies regeln Sie mittels CSS über entsprechende Abstände (siehe Kapitel 8).

Ein semantisch korrekter Einsatz wäre zum Beispiel am Ende eines jeden Verses in einem Lied oder Gedicht, in dem Sie für jede Strophe einen Absatz verwenden.

Silbentrennung

Auch die Trennung von Wörtern können Sie mit CSS regeln. Damit diese korrekt durchgeführt werden kann, muss der Browser wissen, welche Sprache mit welchen Trennungsregeln dazu benutzt werden soll.

```
① <html lang="de">
  <head>
    ...
  ② <style>
    article { hyphens: auto; }
  </style>
  </head>
  <body>
  ...
  ③ <article>
    <h1>Eine Überschrift</h1>
    <p>Sowohl für die Überschrift als auch für den hier
       geschriebenen Text wurde mittels hyphens die Silben-
       trennung aktiviert.</p>
  </article>
  ...
  </body>
</html>
```

- ① **Die Sprache (deutsch) wird angegeben.** Eine Liste aller Sprachen finden Sie unter https://www.w3schools.com/tags/ref_language_codes.asp
- ② Mit `hyphens: auto` aktivieren Sie die automatische Silbentrennung für alle Texte in Artikeln.
- ③ Die Worte in Artikeln werden den Regeln der angegebenen Sprache entsprechend getrennt.

Nicht immer sorgt der Automatismus für eine korrekte Trennung. Wenn Sie vermeiden wollen, dass sich dies negativ auf Ihre Reputation auswirkt (insbesondere, wenn sich Ihre Website mit Sprache oder verwandten Themen beschäftigt), sollten Sie sich nicht auf den Browser verlassen.

Besonders lange Wörter können das Layout negativ beeinflussen, wenn sie nicht getrennt werden. Sie haben die Möglichkeit, händisch ein HTML-Entity mitzugeben, das dem Browser mitteilt, wo ein Wort zu trennen ist. Dies ist ein benanntes Entity und wird wie folgt notiert: ­

Beispiel

```
Donau&shy;;dampf&shy;;schiff&shy;;fahrts&shy;;gesellschaft
```

Obwohl das mühsam ist, hat es doch einen Vorteil: Sie bestimmen exakt, welche Wörter wo getrennt werden dürfen. Da Sie das meist nur bei sehr langen Wörtern benötigen, ist dieses Vorgehen durchaus praktisch nutzbar. Bedenken Sie auch, dass im Quelltext das Wort „Donaudampfschiffahrtsgesellschaft“ nicht mehr vorkommt, wenn Sie die Entities eingefügt haben (z. B. wenn Sie im Editor danach suchen). **Google erkennt das Wort dennoch**, sodass durch die Entities die Auffindbarkeit Ihrer Site in der Suchmaschine nicht leidet.

4.9 Übungen

Übung 1: Theoriefragen zu Textabständen

Level		Zeit	ca. 3 min
Ergebnisdatei	<i>kap04\uebung1-2.doc</i>		

1. Erläutern Sie den Unterschied zwischen einer Zeilenhöhe, einem Zeichenabstand und einem Wortzwischenraum.
2. Wie groß sollte eine Zeilenhöhe in einem Fließtext mindestens sein, damit das Lesen am Bildschirm vereinfacht wird? Wovon ist die Zeilenhöhe abhängig?

Übung 2: Lesbarkeit verbessern

Level		Zeit	ca. 15 min
Übungsdatei	<i>kap04\chalkidiki.htm</i>		
Ergebnisdateien	<i>kap04\uebung3.htm, kap04\uebung4.htm, kap04\uebung5.htm, kap04\uebung6.htm</i>		

1. Bearbeiten Sie die Webseite zur Geschichte Chalkidikis aus dem vorherigen Kapitel weiter. Verdoppeln Sie die Zeilenhöhe der Überschrift.
2. Zentrieren Sie die Überschrift.
3. Die Abstände der einzelnen Absätze setzen Sie auf 150 %, die Zeichenabstände erweitern Sie um jeweils einen Pixel.
4. Rücken Sie die einzelnen Absätze um 50 Pixel ein.

The screenshot shows a Firefox browser window with the title bar "Firefox". The main content area displays a page titled "Die Geschichte von Chalkidiki". The page text discusses the history of Chalkidiki, mentioning its colonization in the 7th and 8th centuries BC by Greeks from Chalkis, its later unification under Macedonia, and its subsequent history through Roman, Byzantine, and Ottoman rule, ending with its independence in 1832.

Die Geschichte von Chalkidiki

Die **Chalkidiki** wurde im 7. und 8. Jh. v. Chr. kolonisiert. Die ersten Siedler, von denen die Chalkidiki ihren Namen erhielt, stammten aus der Stadt **Chalkis** auf der **Insel Eboa**. Zu jener Zeit entstehen überall in Griechenland voneinander unabhängige und oft untereinander verfeindete Stadtstaaten.

Nach zahlreichen leidvollen Kriegen wird im 5. Jh. v. Chr. der "Chalkidische Bund" geschlossen, um militärisch, politisch und wirtschaftlich eng zusammenzuarbeiten. Mitte des 4. Jh. v. Chr. wird die Chalkidiki dennoch von seinen starken Nachbarn, den staatlich geeinten und straff organisierten **Makedonen**, unterworfen. Unter König Philipp II. und später seinem Sohn Alexander dem Großen entsteht erstmals ein geeintes Griechenland. Allerdings ist Griechenland nur noch ein kleiner, relativ unbedeutender Teil im mächtigen Reich von Alexander dem Großen.

In der römischen Zeit wird Makedonien erobert und Chalkidiki wird durch Überfälle der **Goten** im Jahr 269 weitgehend zerstört und entvölkert. Ende des 4. Jh. fällt ganz Griechenland an **Ostrom**. In den folgenden Jahrhunderten der oströmisch-byzantinischen Zeit werden die Städte der Chalkidiki immer wieder Opfer von Plünderei und Zerstörung.

Mitte des 15. Jh. wird Griechenland, und somit auch die Chalkidiki, von den **Türken** erobert. Das Byzantinische Reich ist 1453 endgültig zerstört. Im 16. Jh. führen die Türken auf der Chalkidiki die Seidenraupenzucht und den Tabakanbau ein. Die Region erlebt einen gewaltigen wirtschaftlichen Aufschwung.

Nach Aufständen der Griechen gegen die türkische Fremdherrschaft wird Griechenland 1830 ein **unabhängiger Staat**. Aufstände auf der Chalkidiki werden aber niedergeschlagen. 1832 wird ein **Bayer** zum ersten König Griechenlands ernannt. Die Chalkidiki, viele Inseln und Nordgriechenland bleiben unter türkischer Herrschaft.

Im ersten Balkankrieg 1912-1913 verbünden sich Griechenland, Bulgarien, Serbien und

Das Endergebnis der Übung

5

Farben

5.1 Farbwerte

Die Buchstaben **R**, **G** und **B** stehen für die Grundfarben Rot, Grün und Blau (engl.: red, green, blue). Aus diesen drei **Grundfarben** können Sie durch Mischen über 16 Millionen verschiedene Farbvarianten für die Darstellung am Bildschirm erzeugen.

Jeder Bildpunkt wird in der Farbauflösung von 24 Bit durch drei Bytes (also 3 mal 8 Bit) beschrieben, den Farbwerten der Grundfarben. Der Inhalt dieser Bytes kann Werte von 0 bis 255 annehmen. Dabei entsprechen diese Werte dem Grad der Farbintensität (je höher, desto heller – additive Farbmischung). Die Farbkombination 255, 255, 255 entspricht der Farbe Weiß, die Kombination 0, 0, 0 entspricht der Farbe Schwarz.

Mithilfe der drei Grundfarben sind demzufolge $256 \times 256 \times 256 = 16.777.216$ Farbkombinationen möglich. Es wird von einer **24-Bit-Farbtiefe** gesprochen.

Die drei Grundfarben:

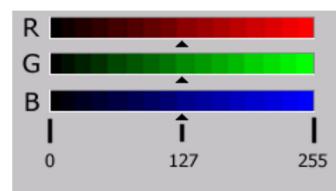
R: 255, G: 0, B: 0 = Rot
 R: 0, G: 255, B: 0 = Grün
 R: 0, G: 0, B: 255 = Blau

Einige Mischfarben:

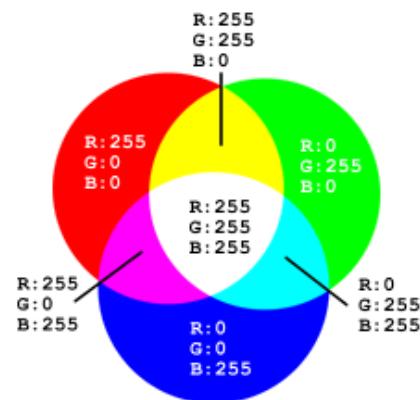
R: 0, G: 255, B: 255 = Aqua
 R: 255, G: 255, B: 0 = Gelb
 R: 255, G: 0, B: 255 = Magenta
 R: 255, G: 255, B: 255 = Weiß

Die einzige Farbe, die ohne Mischung entsteht:

R: 0, G: 0, B: 0 = Schwarz



Die drei Grundfarben in den verschiedenen Abstufungen
 (kap05\rgb-modell.htm)



Mischen der drei Grundfarben
 (kap05\rgb.htm)

Hexadezimal

Unser dezimales Zahlensystem kennt zehn Ziffern (0–9). Das hexadezimale Zahlensystem kennt 16 Ziffern, für die wir keine Zeichen in unserem System haben, werden als Buchstaben notiert: 10 = A, 11 = B, 12 = C, 13 = D, 14 = E, 15 = F.

Das Hexadezimalsystem enthält die folgenden Ziffern:

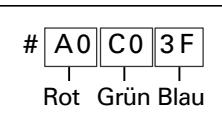
0 1 2 3 4 5 6 7 8 9 A B C D E F

Dezimale und hexadezimale Zahlen im Vergleich:

Hexadezimalzahl	Dezimalzahl
0	0
10	A
15	F
16	10
255	FF

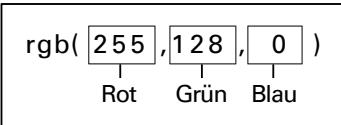
Eine hexadezimale Zahl kann 16 verschiedene Zustände besitzen (0–15). Für die Angabe eines bestimmten Farbwertes stehen jeweils zwei Stellen zur Verfügung, dies ergibt $16 \times 16 = 256$ Möglichkeiten.

Die Farbanteile geben Sie jeweils als zweistellige Werte zwischen 0 und F an. Der erste Wert R steht für den Rot-Anteil, G für den Grün-Anteil und B für den Blau-Anteil. Sind bei der Farbangabe in jedem Anteil beide Stellen gleich, brauchen Sie nur jeweils eine Stelle anzugeben. Für #FFAA88 genügt dann #FA8, für #000000 reicht die Angabe von #000. Angaben, die nicht verkürzt werden können, sind z. B. #FF36FF oder #EAEAEA.



Dezimalwerte

Bei der dezimalen Schreibweise werden die hexadezimalen Werte als Werte zwischen 0 und 255 angegeben. Beispielsweise steht der Hex-Wert 00 für die dezimale 0, der Hex-Wert FF für die Dezimalzahl 255 und der Hex-Wert 80 für die Dezimalzahl 128. Die drei Farbanteile werden durch Kommas oder Leerzeichen voneinander getrennt und in Klammern gesetzt. Davor geben Sie das Schlüsselwort `rgb` an.



`color: rgb(255, 128, 0);`

oder

`color: rgb(255 128 0);`

Dezimalwerte mit Alpha-Kanal

Neben der Farbe können Sie die **Deckkraft** der Farbe mit einem Wert zwischen 0 und 1 angeben, wobei 1 für komplett deckend und 0 für vollkommen transparent steht.

Notieren Sie die folgende Angabe, wenn Sie möchten, dass der Hintergrund durch die angegebene Farbe durchscheinen soll. Der letzte Wert steht für den Grad der Transparenz. Bei einer Angabe von 0.5 ist die Farbe zu 50 % „durchlässig“.

```
color: rgba(255, 128, 0, 0.5);
```

oder

```
color: rgb(255 128 0 / 0.5);
```

oder

```
color: rgb(255 128 0 / 50%);
```

Hexadezimale Farbangaben lassen sich als ein Wert aus einer Vorlage (z. B. Grafikprogramm) in den eigenen Quelltext kopieren. Bei Dezimal-Angaben sind dagegen drei bzw. vier Werte inklusive Alpha-Kanal zu übertragen, was die Übertragung umständlich und damit fehleranfällig macht. Deswegen werden Sie in der Regel hexadezimalen Angaben in fremden Quelltexten begegnen.

Prozentwerte

Die Angabe der Farbanteile erfolgt, wie bei den Dezimalwerten, mit dem vorangestellten Schlüsselwort `rgb`. Beispielsweise sind Angaben von `rgb(100%, 50%, 0%)` oder `rgb(66.66%, 0.5%, 23.12%)` möglich.

Farbnamen

Sie haben die Möglichkeit, den englischen Namen der jeweiligen Farbe anzugeben. Die Farbnamen sind dabei nicht case-sensitive, d. h., es wird keine Groß- und Kleinschreibung berücksichtigt.

Die 16 standardisierten Farbnamen lauten (vgl. kap05\farbname.htm):			
black = schwarz	maroon = dunkelrot	green = dunkelgrün	olive = olivgrün
navy = dunkelblau	purple = violett	teal = blaugrün	gray = dunkelgrau
silver = hellgrau	red = rot	lime = hellgrün	yellow = gelb
blue = blau	fuchsia = magenta	aqua = hellblau	white = weiß

black = schwarz	maroon = dunkelrot	green = dunkelgrün	olive = olivgrün
navy = dunkelblau	purple = violett	teal = blaugrün	gray = dunkelgrau
silver = hellgrau	red = rot	lime = hellgrün	yellow = gelb
blue = blau	fuchsia = magenta	aqua = hellblau	white = weiß

Die Darstellung von anderen Farbnamen als den offiziell standardisierten ist zwar vom jeweiligen Browser abhängig, jedoch interpretieren die aktuellen Browser weitere Farbnamen vollkommen identisch.

216 dieser Farben werden auch als websichere Farben bezeichnet. Dies bedeutet, dass diese Farben in der Regel von jedem Browser im selben Farbton dargestellt werden. Die Farben werden dabei aus allen Kombinationen der Werte 00, 33, 66, 99, CC und FF gebildet.

Eine Übersicht der erweiterten Farbnamen finden Sie in der HTML-Datei *kap05\farbname.htm*. Darin sind die Farbnamen mit den entsprechenden hexadezimalen und RGB-Werten aufgelistet.

Indem Sie die Farbwerte hexadezimal oder dezimal angeben, arbeiten Sie browserunabhängig und haben die freie Auswahl aus 16,7 Millionen Farben.

5.2 Textfarbe ändern

Nachdem Sie in den vorigen Kapiteln die Schrift und die Ausrichtung eines Textes geändert haben, können Sie nun Farbe in Ihre Webseite bringen. In einigen der bisherigen Beispiele haben Sie bereits die entsprechende CSS-Eigenschaft angewendet.

```
color: Farbwert;
```

Werte	<ul style="list-style-type: none">✓ Schlüsselwort als Farbname✓ Hexadezimal: #RRGGBB oder #RGB✓ Dezimalwert: rgb(rrr, ggg, bbb), rgb(rrr ggg bbb)✓ Dezimalwert plus Alphakanal: rgba(rrr, ggg, bbb, a.a), rgb(rrr, ggg, bbb / a.a)✓ Prozentwert: rgb(rrr.rr%, ggg.gg%, bbb.bb%)
Standardwert	Abhängig vom Element (normaler Text, (besuchter) Link) und von den Browbereinstellungen (meist schwarz)
Vererbbar	Ja, auf die Unterelemente
Anwendbar auf	Alle Elemente

Sie haben die Wahl, welchen Wert Sie als Farbwert angeben. Alle Angaben werden von den aktuellen Browsern gleichermaßen richtig umgesetzt, allerdings hat sich die Angabe von Hexadezimalwerten als Quasi-Standard etabliert.

Beispiel: *kap05\textfarbe.htm*

Sie erstellen ein HTML-Dokument, in dem jeder Textabsatz über eine andere Angabe des Farbwertes formatiert wird. Durch die gleichzeitige Verwendung verschiedener Farbwerte erhalten Sie einen Eindruck, wie sich die Farben aus den RGB-Farbanteilen zusammensetzen. Die KlassenSelektoren werden anhand des Farbnamens und der möglichen Werte bezeichnet.

```
<html lang="de">
<head>
    <title>Farbangaben: Schreibweisen</title>
    <style>
        p                               { color: #0000FF; } /* blau */
        .rot-hexlang                   { color: #FF0000; }
        .gruen-hexkurz                 { color: #0F0; }
```

```

④ .blau-dezimal           { color: rgb(0, 0, 127); }
⑤ .blau-dezimal-transparent { color: rgba(0, 0, 127, 0.3); }
⑥ .lila-prozent          { color: rgb(50%, 0%, 50%); }
⑦ .ocker-name             { color: olive; }

</style>
</head>
<body>
    <h1>Eigenschaft "Textfarbe" ändern</h1>
    <p class="rot-hexlang">Roter Text durch Angabe von:  
#FF0000;</p>
    <p class="gruen-hexkurz">Hellgrüner Text durch Angabe von:  
#0F0;</p>
    <p class="blau-dezimal">Dunkelblauer Text durch Angabe von:  
rgb(0,0,127);</p>
    <p class="blau-dezimal-transparent">Dunkelblauer,  
teiltransparenter Text durch Angabe von:  
rgba(0,0,127,0.3);</p>
    <p class="lila-prozent">Lila Text durch Angabe von: rgb(50%,  
0%, 50%);</p>
    <p class="ocker-name">Olivfarbener Text durch Angabe von:  
olive;</p>
</body>
</html>

```

- ① Für alle Absätze geben Sie eine blaue Textfarbe #0000FF vor.
- ② Ein Element, das die Klasse rot-hexlang erhält, bekommt roten Text (#FF0000).
- ③ Die Kurzform der Farbangabe von #00FF00 ist in der Deklaration von grün-hexkurz hinterlegt. Elemente mit dieser Klasse werden grün eingefärbt (#0F0).
- ④ Eine dunkelblaue Schrift wird über die Dezimalangabe definiert. Mit dem Wert 127 des Farbanteils Blau, der die Hälfte des maximal möglichen Wertes 255 darstellt, legen Sie eine dunkelblaue Farbe fest. Ein höherer Wert würde das Blau aufhellen, ein niedrigerer Wert würde die Farbe weiter abdunkeln.
- ⑤ Der Schrift wird derselbe Farbwert zugewiesen wie der dunkelblauen Schrift. Da die Farbe aber nur zu 30 % deckt, scheint der helle Hintergrund durch. Vor einem dunklen Hintergrund erscheint diese Schrift dunkler. In beiden Fällen ist die Schrift schlecht zu lesen. Daher sollten Sie mit einem Wert näher an 1 für eine stärkere Deckung von Schriften sorgen.
- ⑥ Die Farbe Lila erhalten Sie, wenn Sie die beiden Farbanteile Rot und Blau zu je 50 % mischen.
- ⑦ Mit dem englischen Farbnamen olive legen Sie die Schriftfarbe Ocker fest.

Eigenschaft "Textfarbe" ändern

Roter Text durch Angabe von: #FF0000;

Hellgrüner Text durch Angabe von: #0F0;

Dunkelblauer Text durch Angabe von: rgb(0,0,127);

Dunkelblauer, teiltransparenter Text durch Angabe von: rgba(0,0,127,0.3);

Lila Text durch Angabe von: rgb(50%,0%,50%);

Olivfarbener Text durch Angabe von: olive;

Texte in verschiedenen Farben (kap05\textfarbe.htm)

Mit der Eigenschaft `color` beeinflussen Sie nicht allein die Farbe von Buchstaben, sondern auch Unterstreichungen (`text-decoration`) und Aufzählungszeichen von sortierten und unsortierten Listen (`ul` und `ol`).

5.3 Hintergrundfarben

background-color: Farbwert;

Werte	<ul style="list-style-type: none"> ✓ Schlüsselwort als Farbname ✓ Hexadezimal: #RRGGBB oder #RGB ✓ Dezimalwert: rgb(rrr, ggg, bbb), rgb(rrr ggg bbb) ✓ Dezimalwert plus Alphakanal: rgba(rrr, ggg, bbb, a.a), rgb(rrr, ggg, bbb / a.a) ✓ Prozentwert: rgb(rrr.rr%, ggg.gg%, bbb.bb%)
Standardwert	Abhängig von den Browsereinstellungen, meist weiß
Vererbbar	Ja
Anwendbar auf	Alle Elemente

Beispiel: *kap05\hintergrundfarbe.htm*

Dieselben Farbangaben werden diesmal als Wert für das Attribut `background-color` angegeben. Daher nehmen Sie für das Beispiel nur die Stylesheet-Definition vor.

```
<style>
p { font: 14px Georgia, Times, "Times New Roman", serif;
    color: #FFFFFF; }
.rot-hexlang          { background-color: #FF0000; }
.gruen-hexkurz        { background-color: #0F0; }
.blau-dezimal         { background-color: rgb(0, 0, 127); }
.blau-dezimal-transparent { background-color: rgb(0, 0, 127,
0.3); }
.lila-prozent         { background-color: rgb(50%, 0%, 50%); }
.ocker-name           { background-color: olive; }
</style>
```

Hintergrundfarben verwenden

Damit sich die Schrift von dem Hintergrund abhebt, wird sie auf Weiß gesetzt.

Hintergrundfarbe ändern

Roter Hintergrund durch Angabe von: #FF0000;
Hellgrüner Hintergrund durch Angabe von: #0F0;
Dunkelblauer Hintergrund durch Angabe von: rgb(0, 0, 127);
Dunkelblauer, teiltransparenter Hintergrund durch Angabe von: rgba(0, 0, 127, 0.3);
Lila Hintergrund durch Angabe von: rgb(50%, 0%, 50%);
Hintergrund in ocker durch Angabe von: olive;

Texte mit verschiedenen Hintergrundfarben
(*kap05\hintergrundfarbe.htm*)



Haben Sie bemerkt, wie schlecht manche Texte lesbar sind? Achten Sie deshalb immer auf ausreichende Hell-Dunkel-Kontraste, um das Lesen allen zu ermöglichen – auch unter widrigen Umständen wie bei hellem Sonnenschein oder schlechten Displays.

Andere Elemente einfärben mit `background-color`

Im Gegensatz zur Eigenschaft `color`, die grundlegend auf textliche Inhalte beschränkt bleibt, können Sie mit `background-color` nahezu jedem Element einen farbigen Hintergrund zuweisen. Neben dem eigentlichen Seitenhintergrund sind dies z. B. Tabellenzellen, Listen-einträge oder Links.

Beispiel: `kap05\hintergrundfarbe-andere.htm`

Dem Seitenhintergrund, einer Überschrift, einer Trennlinie, einem Bereich, einer Tabelle und ihren Zellen weisen Sie unterschiedliche Hintergründe zu.

```
<html lang="de">
<head>
    <title>Hintergrundfarben verschiedener Elemente</title>
    <style>
        ① body      { background-color: lightyellow; }
        ② h1       { background-color: #DC143C; color:
                      rgb(255,250,240); }
        hr, table { background-color: deepskyblue; }
        div      { background-color: rgb(224,255,255); }
        td       { background-color: rgb(94.12%, 90.2%, 55.9%); }
    </style>
</head>
<body>
    ④ <h1>Farbige Elemente</h1>
    <div>
        ⑤ <p>Dies ist der erste Absatz innerhalb des Elementes
            "div".</p>
        <table>
            ⑥ <tr><td>Eine blau umrandete Tabelle</td>
                <td>mit Zellen in der Farbe Khaki.</td></tr>
            <tr><td>&nbsp;;</td>
                <td>&nbsp;;</td></tr>
        </table>
        <p>Dies ist der letzte Absatz innerhalb des Elementes
            "div".</p>
    </div>
</body>
</html>
```

- ① Den Seitenhintergrund legen Sie über den Typ-Selektor `body` für das gesamte Dokument fest.
- ② Den anderen Typ-Selektoren weisen Sie ebenfalls die CSS-Eigenschaft `background-color` mit unterschiedlichen Werten zu.

- ③ <body> löst die Verwendung der Hintergrundfarbe Hellgelb (lightyellow) aus.
- ④ Der Überschrift h2 wird neben dem rötlichen Hintergrund die Schriftfarbe Blumenweiß zugewiesen, um den Kontrast zwischen Vorder- und Hintergrund aufrechtzuerhalten.
- ⑤ Das Element div kann weitere Inhalte enthalten, wie die nachfolgenden Absätze und die Tabelle. Die freien Flächen sollen mit einem Hintergrund in der Farbe Cyan sichtbar gemacht werden.
- ⑥ Die Tabelle erhält einen himmelblauen Hintergrund. Durch die Zellen, die einen anderen Hintergrund haben und automatisch die Fläche der Tabelle füllen, sind nur die Zellzwischenräume als Tabellenhintergrund sichtbar.

Farbige Elemente

Dies ist der erste Absatz innerhalb des Elementes "div".

Eine blau umrandete Tabelle mit Zellen in der Farbe Khaki.

Dies ist der letzte Absatz innerhalb des Elementes "div".

Elemente mit verschiedenen Hintergrundfarben
(kap05\hintergrundfarbe-andere.htm)

5.4 Übung

Theoriefragen zu Farbangaben

Level		Zeit	ca. 15 min
Ergebnisdateien	kap05\uebung1.htm , kap05\uebung2.htm , kap05\uebung3.htm		

1. Welche Farbe erhalten Sie, wenn Sie Grün und Rot mischen?
2. Erstellen Sie mehrere Absätze mit blauer Textfarbe. Stellen Sie die fünf möglichen CSS-Farbangaben dar.
3. Welche Farbangaben können als dreistellige Hexadezimalangabe zusammengefasst werden?

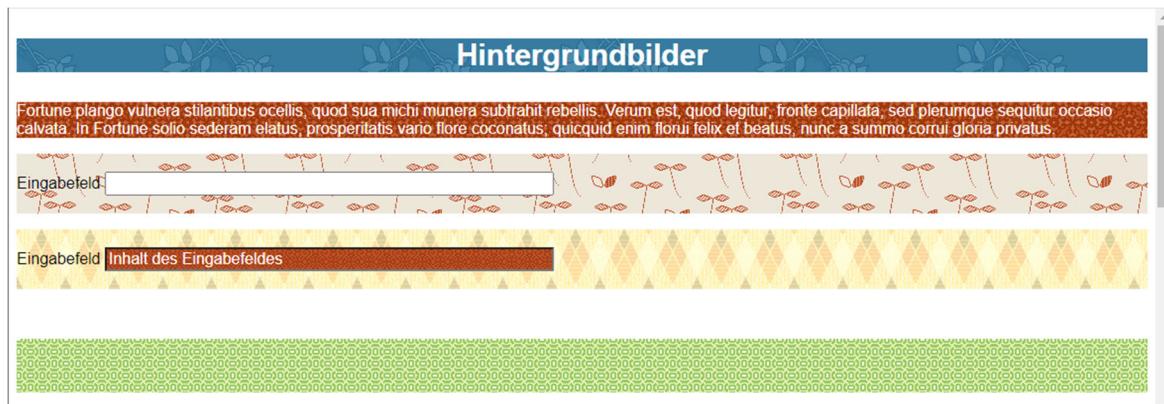
#FCFCFC, #AAAAAA, #355335, #BB7733, #F000FF

6

Hintergrundbilder und Filter nutzen

6.1 Einleitung zu Hintergründen

Hintergründe eröffnen zahllose Design-Möglichkeiten – Sie können einem Element auch mehrere Hintergründe zuweisen. Zudem lassen sich Hintergründe (genau wie andere Elemente) mit der CSS-filter-Eigenschaft weichzeichnen, umfärben und/oder mit zahlreichen weiteren grafischen Effekten versehen.



Elemente mit verschiedenen Hintergründen ([kap06\background-images.htm](#))

Übersicht

In der folgenden Übersicht finden Sie die aktuell im Standard festgelegten CSS-Eigenschaften.

Mit diesen Eigenschaften wird auch das Verhalten von mehreren Hintergründen beschrieben, sodass dafür keine weitere Eigenschaft nötig ist.

- ✓ background (Kurzform/Kurzschriftweise)
- ✓ background-attachment
- ✓ background-image
- ✓ background-position
- ✓ background-size
- ✓ background-color
- ✓ background-origin
- ✓ background-repeat

6.2 Hintergrundgrafiken verwenden

Neben der im letzten Kapitel beschriebenen Möglichkeit, HTML-Elemente mit einer Farbe zu unterlegen, können Sie auch Grafiken für Hintergründe verwenden. Hierfür setzen Sie die Eigenschaft `background-image` ein.

```
background-image:url( Wert );
```

Werte	✓ none ✓ URL
Standardwert	none
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Mit CSS können Sie einzelnen Elementen eine oder mehrere Hintergrundgrafiken zuweisen. Hierfür nutzen Sie Grafiken in den Formaten GIF, JPEG oder PNG. Den Pfad zur Grafik geben Sie innerhalb der runden Klammern an, wobei die Anführungszeichen nur bei Leerzeichen im Dateinamen angegeben werden müssen. Es sind relative und absolute Pfadangaben erlaubt, z. B. `.. /images/bild.png` oder `https://www.example.com/images/bild.jpg`.

Beispiel: `kap06\pattern.css`

In einer externen Stylesheet-Datei bestimmen Sie die möglichen Hintergrundgrafiken und legen entsprechend den Dateinamen die Selektoren fest.

```
① .back01 { background-image:url(patterns/img_01.gif) }
.back02 { background-image:url(patterns/img_02.gif) }
.back03 { background-image:url(patterns/img_03.gif) }
...
② .back01, .back02, .back04, .back05, .back07, .back08, .back11
{ color: black; }
.back03, .back06, .back09, .back10, .back12 { color: white; }
```

- ① Den Klassen `back01` bis `back12` werden die entsprechenden Hintergrundgrafiken zugewiesen. Die Grafiken befinden sich jeweils im Unterordner `patterns` des aktuellen Dokumentenordners.
- ② Oft leidet die Lesbarkeit erheblich, wenn Texte mit Grafiken hinterlegt werden. Wenn Sie Hintergrundfarben oder -bilder verwenden, geben Sie unbedingt immer eine passende Schriftfarbe an. Wie in der Abbildung am Anfang dieses Kapitels ersichtlich, lassen sich Texte nicht lesen, wenn der Hintergrund sehr helle und dunkle Bereiche enthält. Das betrifft vor allem die Absätze 5 und 10. Weder schwarze noch weiße Schrift können die Lesbarkeit wiederherstellen.

Beispiel: *kap06\background-images-kurz.htm*

In diesem Beispiel wird das HTML-Dokument erstellt, das als Abbildung auf der ersten Seite dieses Kapitels dargestellt ist.

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>Hintergrundmotive</title>
    ①      <link rel="stylesheet" href="pattern.css">
  </head>
  <body>
    ②      <p class="back01">01. Lorem ipsum dolor sit...</p>
    <p class="back02">02. Repudiandae magnam...</p>
    <p class="back03">03. Voluptate iure in ...</p>
    ③      ...
  </body>
</html>
```

- ① Mit dem link-Element verknüpfen Sie die externen Formatdefinitionen der Datei *pattern.css* mit dem HTML-Dokument.
- ② Dem ersten Absatz weisen Sie über die Klasse back01 die Hintergrundgrafik *patterns/img_01.gif* zu, dem zweiten Absatz mit der Klasse back02 die Hintergrundgrafik *patterns/img_02.gif*.
- ③ Mit den folgenden 11 Absätzen verfahren Sie analog.

Außergewöhnliche Hintergründe (auch kostenlose), die Sie für Ihre HTML-Dokumente nutzen können, finden Sie im Internet mit der Suchmaschine Ihrer Wahl.

Besonders hochwertige Grafiken erhalten Sie auf den kostenpflichtigen Angeboten von Adobe Stock, Shutterstock, iStock Photo und ähnlichen Anbietern.

Oft gilt, „weniger ist mehr“: Subtile Hintergründe wirken edel und unaufdringlich. Beispiele zum Download finden Sie etwa unter <https://www.toptal.com/designers/subtlepatterns/>.

6.3 Hintergrund wiederholen

Standardmäßig wird eine Grafik „gekachelt“, wenn sie als Hintergrundbild eingebunden ist. Das heißt, sie wird vertikal und horizontal wiederholt, um die gesamte Fläche des zugewiesenen Elements zu füllen (so als ob Kacheln auf einer Wand verlegt werden). Über die CSS-Eigenschaft **background-repeat** können Sie festlegen, in welche Richtung eine Hintergrundgrafik wiederholt werden soll.

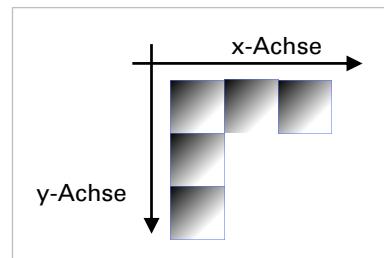
```
background-repeat: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ repeat = vertikale und horizontale Wiederholung ✓ repeat-x = horizontale Wiederholung (x-Achse) ✓ repeat-y = vertikale Wiederholung (y-Achse) ✓ no-repeat = keine Wiederholung; Bild wird als einzelne Grafik dargestellt
Standardwert	Repeat
Vererbbar	Nein
Anwendbar auf	Alle Elemente



Diese Eigenschaft können Sie nur zusammen mit einer Hintergrundgrafik verwenden.

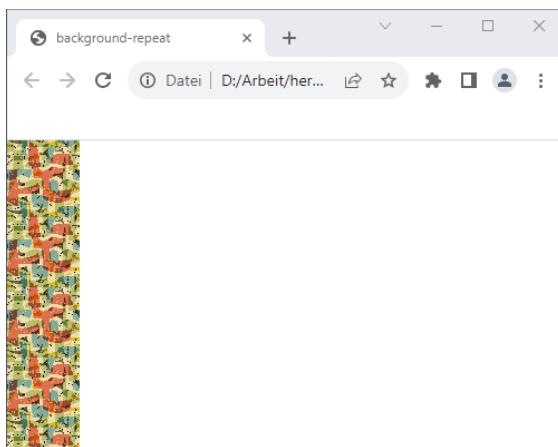
Dank background-repeat können Sie Einfluss darauf nehmen, ob eine Hintergrundgrafik entlang der x-Achse, der y-Achse oder in beide Richtungen wiederholt werden soll.



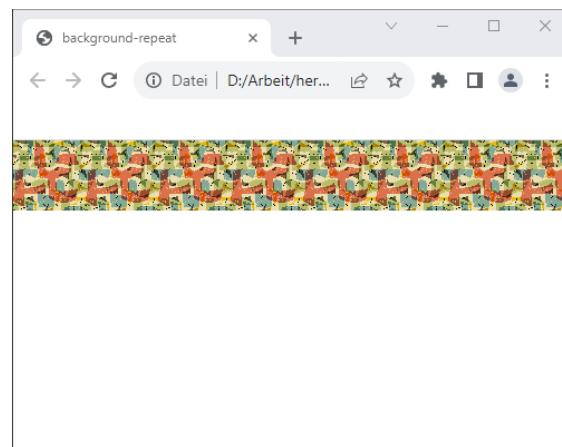
Beispiel: *kap06\background-repeat-y.htm*

Das Element body erhält über die Angabe der CSS-Eigenschaften einen Hintergrund, der von oben nach unten verläuft. Seitlich wird die Grafik nicht vervielfältigt.

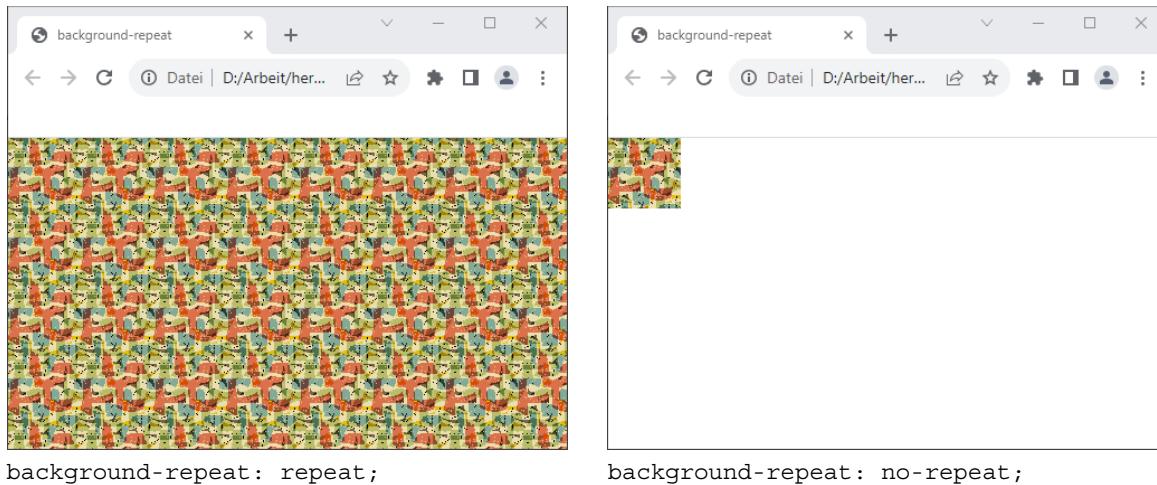
```
<body style="background-image:url(patterns/img_12.gif);
            background-repeat: repeat-y;">
</body>
```



background-repeat: repeat-y;



background-repeat: repeat-x;



Die Darstellung von Elementen erfolgt in mehreren Ebenen. Die unterste wird mit der Hintergrundfarbe gefüllt, darüber liegen die Hintergrundbilder. Im Vordergrund befinden sich die im HTML eingesetzten Bilder und die Schrift. Sofern Sie die Eigenschaft `background-repeat` mit den Werten `repeat-x`, `repeat-y` und `no-repeat` verwenden, können Sie zusätzlich eine Hintergrundfarbe angeben, die ebenfalls angezeigt wird. Sie sehen dann einen eingefärbten Hintergrund mit einer zusätzlichen Grafik.

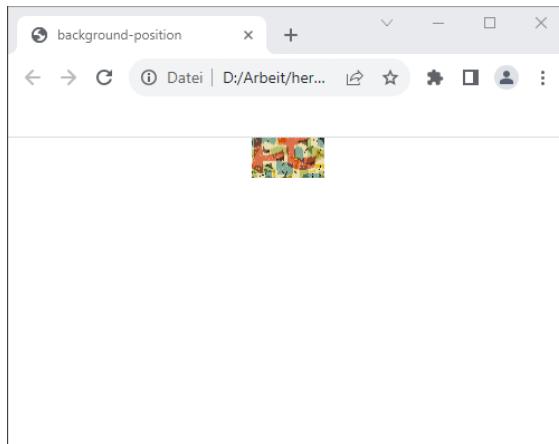
6.4 Hintergrund positionieren

Ein Hintergrundbild können Sie an einer bestimmten Stelle positionieren. Dann beginnt die Wiederholung der Grafik ab dieser Position. Die CSS-Eigenschaft dazu lautet `background-position` und kann nur in Zusammenhang mit `background-image` eingesetzt werden.

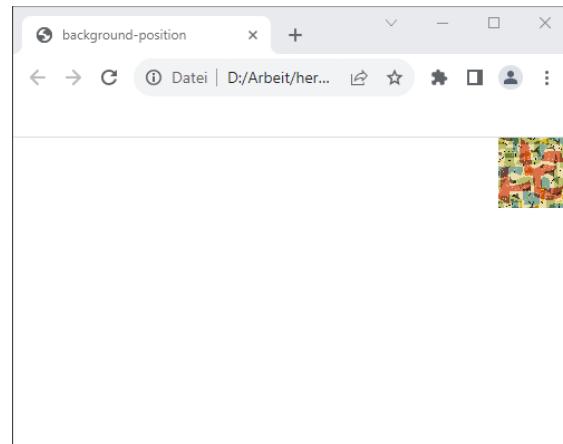
background-position: Wert;

Werte	<ul style="list-style-type: none"> ✓ top = horizontale Ausrichtung oben ✓ center = horizontale Ausrichtung mittig ✓ bottom = horizontale Ausrichtung unten ✓ left = vertikale Ausrichtung links ✓ center = vertikale Ausrichtung zentriert ✓ right = vertikale Ausrichtung rechts ✓ Längenangabe = Maßeinheiten für x- und y-Achse, z. B. 100 px, 10 mm ✓ Prozentangabe = Angaben für x- und y-Achse, z. B. 20 %
Standardwert	top left bzw. 0 0
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Die Angabe der Positionierungen erfolgt in der Schreibweise `background-position: horizontal vertikal`. Das heißt, zuerst werden die horizontalen Werte und danach die vertikalen Werte angegeben. Damit sind nur Kombinationen möglich wie `top left`, `center right`, `bottom center` usw.



`background-position: center center;`¹



`background-position: top right;`
(kap06/background-position2.html)

In der ersten Abbildung wird der Hintergrund vertikal und horizontal zentriert. Dennoch wird die Grafik oben bündig angezeigt, da der body einer Webseite nur so hoch ist wie sein Inhalt.

Bei Positionsangaben sind die beiden Werte als Abstand von der linken oberen Ecke des Elements anzugeben. Die Position können Sie dabei in verschiedenen Einheiten angeben, wie z. B.
`background-position: 10px 10%;`

¹ Die Hintergrundgrafik wirkt sich nicht auf die Höhe des Elternelementes aus. Da das Eltern-element hier leer ist, hat es keine Höhe. Es ist lediglich eine horizontale Linie ohne vertikale Ausdehnung. Wird darin die Hintergrundgrafik zentriert, wird die Mitte der Grafik an der Mitte des Elementes ausgerichtet. Das führt hier im leeren body dazu, dass nur die untere Hälfte der Hintergrundgrafik zu sehen ist.

Bei der prozentualen Angabe wird der Hintergrund relativ im Element positioniert. Dies bedeutet, wenn sich durch eine Größenänderung des Browserfensters auch die Größe des Elements ändert, passt sich die Position des Hintergrundbildes automatisch an. Die Angabe "0 0" ist mit der linken oberen Ecke (top left) des Elements und "100% 100%" mit der rechten unteren Ecke (bottom right) gleichzusetzen.

Beachten Sie das Verhalten von Positionierung und Wiederholung, wenn Sie beide Eigenschaften verwenden (siehe nebenstehende Abbildungen).

Die Positionierung der Hintergrundgrafik wird nur dann komplett durchgeführt, wenn die Eigenschaft auf `background-repeat: no-repeat` gesetzt wurde.

Geben Sie als Wiederholung `repeat-y` an, wird bei `background-position` nur der x-Wert beachtet, bei `repeat-x` der y-Wert.

Wenn keine Wiederholung angegeben ist und die Grafik den Hintergrund komplett bedeckt, wird kein Positions倅t beachtet.

Positionierung des Hintergrunds

`background-repeat:repeat-x; background-position:40px 50%;`
Berücksichtigt wird nur die vertikale Verschiebung von 50%.



`background-repeat:repeat-y; background-position:40px 50%;`
Berücksichtigt wird nur die horizontale Verschiebung von 50 Pixel.



`background-repeat:no-repeat; background-position:40px 50%;`
Hier werden beide Angaben zur Darstellung herangezogen.



`background-repeat:repeat; background-position:40px 50%;`
Keine der Angaben findet Berücksichtigung.



Verschiedene Positionierungen des Hintergrunds
(kap06\background-position.htm)

Beispiel: kap06\background-position.htm

Die in der Abbildung sichtbaren Positionierungen wurden mit dem nachfolgenden Quellcode erzeugt. Die Stylesheet-Definitionen wurden den Absäten über die Klassen-Selektoren zugewiesen.

```

...
<style>
①   body { font: 1em Verdana,Arial,sans-serif; }
②   p.hintergrund { background-color: yellow;
                     background-image:
                         url(patterns/img_08.gif);
                     background-position: 40px 50%;
                     height: 80px; }

③   .absatz1 { background-repeat: repeat-x; }
④   .absatz2 { background-repeat: repeat-y; }
⑤   .absatz3 { background-repeat: no-repeat; }
⑥   .absatz4 { background-repeat: repeat; }

</style>
...

```

- ① Die Standardeinstellungen für die gesamte Webseite werden definiert.

- ② Alle Absätze mit der Klasse `hintergrund` erhalten eine gelbe Hintergrundfarbe und eine Hintergrundgrafik. Die Grafik soll stets 40 Pixel von links und auf halber Höhe des Elements beginnen.
- ③ Der Hintergrund vom Absatz mit der Klasse `absatz1` wird horizontal wiederholt. Dies bedeutet, dass die horizontale Positionsangabe ignoriert wird. Der Hintergrund beginnt auf der halben Höhe des Absatzes.
- ④ Im Absatz mit der Klasse `absatz2` wird der Hintergrund vertikal wiederholt, sodass die vertikale Positionsangabe außer Acht gelassen wird. Der Hintergrund beginnt somit erst 40 Pixel vom linken Rand.
- ⑤ Hier wird der Hintergrund nicht wiederholt und die einzelne Grafik als Hintergrund angezeigt. Beide Positionsangaben werden beachtet.
- ⑥ Da im Selektor `absatz4` die Wiederholung der Hintergrundgrafik eingeschaltet ist, werden jegliche Positionsangaben ignoriert.

6.5 Hintergrund festsetzen

Mit der CSS-Eigenschaft `background-attachment` können Sie das Verhalten des Hintergrunds beim Scrollen festlegen. Ein Fixieren des Hintergrunds wird oft mit dem vom Druck bekannten Wasserzeichen gleichgesetzt.

background-attachment: Wert;

background-attachment: Wert;	
Werte	<ul style="list-style-type: none"> ✓ <code>scroll</code> = bewegt sich mit der Webseite beim Scrollen mit ✓ <code>fixed</code> = bleibt stehen; Wasserzeicheneffekt
Standardwert	<code>scroll</code>
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Beispiel: [kap06\background-attachment.htm](#)

Um das Scrollen des Hintergrunds zu vermeiden, können Sie die Position des Bilds fixieren.

	<pre><html lang="de"> <head> <title>Hintergrundgrafik</title> <style> body { background-image: url(patterns/logo-erde.gif); background-position: center center; background-repeat: no-repeat; background-attachment: fixed; }</pre>
① ② ③ ④	

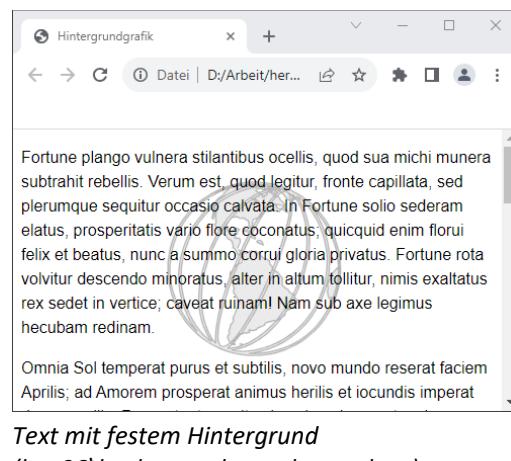
```

⑤      p      { font: 1rem Georgia,Verdana,Arial,helvetica,
           sans-serif; line-height: 1.5; }

    </style>
</head>
<body>
  <p>Fortune plango vulnera stilantibus ocellis, quod sua
     michi ...</p>
</body>
</html>

```

- ① Die Hintergrundgrafik weisen Sie dem Element body zu. Als Grafik verwenden Sie die Datei *logo-erde.gif*, die sich im Unterverzeichnis *patterns* befindet.
- ② Die Ausrichtung erfolgt horizontal und vertikal zentriert.
- ③ Die Grafik wird nur einmal angezeigt und durch die Angabe von *no-repeat* nicht für die gesamte Webseite gekachelt.
- ④ Damit die Grafik auch beim Scrollen der Webseite sichtbar bleibt, setzen Sie für die Eigenschaft *background-attachment* den Wert *fixed*.
- ⑤ Zum besseren Lesen des Inhalts weisen Sie eine 1,5-fache Zeilenhöhe zu.



Viele Möglichkeiten von CSS erschließen sich nicht auf den ersten Blick. Welche Effekte man mit *background-attachment* erreichen kann, zeigt Eric Meyer auf der englischen Seite <https://meyerweb.com/eric/css/edge/complexspiral/glassy.html>

6.6 Mehrfache Hintergründe verwenden

Wenn ein Hintergrund ein Element nicht komplett ausfüllt, weil Sie dies mit der Eigenschaft *background-repeat* so festgelegt haben oder weil ein Hintergrundbild teilweise transparent ist, können Sie die freie Fläche mit weiteren Hintergründen füllen.

Wollen Sie einem Element mehrere Hintergründe zuordnen, so sind die Angaben für die einzelnen Bilder durch Kommata voneinander zu trennen.

Beispiel: *kap06\multiple-background-images.htm*

```

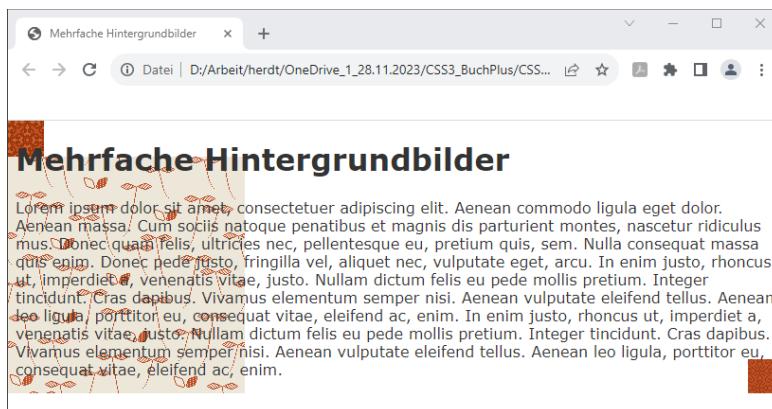
...
<style>
  body { background:
    url(patterns/img_03.gif) top left no-repeat,
    url(patterns/img_04.gif) bottom left no-repeat,

```

```
(3) url(patterns/img_09.gif) bottom right no-repeat;
}   </style>
...
```

Dem Rumpf des Dokumentes werden drei Hintergründe zugewiesen:

① *img_03.gif*, ② *img_04.gif* und ③ *img_09.gif*.



Die Datei „kap06\multiple-background-images.htm“

Welche eindrucksvollen Möglichkeiten sich mit mehrfachen Hintergründen ergeben, beschreibt Alex Walker in seinem Artikel „The Cicada Principle“ (<https://www.sitepoint.com/the-cicada-principle-and-why-it-matters-to-web-designers/>). Eine deutsche Übersetzung einer älteren Version des Artikels finden Sie unter <https://haunschild.de/css3/das-zikaden-prinzip-the-cicada-principle-und-warum-es-webdesigner-etwas-an geht/>.

6.7 Größe der Hintergrundgrafik

Sie können auch die Größe der verwendeten Hintergrundgrafik angeben. Dies erledigen Sie mit der Eigenschaft `background-size`.

```
background-size: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ auto = berechneter Wert ✓ Zahlenwert mit Maßeinheit (z. B. 10em oder 33%) ✓ cover ✓ contain
Standardwert	auto (Originalgröße des Bildes)
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Pixelbilder sind im Vergleich zu Texten sehr groß. Insbesondere, damit Ihre Webseite auf hochauflösenden Displays gut aussieht, sollten Sie hochwertige Bilder benutzen. Das kann vor allem für mobile Nutzer ein Problem sein. Zwar sind die mobilen Netze inzwischen so schnell, dass diese oft sogar als „Turbo“ für ein schwaches WLAN genutzt werden, Sie sollten aber bedenken, dass echte Flatrates für mobile Netze nach wie vor eher selten sind. Das bedeutet, dass Bilder beim Benutzer Kosten verursachen – umso mehr, je größer die zu übertragende Datenmenge ist. Da Hintergrundgrafiken reine Schmuckgrafiken sind (inhaltlich relevante Bilder gehören in die HTML-Struktur und sollten nicht per CSS eingefügt werden), nutzen Sie für Hintergründe besonders effiziente Formate, also möglichst SVG. Durch die unbegrenzte Skalierbarkeit von Vektorgrafiken sehen Bilder im SVG-Format zudem auf jedem Display gestochen scharf aus.

Beispiel: *kap06\background-size.htm*

```
...
<style>
    background-image: url(patterns/img_03.gif);
    background-repeat: no-repeat;
    background-size: 100px 200px;
}
</style>
...
```

Beispiel: *kap06\background-auto-size.htm*

```
...
<style>
body { background-image: url(patterns/img_03.gif);
       background-repeat: no-repeat;
       background-position: center center;
       background-size: 100% 100%, auto; }
</style>
...
```

Die Reihenfolge der Größenangabe ist immer Breite mal Höhe. Die Eigenschaft `auto` berechnet die Seitenlänge, für die `auto` angegeben ist, also die Breite, wenn `auto` zuerst angegeben wird, und die Höhe, wenn `auto` als zweiter Wert notiert wird. Bei dieser Berechnung kommt es zu keiner Streckung oder Stauchung. Wenn die Grafik den gesamten Container ausfüllen soll, in einer Richtung also 100% angegeben wird, kann die Hintergrundgrafik größer werden als der zugehörige Container. In diesen Fällen ist die Grafik nur teilweise sichtbar.

Schlüsselwörter

Schlüsselwörter helfen Ihnen dabei, die Grafik komplett anzuzeigen oder das Element komplett mit dem Hintergrund auszufüllen.

Beispiel `contain`

```
background-size: contain;
```

Verwenden Sie das Schlüsselwort `contain`, um die Grafik so groß wie möglich anzeigen zu lassen, ohne dass Teile davon abgeschnitten werden.

Beispiel `cover`

```
background-size: cover;
```

Dass Schlüsselwort `cover` stellt sicher, dass das gesamte ausgewählte Element mit der Grafik gefüllt wird. Dabei wird die Grafik weder gestaucht noch gestreckt, sodass in der Regel Teile der Grafik abgeschnitten werden.

Durch geschickte Positionsangaben für die Hintergrundgrafik können Sie sicherstellen, dass der wichtigste Ausschnitt des Bildes dennoch im sichtbaren Bereich bleibt.

6.8 Bereich für Hintergrundposition und Hintergrundanzeige festlegen

Sofern sie keine anderen Angaben machen, bedecken Hintergrundbilder den Inhalt eines Elementes und den Innenabstand. Das ist die Fläche innerhalb eines Rahmens.

Dieses Verhalten können Sie mit CSS ändern und angeben, ob der Hintergrund größer sein soll, also auch die für den Rahmen reservierte Fläche füllen soll. Sie können auch festlegen, dass nur der Inhalt mit einem Hintergrund gefüllt werden soll.

```
background-origin: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ <code>padding-box</code> = Element-Inhalt und Innenabstand (Standard) ✓ <code>border-box</code> = zusätzlich der Rahmen-Bereich ✓ <code>content-box</code> = nur die Fläche für den Element-Inhalt
Standardwert	<code>padding-box</code>
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Beispiel: `kap06\background-origin.htm`

Die Eigenschaft `background-origin` erlaubt Ihnen, den Hintergrund an den unterschiedlichen Bereichen einer CSS-Box auszurichten, also bündig zu

- ✓ Rahmen,
- ✓ Innenabstand (Standard) oder
- ✓ Inhalt,

und die entsprechende Fläche durch Kacheln, Stauchen oder Strecken zu füllen.

```

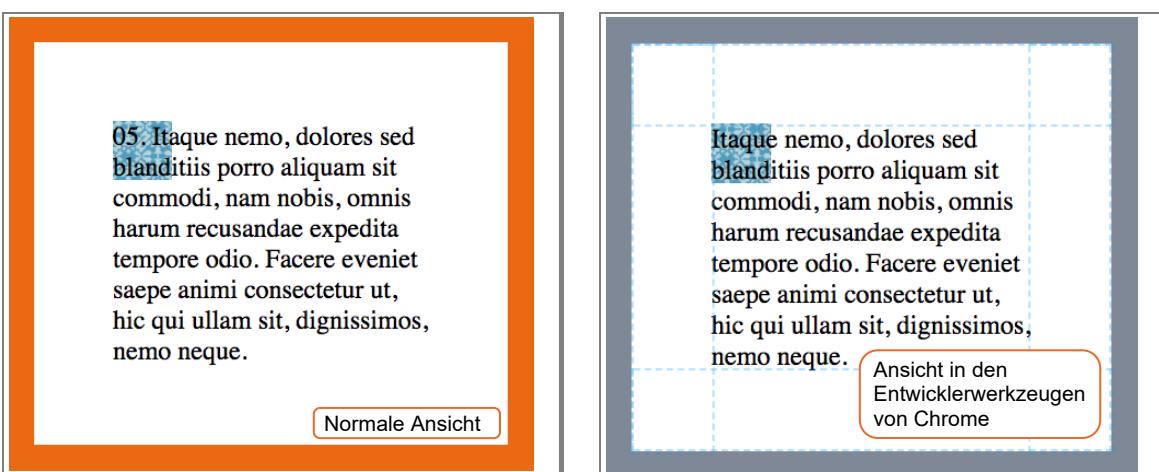
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>Hintergrundbereich festlegen</title>
    <style>
      ① .background-origin {
        border: 1em solid #eb6a27;
        padding: 3em;
      ② background-image: url("./patterns/img_02.gif");
      ③ background-repeat: no-repeat;
      ④ background-origin: content-box; }

      ⑤ </style>
    </head>

    <body>
      <p class="background-origin">Itaque nemo, dolores sed
      blanditiis porro aliquam sit commodi, nam nobis, omnis
      harum recusandae expedita tempore odio. Facere eveniet
      saepe animi consectetur ut, hic qui ullam sit,
      dignissimos, nemo neque.</p>
    </body>
  </html>

```

- ① Für das Element mit der Hintergrundgrafik werden ein Rahmen und ein Innenabstand festgelegt.
- ② Der Pfad zur Hintergrundgrafik wird angegeben.
- ③ Die Grafik wird nur einmal angezeigt und durch die Angabe von no-repeat nicht für das gesamte Element gekachelt. So wird gut sichtbar, wo diese platziert wird.
- ④ Die Angabe background-origin: content-box legt fest, dass der Hintergrund nur für den Bereich des Elements gilt, indem sich der eigentliche Inhalt (engl.: content) befindet.
- ⑤ Das Stylesheet und der Dokumentenkopf werden geschlossen und der Dokumentenrumpf mit dem eigentlichen Inhalt beginnt. Darin befindet sich ein Absatz mit der Klasse background-origin, auf den sich die Angaben des Stylesheets beziehen.



- ✓ Das Hintergrundbild wird im Inhaltsbereich platziert (ohne weitere Positionsangaben oben links).
- ✓ In den Entwicklerwerkzeugen von Chrome (rechte Abbildung) werden die einzelnen Bereiche des CSS-Box-Modells hervorgehoben: der Rahmen (`border`) wird grau dargestellt, die gestrichelten Linien zeigen die Grenze zwischen Innenabstand (`padding`) und Inhaltsbereich (`content`).

```
background-clip: Wert;
```

Mit der Eigenschaft `background-clip` können Sie festlegen, welcher Bereich, den Sie mit Hintergrundbildern oder -farben gefüllt haben, tatsächlich angezeigt werden soll.

Das bedeutet: Unabhängig davon, ob Sie mit der Kachelung einer Hintergrundgrafik am Außenbereich des Rahmens oder des Inhaltes beginnen, können Sie festlegen, dass Hintergründe in folgenden Bereichen des Elementes tatsächlich gezeigt werden soll:

Werte	<ul style="list-style-type: none"> ✓ <code>padding-box</code> = Element-Inhalt und Innenabstand (Standard) ✓ <code>border-box</code> = zusätzlich der Rahmen-Bereich ✓ <code>content-box</code> = nur die Fläche für den Element-Inhalt
Standardwert	<code>padding-box</code>
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Beachten Sie, dass sich Hintergründe immer hinter dem Rahmen befinden und so bei nicht durchsichtigen, durchgezogenen Rahmenlinien nicht gesehen werden können. Bei gepunkteten oder gestrichelten Rahmenlinien scheint der Hintergrund an den Stellen ohne Rahmenlinie durch.

6.9 Angabe des Hintergrunds in Kurzform

Anstatt jede Hintergrundeigenschaft einzeln anzugeben, können Sie die Kurzform verwenden.

```
background: bg-image || position [ / bg-size>]? || repeat-style  
|| attachment || box || box || 'background-color'
```

Die Reihenfolge der Angaben ist: Grafik, Wiederholung, Festsetzen, Position und Größe des Hintergrundbildes, Hintergrundfarbe. Die Werte werden durch ein Leerzeichen voneinander getrennt. Angaben mit einem Leerzeichen im Wert sind in Anführungszeichen zu setzen.

Sie können diese Angaben mehrfach machen, um mehrere Hintergründe in Ebenen anzugeben, dann darf ausschließlich die letzte Hintergrundebene eine Farbangabe enthalten. Die Box-Angaben beziehen sich auf `background-origin` und `background-clip`. Wenn Sie dort nur eine Angabe machen, wirkt die sich für beide Eigenschaften aus. Geben Sie hier zwei Werte an, steht der erste für `background-origin` und der zweite für `background-clip`.



Lernvideo: Hintergrund kontrollieren

6.10 CSS-Filter

Mit der CSS-Eigenschaft `filter` wenden Sie verschiedene visuelle Effekte auf HTML-Elemente wie etwa (Hintergrund-)Bilder an, unter anderem Unschärfe, Helligkeit, Kontrast, Sättigung, Farbton, Invertierung oder Graustufen. Dazu stehen verschiedene Funktionen zur Verfügung.

```
filter: funktion1(wert1) funktion2(wert2) ...;
```

Funktionen	<ul style="list-style-type: none"> ✓ <code>blur()</code> wendet ein Weichzeichnung an. ✓ <code>brightness()</code> bietet die Möglichkeit, die Helligkeit zu steuern (0% = komplett schwarz, 100% = unverändert, Werte über 100% = graduelle Aufhellung) ✓ <code>contrast()</code> bietet die Möglichkeit, den Kontrast zu steuern (0% = grau, 100% = unverändert, Werte über 100% = graduelle Kontraststeigerung) ✓ <code>drop-shadow()</code> wendet den Parameter <code><shadow></code> als Schlagschatten an, der den Konturen des Bildes folgt ✓ <code>grayscale()</code> konvertiert das Bild in Graustufen (100 % = vollständige Graustufen, 0% = unverändert) ✓ <code>hue-rotate()</code> ändert den Farbton des Elements. Der Wert <code><angle></code> definiert die Gradzahl um den Farbton-Farbkreis (von 0 bis 360). Ein Wert von 0 Grad lässt das Element unverändert. ✓ <code>invert()</code>: setzt die Elementfarben ins Negativ (100 % vollständige Invertierung, 0% unverändert) ✓ <code>opacity()</code> wendet eine Transparenz an (0% vollständig, 100% unverändert) ✓ <code>saturate()</code> verändert die Farbsättigung (0% Graustufen, 100% unverändert, Werte über 100% graduelle Erhöhung der Sättigung) ✓ <code>sepia()</code> konvertiert die Elementfarben in Sepia (100% = vollständig sepia, 0% keine Veränderung)
Standardwert	-
Vererbbar	Nein
Anwendbar auf	Alle Bild- und Containerelemente

Beispiel: `kap06\blur.htm`

Der `blur()`-Filter zeichnet das Element weich, indem die Pixel verwischt werden. Sie können dabei den Grad der Unschärfe als Pixelwert angeben. Je höher Sie den Wert setzen, desto unschärfer wird das Element.

```
<html lang="de">
<head>
  <title>Bild weichzeichnen</title>
```

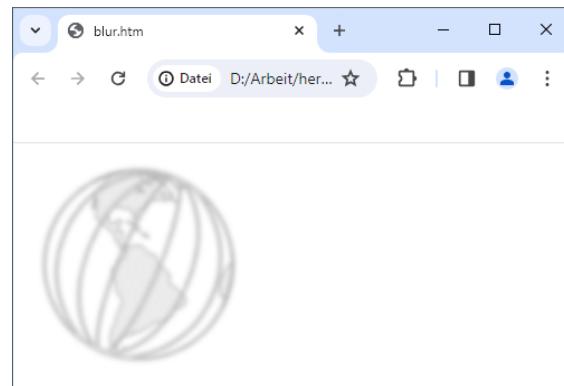
```

① <style>
    img {
        filter: blur(2px);
    }
</style>
</head>
<body>
② 
</body>
</html>

```

- ① Sie weisen dem HTML-Tag `img` die CSS-Filterfunktion `blur` mit dem Wert `2px` zu.
- ② Sie fügen mit dem HTML-Tag `img` ein Bild in den `body`-Bereich der Seite ein.

Die Werte für die Filterfunktionen können in verschiedenen Einheiten angegeben werden, wie z.B. Pixel (`px`) oder Prozent (%). Mit der Funktion `none` entfernen Sie alle Filtereffekte.



Beispiel: `kap06\drop-shadow.htm`

Bildelement weichzeichnen

Der `drop-shadow()`-Filter versieht das Element mit einem Schlagschatten, dessen Richtung und Intensität Sie steuern können.

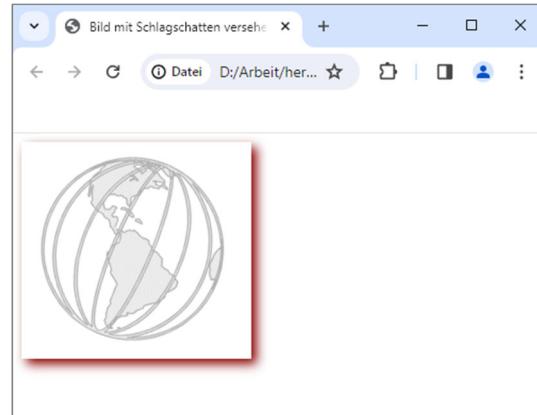
```

<html lang="de">
<head>
    <title>Bild weichzeichnen</title>
    ① <style>
        img {
            filter: drop-shadow(5px 5px 5px #8B0000);
        }
    </style>
</head>
<body>
    ② 
</body>
</html>

```

- ① Sie wenden dem HTML-Tag `img` die CSS-Filterfunktion `drop-shadow()` zu.
- ✓ Der erste px-Wert steht für den horizontalen Versatz des Schattens. Ein positiver Wert verschiebt den Schatten nach rechts, ein negativer nach links.
 - ✓ Der zweite px-Wert steht für den vertikalen Versatz. Ein positiver Wert verschiebt den Schatten nach unten, ein negativer Wert nach oben.
 - ✓ Der dritte px-Wert steht für den Weichzeichnungsgrad des Schattens. Je größer der Wert, desto unschärfer und damit großflächiger wird der Schatten.

- ② Sie fügen mit dem HTML-Tag `img` ein GIF-Bild in den `body`-Bereich der Seite ein.



Bildelement mit Schlagschatten versehen

Weitere Beispiele für Filtereffekte

Original	
<pre><style> img { filter: hue-rotate(90deg) ; } </style></pre>	
<pre><style> img { filter: brightness(150%) ; } </style></pre>	
<pre><style> img { filter: invert(100%) ; } </style></pre>	

Sie können auch mehrere Filter kombinieren, um komplexere Effekte zu erzielen (*kap06\filter-kombi.htm*): Die Filter werden stets in der angegebenen Reihenfolge zugewiesen.

<pre><style> img { filter: grayscale(80%) drop-shadow (8px 8px 5px grey) ; } </style></pre>	
---	--

Eine ausführliche CSS-Filterreferenz finden Sie unter anderem hier:

<https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/filter>

Eine weitere Möglichkeit, um Ihren HTML-Elementen grafische Effekte zuzuweisen, sind die CSS-Blendmodi. Darüber informieren Sie sich hier (in englischer Sprache):

https://www.w3schools.com/cssref/pr_mix-blend-mode.php

6.11 Übungen

Übung 1: Fixierter Hintergrund

Level		Zeit	ca. 15 min
Ergebnisdatei	kap06\uebung1.htm		

1. Erzeugen Sie eine Webseite, in der Sie am unteren Rand eine Hintergrundgrafik einfügen. Diese soll von links nach rechts verlaufen und beim Scrollen fixiert bleiben. Füllen Sie die Webseite mit Inhalt.

Übung 2: Formular mit Hintergrundgrafik

Level		Zeit	ca. 30 min
Ergebnisdateien	kap06\uebung2.htm , kap06\uebung3.htm		

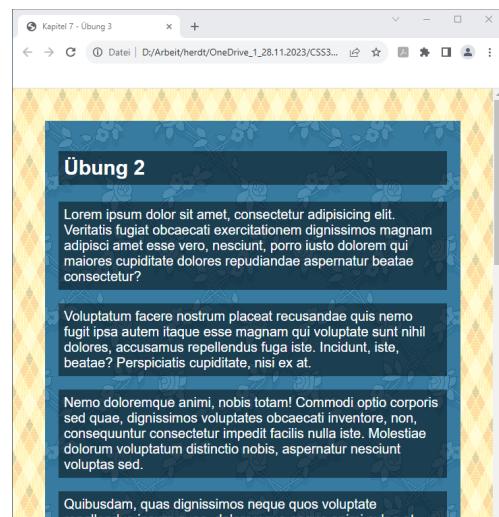
1. Erstellen Sie eine Seite mit einem Artikel und einer Überschrift und fügen Sie beliebigen Text hinzu. Führen Sie die nachfolgenden Formatierungen durch.

Gesamter Dokumenten-Rumpf:

Innenabstand entsprechend der doppelten Schriftgröße; Schrift auf das 1,25 fache der normalen Größe vergrößern; Helvetica oder ein geeigneter Ersatz; eine sich wiederholende Hintergrundgrafik, die nicht fixiert ist.

Artikel: Innenabstand entsprechend der doppelten Schriftgröße; gekachelte und fixierte Hintergrundgrafik.

Absätze und Überschrift: Sorgen Sie mit einer geeigneten Kombination von Hintergrundfarben und Schriftfarbe für eine gute Lesbarkeit der Texte.



Mögliche Ergebnisdarstellung der Dateien „kap06\uebung2.htm“ und „kap06\uebung3.htm“ (Welche Farben und Hintergründe Sie verwenden, ist Ihnen freigestellt.)

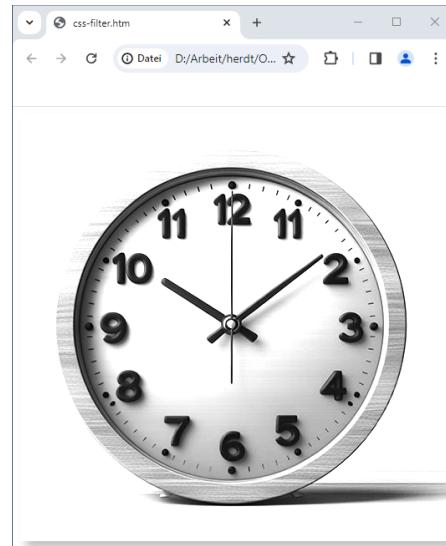
2. Optimieren Sie die Stylesheet-Definitionen durch das Zusammenfassen von Eigenschaften.
Die Darstellung von *uebung2.htm* und *uebung3.htm* soll identisch sein.

Übung 3: Bild anpassen

Level		Zeit	ca. 10 min
Übungsdatei	<i>Kap06/uhr.jpg</i>		
Ergebnisdatei	<i>Kap06/css-filter.htm</i>		

1. Erstellen Sie eine Webseite und fügen Sie das Bild *uhr.jpg* ein.
2. Stellen Sie das Bild in Graustufen, aufgehellt und mit einem Schlagschatten dar.

Achten Sie dabei darauf, dass nur das Bild selbst aufgehellt werden soll, jedoch nicht der Schlagschatten!



Mögliches Ergebnis

7

Listen gestalten

7.1 Einleitung zu Listen

Es gibt zahlreiche Anwendungsmöglichkeiten von Listen. Sie können durch eine Nummerierung die Rangfolge festlegen oder die Reihenfolge eines Arbeitsablaufs kennzeichnen. Unnummerierte Listen verwenden Sie für alle anderen Aufzählungen wie Linklisten, zum Beispiel die Navigation einer Site.

Eine Besonderheit sind Beschreibungslisten, mit denen Sie Begriffen Erklärungen mitgeben können.

Mit den folgenden CSS-Eigenschaften können Sie die Aufzählungszeichen von nummerierten und unnummerierten Listen formatieren.

- ✓ `list-style-type`
- ✓ `list-style-image`
- ✓ `list-style-position`
- ✓ `list-style`

7.2 Listentyp

Neben den üblichen Formatierungen für Elemente können Sie zusätzlich einige Eigenschaften beeinflussen, die exklusiv der Gestaltung von Listen dienen. Zahlreiche Werte stehen Ihnen allein zur Beeinflussung des Listentyps zur Verfügung.

```
list-style-type: Wert;
```

Werte für 	✓ none = kein Aufzählungszeichen ✓ circle = ungefüllter Kreis; nur Rahmen ✓ square = gefülltes Quadrat ✓ disc = gefüllter Kreis
Werte für 	✓ lower-alpha = Kleinbuchstaben (a, b, c ...) ✓ upper-alpha = Großbuchstaben (A, B, C ...) ✓ lower-roman = römische Zahlen, Kleinschreibung (i, ii, iii ...) ✓ upper-roman = römische Zahlen, Großschreibung (I, II, III ...) ✓ lower-greek = kleine griechische Buchstaben (α, β, γ ...) ✓ lower-latin = kleine ASCII-Zeichen (a, b, c ...) ✓ upper-latin = große ASCII-Zeichen (A, B, C ...) ✓ decimal = Dezimalzahlen (1, 2, 3 ...) ✓ decimal-leading-zero = Dezimalzahlen mit führender 0 (01, 02, 03 ...)
Standardwerte	disc für und decimal für
Vererbbar	Ja
Anwendbar auf	Alle ol- und ul-Elemente und das später folgende display:list-item

Für die Eigenschaft list-style-type existieren weitere Werte wie armenian (für die armenische Nummerierung) oder georgian (für die georgische Nummerierung). Allerdings werden nicht alle von jedem Browser unterstützt. Eine vollständige Liste finden Sie beim W3C unter <https://www.w3.org/TR/css-lists-3/>.

Den Startwert einer Aufzählung legen Sie über das HTML-Attribut start fest. Hierfür gibt es keine separate Eigenschaft in CSS.

Beispiel: kap07\list-type.htm

Zur Veranschaulichung wird eine Webseite angelegt, bei der einige verschiedene Werte zur Anwendung kommen. Abgedruckt werden nur vereinzelte Werte, das komplette Beispiel finden Sie in der Datei *kap07/list-type.htm*.

①	<pre><p>Nummerierung mit vorangestellter Null</p> <ol style="list-style-type: decimal-leading-zero;" start="9"> Frühstück Mittag Abendessen <!-- ... --> <p>lower-greek</p></pre>
---	--

```

② <ol start="9" style="list-style-type: lower-greek;">
    <li>Frühstück</li>
    <li>Mittag</li>
    <li>Abendessen</li>
</ol>
<!-- ... -->
<p>lower-latin</p>
③ <ol start="99" style="list-style-type: lower-latin;">
    <li>Frühstück</li>
    <li>Mittag</li>
    <li>Abendessen</li>
</ol>
<!-- ... -->
<p>georgian</p>
④ <ol start="9" style="list-style-type: georgian;">
    <li>Frühstück</li>
    <li>Mittag</li>
    <li>Abendessen</li>
</ol>

```

- ① Mit dem Tag `` leiten Sie eine nummerierte Aufzählung ein, die über das Attribut `start` mit dem Zählerwert 9 beginnen soll. Als Listentyp verwenden Sie eine Nummerierung, bei der einstelligen Zahlen eine führende 0 vorangestellt wird.
- ② Hier weisen Sie der Aufzählung als Zeichen die griechischen Buchstaben zu. Diese sind α , β , γ , δ , ϵ , ζ , η , θ , ι , κ , λ usw.
- ③ Der Wert `lower-latin` bewirkt im Grunde genommen dieselbe Formatierung wie der Wert `lower-alpha`. Die Aufzählungen erscheinen als Kleinbuchstaben. Wird der Zählerwert 26, also der Buchstabe `z`, überschritten, beginnt die zweite Stelle mit dem Zeichen `a`. Die Zahl 99 wird z. B. als `cu` dargestellt.
- ④ Für eine Aufzählung mit georgischen Schriftzeichen geben Sie den Wert `georgian` an.

Das Aussehen von Listen mittels `list-style-type` beeinflussen

<code>none</code>	<code>circle</code>	<code>square</code>	<code>disc</code>
Frühstück Mittag Abendessen	○ Frühstück ○ Mittag ○ Abendessen	■ Frühstück ■ Mittag ■ Abendessen	● Frühstück ● Mittag ● Abendessen
Nummerierung mit vorangestellter Null	lower-greek	lower-latin	upper-alpha
09. Frühstück 10. Mittag 11. Abendessen	τ. Frühstück κ. Mittag λ. Abendessen	cu. Frühstück cv. Mittag cw. Abendessen	I. Frühstück J. Mittag K. Abendessen
armenian	georgian		
Ժ. Frühstück Ժ. Mittag Ժ. Abendessen	օ. Frühstück օ. Mittag օ. Abendessen		

Verschiedene Arten der Nummerierung (`kap07\list-type.htm`)

7.3 Aufzählungszeichen einrücken

Die Aufzählungszeichen der verschiedenen Listen stehen standardmäßig immer etwas abgesetzt vom eigentlichen Listeneintrag. Dies wird besonders dann sichtbar, wenn der Text des Eintrags etwas länger ist und automatisch auf die nächsten Zeilen umbricht. Ein Beispiel sehen Sie in der nebenstehenden Abbildung.

Die CSS-Eigenschaft `list-style-position` erlaubt es Ihnen, die Einrückung der Aufzählungszeichen zu beeinflussen, sodass der Absatz und die Zeichen linksbündig abschließen. Ersichtlich wird dies in dem zweiten Abschnitt der Abbildung.

```
list-style-position: Wert;
```

Abgesetzte Aufzählungszeichen

1. Dies ist ein normales Aufzählungszeichen. Ama me fideliter! Fidem mean nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.
2. Genau wie dieses hier. Ama me fideliter! Fidem mean nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.
3. Oder jenes. Ama me fideliter! Fidem mean nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.

Eingerückte Aufzählungszeichen

- Dies ist ein eingerücktes Aufzählungszeichen. Ama me fideliter! Fidem mean nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.
- Genau wie dieses hier. Ama me fideliter! Fidem mean nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.
- Oder jenes. Ama me fideliter! Fidem mean nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.

Verschiedene Einrückungen ([kap07\list-pos.htm](#))

Werte	✓ <code>inside</code> = mit dem Text eingerückte Grafiken oder Symbole ✓ <code>outside</code> = vom Text abgesetzte Grafiken oder Symbole
Standardwert	<code>outside</code>
Vererbbar	Ja
Anwendbar auf	Alle <code>ol</code> - und <code>ul</code> -Elemente und das später folgende <code>display:list-item</code>

7.4 Listengrafik ändern

Mit der CSS-Eigenschaft `list-style-image` geben Sie eigene Aufzählungszeichen an. So können Sie die bisherigen Grafiken wie Kreis, Quadrat usw. durch eigene Grafiken ersetzen.

```
list-style-image: url( Wert );
```

Werte	✓ <code>none</code> ✓ URL
Standardwert	<code>none</code>
Vererbbar	Ja
Anwendbar auf	Alle <code>ol</code> - und <code>ul</code> -Elemente und das später folgende <code>display:list-item</code>

Die Größe des Aufzählungszeichens hängt immer von der Originalgröße des Bildes ab. Es gibt keine Größeneinschränkung bei den verwendeten Grafiken. Allerdings vergrößern sich die Grafiken (im Gegensatz zu den `list-item`-Werten) nicht mit, wenn Sie die Schrift im Browser vergrößern. Beim Zoomen der gesamten Webseite werden Bitmap-Grafiken zudem „pixelig“, weshalb sich die Verwendung von skalierbaren Vektorgrafiken (SVG) anbietet.

Beispiel: *kap07\list-style-image.htm*

Einer nicht nummerierten Liste wird im Folgenden eine Grafik zugewiesen. Hierfür legen Sie im Kopfbereich der Webseite die beiden Typ-Selektoren `ol` und `ul` mit den CSS-Eigenschaften für Aufzählungen an.

```

<style>
①   ol { list-style-position: inside; }
②   ul {
        list-style-position: outside;
        list-style-image: url(images/lnr-diamond.svg); }
</style>
```

- ① Für die nummerierte Aufzählung wählen Sie eine eingerückte Aufzählung. Durch die automatische Nummerierung benötigen Sie keine separate Grafik. Deshalb geben Sie keine Aufzählungsgrafik an.
- ② Der nicht nummerierten Liste weisen Sie die Grafik *burst.png* aus dem Unterverzeichnis *images* zu. Damit sich diese etwas vom Text abhebt, wählen Sie die standardmäßige Art der Einrückung `outside`.

Aufzählungszeichen ändern

Eingerückte Aufzählungszeichen ohne Grafik

1. Dies ist ein eingerücktes Aufzählungszeichen. Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.
 2. Genau wie dieses hier. Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.
 3. Oder jenes. Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.

Abgesetzte Aufzählungszeichen mit Grafik

⌚ Dies ist ein normales Aufzählungszeichen. Als Grafik wird ein eigenes Symbol verwendet. Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.
 ⌚ Genau wie dieses hier. Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.
 ⌚ Oder jenes. Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.

Grafik als Aufzählungszeichen (kap07\list-style-image.htm – Icon von <https://linearicons.com>)

Mit `display: list-item` können Sie jedes beliebige Element wie einen Listeneintrag aussehen lassen. Dann funktionieren auch die hier genannten Formatierungen (siehe Beispieldatei *kap07\display_list-item.htm*).

7.5 Kurzform der Aufzählung

Um nicht jede Aufzählungseigenschaft einzeln angeben zu müssen, können Sie die Kurzform verwenden.

```
list-style: list-style-type, list-style-position, list-style-image;
```

Die Reihenfolge der Angaben ist: Art der Aufzählung, Position und Listengrafik. Die Werte werden durch ein Leerzeichen voneinander getrennt. Angaben mit einem Leerzeichen im Wert, speziell bei der Angabe der Grafik, sind in Anführungszeichen zu setzen.

7.6 Übung

Bebilderte Aufzählung

Level		Zeit	ca. 10 min
Ergebnisdatei	<i>kap07\uebung1.htm, link.png</i>		

1. Erstellen Sie eine Aufzählung mit einer beliebigen Grafik. Diese soll im Text eingerückt dargestellt werden.

Eingerückte Aufzählungszeichen mit Grafik

➡ Dies ist ein eingerücktes Aufzählungszeichen.
 Fortune plango vulnera stilantibus ocellis, quod sua
 michi munera subtrahit rebellis. Verum est, quod
 legitur, fronte capillata, sed plerumque sequitur occasio
 calvata. In Fortune solio sederam elatus, prosperitatis
 vario flore coconatus; quicquid enim florui felix et
 beatus, nunc a summo corri gloria privatus. Fortune
 rota volvitur descendendo minoratus, alter in altum tollitur,
 nimis exaltatus rex sedet in vertice; caveat ruinam! Nam
 sub axe legimus hecubam redinam.

➡ Genau wie dieses hier. Omnia Sol temperat purus et
 subtilis, novo mundo reserat faciem Aprilis; ad Amorem
 prosperat animus herilis et iocundis imperat deus
 puerilis. Rerum tanta novitas in solemni vere et veris
 auctoritas iubet nos gaudere, vias prebet solitas, et in tue
 vere fides est et probitas tuum retinere. Ama me
 fideliter! Fidem meam nota: de corde totaliter et ex
 mente tota sum presentialiter absens in remota.

Ergebnisdatei *kap07\uebung1.htm*

8

Abstände und Rahmen

8.1 Einführung in das Box-Modell

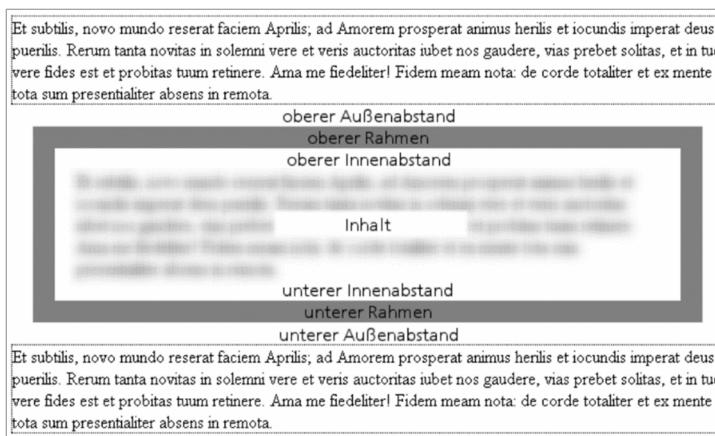
Besonders wichtig für ein gutes Layout einer Webseite ist die Wahl der Abstände und Zwischenräume, die den eigentlichen Inhalt umgeben. Neben den bisher bekannten Möglichkeiten, Abstände zu beeinflussen (line-height für den Zeilenabstand, letter-spacing für den Zeichenabstand und word-spacing für den Wortzwischenraum), stehen Ihnen weitere Eigenschaften zur Verfügung.

Alle Blockelemente können Sie sich als Kästen vorstellen, jeder Kasten enthält folgende Elemente:

- ✓ den Inhalt eines Elementes (Texte oder Grafiken),
- ✓ den Innenabstand von Text oder Grafik zum Rahmen (padding),
- ✓ den Rahmen (border),
- ✓ die Kontur (outline),
- ✓ der Außenabstand um den Textblock herum (margin).

Das Zusammenspiel dieser Eigenschaften wird **Box-Modell** genannt.

Eine Sonderrolle nimmt outline ein, weil mit outline gezeichnete Konturen keinen Raum beanspruchen. Sie fallen mit den Außenabständen zusammen. Konturen werden außerhalb des Rahmens gezeichnet.



Das Box-Modell, dargestellt an einem Absatz (kap08\boxmodell.htm)

Folgende CSS-Eigenschaften stehen Ihnen zur Beeinflussung der Rahmen und Abstände zur Verfügung:

- ✓ border-width
- ✓ border-style
- ✓ border-color
- ✓ border-image
- ✓ border-radius
- ✓ border
- ✓ padding
- ✓ margin
- ✓ outline

8.2 Rahmen

Rahmenbreite

Das auffälligste Stilmittel, um einen Zwischenraum zu erzeugen, ist die Verwendung eines Rahmens, auch Umrandung oder Rand genannt. Legen Sie zunächst die Größe des Rahmens fest:

border-width: Wert;

Werte	<ul style="list-style-type: none"> ✓ thin = dünne Linie ✓ medium = normale Linie ✓ thick = dicke Linie ✓ Zahlenangaben mit entsprechender Maßeinheit
Standardwert	medium
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Rahmenstil

Wenn Sie zur Formatierung eines Elements nur die CSS-Eigenschaft border-width angeben, wird diese nicht dargestellt, da standardmäßig der Rahmen auf nicht sichtbar gesetzt ist. Sie müssen zusätzlich angeben, wie der Rahmen auszusehen hat.

border-style: Wert;

Werte	<ul style="list-style-type: none"> ✓ none = nicht sichtbarer Rand ✓ solid = durchgehende Linie ✓ double = zwei Linien ✓ dotted = gepunktete Linie ✓ dashed = gestrichelte Linie ✓ groove = 3-D-Effekt ✓ ridge = 3-D-Effekt (Umkehrung von groove) ✓ inset = oben und links wird die Farbe dunkler gesetzt ✓ outset = unten und rechts wird die Farbe dunkler gesetzt
-------	---

Standardwert	None
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Die verschiedenen Angaben haben folgendes Aussehen:

border-style: none = nicht sichtbarer Rand
border-style: solid = durchgehende Linie
border-style: double = zwei Linien
border-style: dotted = gepunktete Linie
border-style: dashed = gestrichelte Linie
border-style: groove = 3D-Effekt
border-style: ridge = 3D-Effekt (Umkehrung von groove)
border-style: inset = oben und links wird die Farbe dunkler gesetzt
border-style: outset = unten und rechts wird die Farbe dunkler gesetzt

Verschiedene Arten von Rahmen (*kap08\border-styles.htm*)

Rahmenfarbe

Die Angabe von `border-width` und `border-style` genügt dem Browser, um einen Rahmen zu zeichnen. Dieser wird dann in der Schriftfarbe dargestellt. Um selbst eine Farbe anzugeben, benutzen Sie `border-color`.

border-color: Wert;

Werte	<ul style="list-style-type: none"> ✓ transparent = Schlüsselwort für nicht sichtbare Rahmen ✓ Schlüsselwort als Farbname ✓ Hexadezimal: #RRGGBB oder #RGB ✓ Dezimalwert: rgb (rrr, ggg, bbb) ✓ Prozentwert: rgb (rrr.rr%, ggg.gg%, bbb.bb%)
Standardwert	Nein, abhängig vom Browser
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Mit dem Schlüsselwort `transparent` als Farbangabe wird ein unsichtbarer Rahmen erstellt. Solch ein Rahmen unterscheidet sich von der Stilangabe `none` dadurch, dass der Abstand für den Rahmen trotzdem eingehalten und nicht (wie bei `none`) auf eine Breite von 0 Pixel gesetzt wird. Außerdem wird dadurch das Zusammenfallen von Außenabständen verhindert.

Beispiel: *kap08\border-wsc.htm*

Verschiedene Werte für Rahmenbreite, Rahmenstil und Rahmenfarbe ermöglichen Ihnen eine Vielzahl von Darstellungsmöglichkeiten. Es werden drei Kombinationen, stellvertretend für Schmuckräder, erstellt.

```
<html>
<head>
    <title>Rahmen - Stil-, Farb- und Breitenauswahl</title>
    <style>
        p { font: 1em Helvetica, Arial, Geneva, sans-serif; }
        ① .dottedborder { border-style: dotted;
                            border-color: rgb(100%,0%,0%);
                            border-width: 1px; }
        ② .insetborder { border-style: inset;
                            border-color: #000080;
                            border-width: 5px; }
        ③ .doubleborder { border-style: double;
                            border-color: rgb(0,128,0);
                            border-width: 10px; }
    </style>
</head>
<body>
    <h2>Verschiedene Stile, Farben und Breiten</h2>
    ④ <p class="dottedborder">Ein roter, gepunkteter, 1 Pixel breiter Rahmen...</p>
    <p class="insetborder">Ein blauer, 5 Pixel breiter Rahmen, oben und links etwas dunkler...</p>
    <p class="doubleborder">Ein grüner, 10 Pixel breiter, doppelter Rahmen...</p>
</body>
</html>
```

- ① Ein Element, das diese Klasse zugewiesen bekommt, erhält einen gepunkteten Rahmen in der Farbe Rot und mit einer Breite von einem Pixel.
- ② Hier wird der Rahmen so festgelegt, dass der obere und linke Rahmen automatisch etwas dunkler als der rechte und untere Rahmen dargestellt werden. Damit dieser Effekt sichtbar wird, stellen Sie die Rahmenbreite auf fünf Pixel ein. Als Rahmenfarbe verwenden Sie einen Blauton.
- ③ Für das dritte Element wählen Sie eine doppelte Rahmenlinie in grüner Farbe. Den Rahmen setzen Sie mit zehn Pixeln besonders breit, um den Abstand der Doppellinien sichtbar zu machen.
- ④ Den drei Absätzen im HTML-Dokument weisen Sie die unterschiedlichen Klassen zu.

Verschiedene Stile, Farben und Breiten

Ein roter, gepunkteter, 1 Pixel breiter Rahmen.

Fortune plango vulnera stilantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata. In Fortune solio sederam elatus, prosperitatis vario flore coconatus; quicquid enim florui felix et beatus, nunc a summo corru gloria privatus. Fortune rota volvitur descendere minoratus.

Ein blauer, 5 Pixel breiter Rahmen, oben und links etwas dunkler.

Omnia Sol temperat purus et subtilis, novo mundo reserat faciem Aprilis; ad Amorem prosperat animus herilis et iocundis imperat deus puerilis. Rerum tanta novitas in solemni vere et veris auctoritas iubet nos gaudere, vias prebet solitas, et in tue vere fides est et probitas tuum retinere. Ama me fideliter! Fidem meam nota.

Ein grüner, 10 Pixel breiter, doppelter Rahmen.

Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota. Qui autem auscultare nolet, exsurgat foras, ut sit, ubi sedat ile qui auscultare vult. Rerum tanta novitas in solemni vere et veris auctoritas iubet nos gaudere, vias prebet solitas, et in tue vere fides est et probitas tuum retinere.

Verschiedene Rahmenstile (kap08\border-wsc.htm)

8.3 Separate Wertangaben

Bisher haben Sie für jede Eigenschaft der Rahmen nur einen Wert angegeben, der für alle vier Seiten gilt. Sie können aber für jede Seite einen separaten Wert angeben. Dadurch haben Sie die Möglichkeit, für jede Seite (oben, rechts, unten, links) unterschiedlich breite und verschieden aussehende Ränder zu gestalten.

Ein Wert	Dieser Wert ist gültig für alle vier Seiten.
Zwei Werte	Der erste Wert steht für die obere und untere Seite, der zweite Wert ist für die linke und rechte Seite.
Drei Werte	Der erste Wert steht für die obere, der zweite Wert für die linke und rechte und der dritte Wert für die untere Seite.
Vier Werte	Jeder Wert steht für eine Seite. Die Reihenfolge, die oben beginnt und in Uhrzeigerrichtung verläuft, ist: oben, rechts, unten, links.

Beispiel: kap08\separatewerte.htm

Der Auszug aus einer Stylesheet-Definition zeigt Ihnen die Angabe der verschiedenen Werte. Für die Rahmenbreite werden zwei Werte angegeben, für die Rahmenfarbe jeweils ein Wert pro Seite und bei den Rahmenbreiten wieder nur drei Werte.

```
p { border-style: solid dotted;
    border-color: rgb(100%,0%,0%) green #00FFFF #FF00FF;
    border-width: 1px 3mm thick; }
```

Das Ergebnis der Formatierung sieht so aus:

- Stil: oben und unten durchgehende Linie;
- Farbe: oben rot, rechts grün, unten cyan, links magenta;
- Breite: oben 1 Pixel, links und rechts 3 mm, unten dicke Linie (thick)

Sie können Stile auch exklusiv für eine einzelne Seite angeben:

CSS-Eigenschaft	Beispiel und Auswirkung
border-top-width border-right-width border-bottom-width border-left-width	<pre>.rahmenbreite { border-style: solid; border-color: rgb(100%, 0%, 0%); border-top-width: 15px; border-right-width: 10px; border-bottom-width: 5px; border-left-width: 0px; }</pre> <p>Rahmenbreiten: oben: 15 Pixel, rechts: 10 Pixel, unten: 5 Pixel, links: kein Rahmen</p>
border-top-style border-right-style border-bottom-style border-left-style	<pre>.rahmenstil { border-color: #00FFFF; border-width: 10px; border-top-style: solid; border-right-style: double; border-bottom-style: dashed; border-left-style: none; }</pre> <p>Rahmenstil: oben: durchgehend, rechts: doppelt, unten: gestrichelt, links: kein Rahmen</p>
border-top-color border-right-color border-bottom-color border-left-color	<pre>.rahmenfarbe { border-style: solid; border-width: 10px; border-top-color: rgb(0, 255, 0); border-right-color: red; border-bottom-color: #00FFFF; border-left-color: rgb(100%, 0%, 100%); }</pre> <p>Rahmenfarbe: oben: grün, rechts: rot, unten: cyan, links: magenta</p>

8.4 Kurzform von Rahmen

Die verschiedenen Eigenschaftswerte können Sie auch in Kurzform angeben.

border: border-width border-style border-color;
--

Die Reihenfolge der Angaben ist: Rahmenbreite, Rahmenstil und Rahmenfarbe. Die Werte werden durch ein Leerzeichen voneinander getrennt. Die Breitenangabe und die Farbangabe sind optional. Der Wert `border-width` kann jedoch nur im Zusammenhang mit dem Rahmenstil angegeben werden, da die Eigenschaft `border-style` den Startwert `none` hat und der Rahmen somit nicht sichtbar ist.

Auch für die Rahmeneigenschaften mit den separaten Angaben der Seiten können Sie die Werte zusammenfassen.

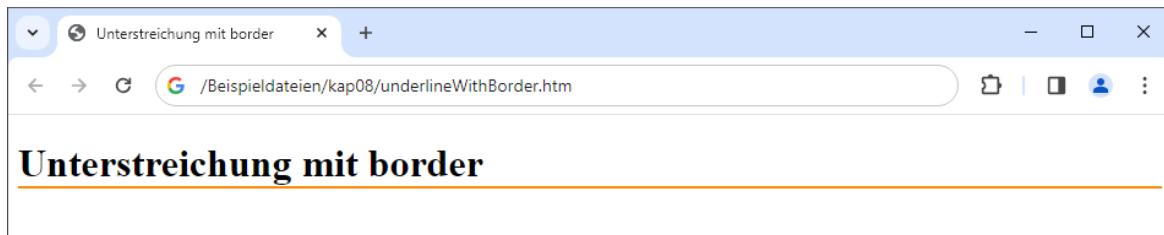
```
/* oberer Rahmen */
border-top: border-width border-style border-color;

/* rechter Rahmen */
border-right: border-width border-style border-color;

/* unterer Rahmen */
border-bottom: border-width border-style border-color;

/* linker Rahmen */
border-left: border-width border-style border-color;
```

So können Sie z. B. mit der Angabe von `border-bottom: .125rem solid #f80;` eine Überschrift unterstreichen. Diese Unterstreichung ist so breit wie das Element, erstreckt sich also über die gesamte Breite – auch wenn der Text kürzer ist. Eine so gestaltete Unterstreichung wird selten mit den ebenfalls standardmäßig unterstrichenen Links verwechselt.



kap08\underlineWithBorder.htm

8.5 Konturen mit `outline`

Mit dieser Eigenschaft legen Sie eine Kontur um ein Element. Die Kontur (`outline`) unterscheidet sich vom Rahmen wie folgt:

- ✓ Die Kontur nimmt keinen Platz in Anspruch.
- ✓ Die Kontur muss nicht rechteckig sein.
- ✓ Wird eine Kontur um ein Inline-Element gezeichnet und das Inline-Element wird umbrochen, so entstehen mehrere Boxen, die komplett von einer sichtbaren Linie umgeben werden – beim Rahmen (`border`) bleiben ein oder zwei Seiten des Rahmens offen.
- ✓ Separate Seitenangaben wie `top`, `right`, `bottom` und `left` sind nicht möglich. Somit können Sie auch keine speziellen Werte für die einzelnen Seiten angeben. Da ansonsten die Eigenschaften und Werte denen der Rahmen entsprechen, wird hier nicht näher darauf eingegangen.

Unterschiede zwischen `border` und `outline` bei Inline-Elementen

LOREM IPSUM DOLOR SIT AMET, CONSETETUR SADIPSCING ELITR. SED DIAM **DIESER TEXT HIER BESETZT EINEN RAHMEN (BORDER)** NONUMY EIROMOD TEMPOR INVIDUNT UT LABORE ET DOLORE MAGNA ALIQUYAM ERAT, SED DIAM VOLUTPUA.

DUIS AUTEM VEL EUM IRIURE DOLOR IN HENDERIRIT IN VULPIMATE VELIT ESSE MOLESTIE CONSEQUAT. **DIESER TEXT HIER BESETZT EINE KONTUR (OUTLINE)** VEL ILLUM DOLORE EU FEUGAIT NULLA FACILISIS AT VERO EROS ET ACCUMSAN ET IUSTO ODIO DIGNISSIM QUI BLANDIT PRAESENT LUPTATUM ZZRIL DELENIT AUGUE DUIS DOLORE TE FEUGAIT NULLA FACILISI.

*border und outline bei Inline-Elementen
(kap08/borderOutlineDiff.htm)*

```
outline-width : Wert;          /* wie bei border-width */
outline-style : Wert;         /* wie bei border-style */
outline-color : Wert;         /* wie bei border-color */
outline:                  outline-width outline-style outline-color;
```

8.6 Abgerundete Ecken

Mit **border-radius** können Sie Elemente mit abgerundeten Ecken versehen.

```
border-radius: Wert1 Wert2 Wert3 Wert4;
```

Werte	Zahlen mit Angabe einer Maßeinheit wie em, px oder %
Standardwert	Nein, Standard sind nicht abgerundete Ecken
Vererbbar	Nein
Anwendbar auf	Elemente mit Rahmen (außer Tabellen)

Als Wert können Sie einen Radius angeben, der für die Größe des Bogens steht. Dies können wieder Angaben in Pixeln, em, Prozent oder anderen Einheiten sein.

Die vier Werte stehen für die vier Ecken in folgender Reihenfolge: oben links, oben rechts, unten rechts, unten links; die Angaben werden also im Uhrzeigersinn gemacht. Sie können folgende Wertepaare angeben.

```
border-radius: 5px;
```

Alle 4 Ecken haben einen Radius von 5 Pixeln.

```
border-radius: 5px 20px;
```

Die Ecken links oben und rechts unten haben einen Radius von 5 Pixeln; rechts oben und links unten beträgt er 10 Pixel.

```
border-radius: 5px 20px 40px;
```

Die Ecke links oben hat einen Radius von 5 Pixeln. Der zweite Wert gilt für rechts oben und links unten. Die Ecke rechts unten hat einen Radius von 20 Pixeln.

```
border-radius: 5px 20px 40px 60px;
```

Sind vier Werte angegeben, werden diese im Uhrzeigersinn angewendet, beginnend bei der Ecke links oben.

Abgerundete Ecken mit border-radius

border-radius: 5px;

border-radius: 5px 20px;

border-radius: 5px 20px 40px;

border-radius: 5px 20px 40px 60px;

kap08\border-radius.htm

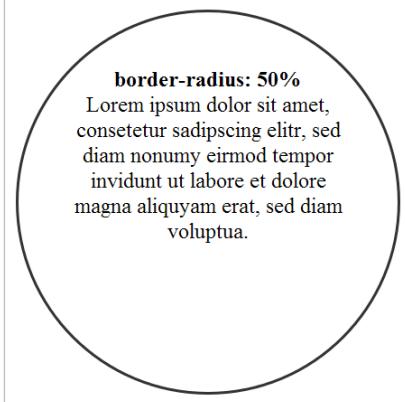
Wenn Sie für eine quadratische Box einen Rahmenradius von 50 % angeben, treffen sich die abgerundeten „Ecken“ und man erhält einen Kreis (Rechtecke werden so zu Ovalen).

```
<html lang="de">
  <head>
    <title>Einen Kreis zeichnen mit border-radius</title>
    <style>
      p {
        border: 2px solid #333;
        padding: 40px;
        width: 200px;
        height: 200px;
        border-radius: 50%;
        text-align: center; }
    </style>
  </head>
  <body>
    <h1>Einen Kreis zeichnen mit border-radius</h1>
    <p>
      <b>border-radius: 50%</b>
      Duis autem ...
    </p>
  </body></html>
```

Absätze sollen identische Angaben für Breite und Höhe haben. Der Radius der Abrundung soll genau der Hälfte der Breiten- und Höhenangabe entsprechen.

Weil der Radius der Rundungen genau halb so groß ist wie die Seiten des Quadrates, entsteht der Eindruck eines Kreises. Der Absatz bleibt als Blockelement aber nach wie vor quadratisch. Dank zentriertem Text bleibt der Text innerhalb des Kreises.

Einen Kreis zeichnen mit border-radius



kap08\border-radius2.htm

Noch mehr Möglichkeiten mit der Langform

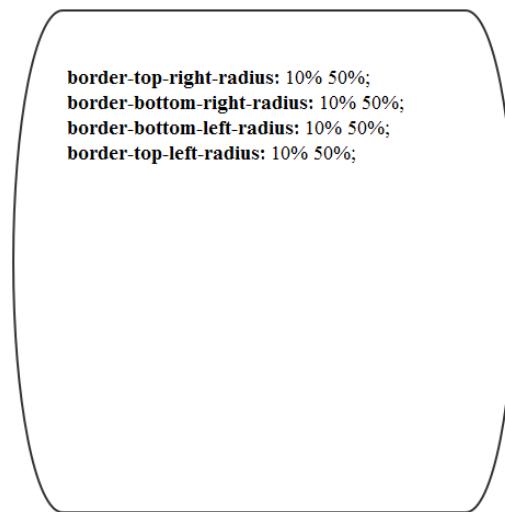
Sie können aber auch für eine oder mehrere Ecken individuelle Werte angeben. Dies geschieht mittels `border-top-right-radius`, `border-bottom-right-radius`, `border-bottom-left-radius` und `border-top-left-radius`. Dann sind pro Ecke sogar zwei Werte erlaubt – der erste für die horizontale, der zweite für die vertikale „Hälfte“ der Rundung.

`border-bottom-left-radius: 12px 24px;`

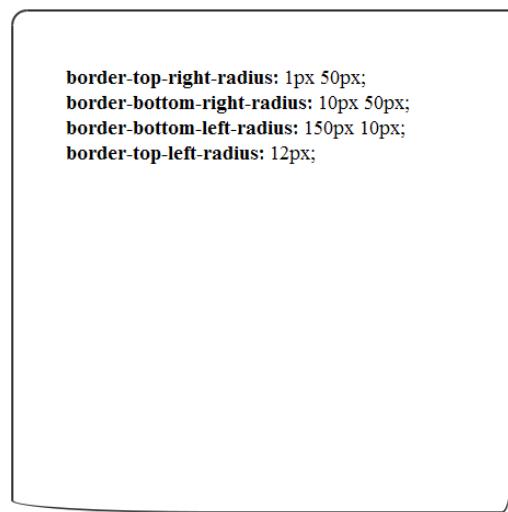
Die folgende Abbildung zeigt zwei Beispiele:

Die Langform von border-radius

Variante 1



Variante 1



kap08\border-radius-long.htm

8.7 Rahmen mit Bildern

Mit `border-image` haben Sie die Möglichkeit, Rahmen mit beliebigen Bildern zu versehen. Darüber hinaus können Sie beeinflussen, wie die Bilder eingesetzt werden. Diese können wie Hintergrundbilder gekachelt oder gestreckt werden.

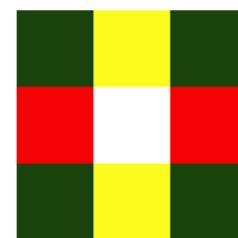
```
border-image: url() Wert1 Wert2 Wert3 Wert4 Schlüsselwort;
```

Werte	<ul style="list-style-type: none"> ✓ Zahlen mit Angabe einer Maßeinheit wie em, px oder % ✓ Schlüsselwort: stretch, repeat, round oder space
Standardwert	Für Schlüsselwort
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Um ein Bild für einen Rahmen verwenden zu können, sollte dieses folgende Voraussetzungen erfüllen:

- ✓ Nutzen Sie möglichst das Format Portable Network Graphics (PNG), um Transparenzen verwenden zu können.
- ✓ Das Bild sollte aus drei mal drei Bereichen bestehen, diese repräsentieren die Ecke oben links, den oberen Rand, die Ecke oben rechts, den linken Rand, den Inhaltbereich, den rechten Rand, die Ecke unten links, den unteren Rand und die Ecke unten rechts.

Eine Grafik, die diese Voraussetzungen erfüllt, ist das Bild `border.png` im Ordner `img`. Dieses Bild enthält keine transparenten Bereiche. Die Kantenlänge des Quadrates beträgt 30 Pixel, jeder der neun Bereiche misst 10×10 Pixel.



`border.png` besteht aus neun Bereichen (stark vergrößert).

Dem Browser müssen Sie mitteilen, wie die Grafik unterteilt werden soll. Die vier Werte für die Schnittlinie müssen angegeben werden. Im vorliegenden Beispiel soll der Browser die Grafik wie folgt aufteilen: 10px vom oberen Rand, 10px vom rechten Rand, 10px vom unteren Rand und 10px vom linken Rand. Die Reihenfolge ist vorgegeben. Das Bild `border.png` wird so genau zwischen den Quadranten geteilt.

Außerdem müssen Sie dem Browser wie bei jedem Rahmen mitteilen, wie dick dieser sein soll.

Notieren Sie dafür folgende Regeln in Ihrem CSS:

```
border-width: 10px;  
border-image: url(img/border.png) 10 10 10 10;
```

Die hier verwendete Grafik dient dazu, die neun Bereiche erkennbar zu machen. Mit Ihren eigenen Bildern können Sie beliebige Rahmen realisieren, die beispielsweise wie ausgerissenes Papier aussehen. Auch ungewöhnliche Schatteneffekte, die über die Möglichkeiten von CSS hinausgehen, oder dreidimensionale Effekte sind denkbar.

Das folgende Beispiel verdeutlicht die Verwendung von Bildern für Rahmen und die Bedeutung der Schlüsselwörter `repeat`, `round` und `stretch`.

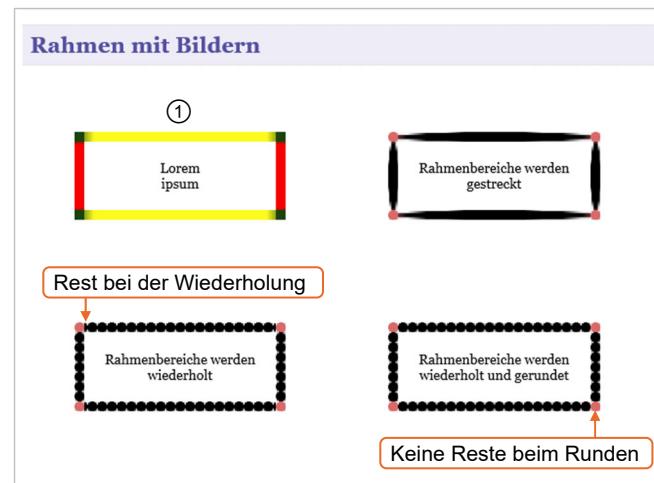
Beispiel: *kap08\border-image.htm*

```
<html>
<head>
    <title>Rahmen mit Bildern</title>
    <style type="text/css">
        ① div {
            float: left;
            margin: 50px;
            width: 150px;
            padding: 18px;
            text-align: center;
            border: 10px solid red;
            border-image: url(img/border.png) 10 10 10 10;
        }
        ② div.fristExample {
            border-image: url(img/borderCircle.png) 10 10 10 10 stretch;
        }
        ③ div.scndExample {
            border-image: url(img/borderCircle.png) 10 10 10 10 repeat;
        }
        ④ div.thrdExample {
            border-image: url(img/borderCircle.png) 10 10 10 10 round;
        }
    </style>
</head>

<body>
    <h1>Rahmen mit Bildern</h1>
    <div>
        Lorem ipsum
    </div>
    <div class="fristExample">
        Rahmenbereiche werden gestreckt
    </div>
    <div class="scndExample">
        Rahmenbereiche werden wiederholt
    </div>
    <div class="thrdExample">
        Rahmenbereiche werden wiederholt und gerundet
    </div>
</body>
</html>
```

- ① Zunächst werden allgemeine Angaben für alle Boxen gemacht, die einen Rahmen bekommen sollen. Als Bild für den Rahmen der ersten Box wird ein Quadrat benutzt, das in 3 mal 3 Bereiche eingeteilt wird.
- ② Für Boxen mit der Klasse `frstExample` legen Sie mittels `stretch` fest, dass Bereiche der Grafik gestreckt werden sollen, um die Ränder zu füllen (im Beispiel ein Kreis).
- ③ In der Klasse `scndExample` weisen Sie den Browser durch `repeat` an, den entsprechenden Bereich so oft zu wiederholen, dass die Box einen kompletten Rahmen erhält, der sich aus Kreisen zusammensetzt. Wenn sich die Höhe oder die Breite des Kastens nicht durch die Höhe des Bereiches teilen lässt, werden die äußeren Kreise abgeschnitten.
- ④ Mit dem Schlüsselwort `round` legen Sie für die Klasse `thrdExample` fest, dass der Browser die Größe der Kreise so anpassen soll, dass diese nicht abgeschnitten werden.

Der erste Kasten ① verdeutlicht, wie die neun Bereiche der Grafik `border.png` verteilt werden. Die weiteren zeigen, wie sich die Schlüsselwörter `stretch`, `repeat` und `round` auswirken.



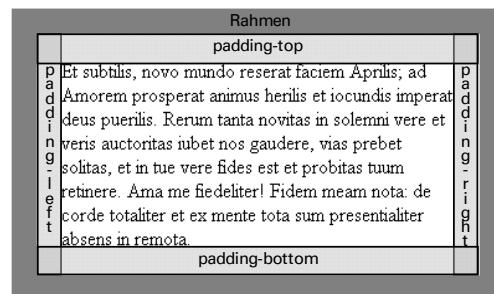
kap08\border-image.htm

! Früher mussten für bestimmte CSS-Features wie etwa auch für `border-image` sogenannte Browser-Präfixe (auch Hersteller-Präfix oder engl.: *vendor prefix*) verwendet werden, etwa `-ms-` für den Internet Explorer oder `-moz-` für den Firefox. Diese Möglichkeit wurde gerne angenommen, um die neuesten Techniken den Nutzern modernster Browser anzubieten. Was im Sinne von progressive enhancement, also der Verbesserung des Nutzererlebnisses für Nutzer mit einer modernen Ausstattung, erst mal gut klingt, hat in der Praxis dazu geführt, dass zahllose Webseiten viele Zeilen unnötigen Code ausliefern. Denn inzwischen sind die meisten der ehemals experimentellen Features standardisiert und von den Browser-Herstellern in ihrer endgültigen Form implementiert. Die mit vielen drei- und vierfachen Angaben für eine einzige Eigenschaft durchsetzten CSS-Dateien werden aber zum größten Teil nicht bereinigt und werden uns wohl begleiten, bis die betroffenen Webseiten eine komplette Neuentwicklung erfahren.

8.8 Innenabstand mit padding

Der Abstand des Blockinhalt zu seinem Rahmen wird Innenabstand genannt. Sie können die CSS-Eigenschaft `padding` benutzen, um die Dimensionen dieses Abstandes zu beeinflussen.

```
padding: Wert;
```



Werte	✓ Zahlenangaben mit entsprechender Maßeinheit ✓ Prozentuale Angaben
Standardwert	0
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Die Angabe eines Wertes bezieht sich – wie auch bei Rahmen – auf alle vier Innenabstände, zwei Werte sind jeweils für oben / unten und rechts / links gedacht. Drei Werte teilen sich auf in oben, rechts/links und unten. Vier Werte geben die Abstände für jeweils eine einzelne Seite an.

Auch bei der Eigenschaft `padding` können Sie eine spezielle Seite des Abstands ansprechen.

```
padding-top: Wert; /* oberer Innenabstand */
padding-right: Wert; /* rechter Innenabstand */
padding-bottom: Wert; /* unterer Innenabstand */
padding-left: Wert; /* linker Innenabstand */
```

Beispiel: [kap08\padding.htm](#)

Der Inhalt verschiedener Absätze wird durch zusätzliche Innenabstände erweitert. Damit der Abstand sichtbar wird, verwenden Sie einen zusätzlichen sichtbaren Rahmen.

```
<html>
<head>
<title>Innenabstände - padding</title>
<style>
①  p { font: 1rem Helvetica, Arial, Geneva, sans-serif;
      text-align: justify;
      border: 1px solid #cc0; }
②  .abstandallg { padding: 10px; }
③  .abstandzwei { padding: 0.2in 20mm; }
```

```

④    .abstanddrei { padding: 12pt 10mm; padding-bottom: 0px; }
      </style>
</head>
<body>
  <h2>Innenabstände mit padding</h2>
⑤  <p>Inhalt ohne Innenabstand;<br>Fortune plango vulnera
     stilantibus...</p>
  <p class="abstandallg">padding: 10px;<br>oben, rechts,
     unten, links 10 Pixel;
  <br>Fortune plango vulnera stilantibus ocellis, quod sua
     michi munera...</p>
  <p class="abstandzwei">padding: 0.2in 20mm;<br>oben/unten:
     0.2Inch; rechts/links: 20mm;<br>Fortune plango vulnera
     stilantibus ocellis,...</p>
  <p class="abstanddrei">padding: 12pt 10mm; padding-bottom:
     0px;<br>oben: 12 Points; rechts/links: 10mm; unten: 0
     Pixel;<br>Fortune...</p>
</body>
</html>

```

- ① Für die Darstellung des Textes legen Sie die Schriftformatierungen fest. Damit der Innenabstand auch auf der rechten Seite deutlich wird, definieren Sie den Text als Blocksatz. Den Abstand machen Sie durch einen 1 Pixel breiten Rahmen deutlich.
- ② Im Klassen-Selektor `abstandallg` legen Sie grundsätzlich für alle vier Seiten einen Innenabstand von 10 Pixeln fest.
- ③ Der Selektor `abstandzwei` erhält zwei Werte, womit oben und unten ein gleicher Abstand von 0,2 Zoll (entspricht ca. 5,1 mm) sowie rechts und links jeweils ein Abstand von 20 mm gewährleistet ist.
- ④ Bei dem dritten Selektor legen Sie ebenfalls über die zwei Werte den oberen/unteren und rechten/linken Abstand fest. Mit `padding-bottom` überschreiben Sie jedoch zusätzlich den unteren Abstand, sodass dieser von 10 mm auf 0 Pixel verkleinert wird.
- ⑤ Es folgen die verschiedenen Absätze, denen die Klassen zugewiesen werden. Der erste Absatz erhält zum besseren Vergleich keine Abstandformatierung und wird deshalb mit seinen Standardabständen angezeigt.

Innenabstände (padding)

Inhalt ohne Innenabstand;
Fortune plango vulnera stilantibus ocellis, quod sua michi munera subtrahit rebellis.
Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata.

`padding: 10px;`
`oben, rechts, unten, links: 10 Pixel;`
Fortune plango vulnera stilantibus ocellis, quod sua michi munera subtrahit rebellis.
Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata.

`padding: 0.2in 20mm;`
`oben/unten: 0.2Inch; rechts/links: 20mm;`
Fortune plango vulnera stilantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata.

`padding: 12pt 10mm; padding-bottom: 0px;`
`oben: 12 Points; rechts/links: 10mm; unten: 0 Pixel;`
Fortune plango vulnera stilantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata.

*Verschiedene Innenabstände in den Absätzen
(kap08\padding.htm)*

8.9 Probleme mit Innenabständen und Rahmen bewältigen

Das CSS-Box-Modell ist äußerst praktisch im Umgang mit Grafiken, weil Sie einer Grafik leicht einen Innenabstand und einen Rahmen mitgeben können: Als Breite für die benötigte Box geben Sie die Breite der Grafik an und erhalten eine Box, die exakt so groß ist wie die Grafik. Innenabstand und Rahmen werden um die Grafik gezeichnet, ohne sie zu beschneiden oder zu verdecken.

Allerdings spielen Grafiken mit festen Breiten-Angaben in Zeiten responsiver Webseiten, die sich automatisch an veränderte Bildschirmgrößen anpassen, praktisch keine Rolle mehr.

Außerdem ist dieses Verhalten hinderlich, wenn Sie einem Element eine relative Breite geben wollen. Ein Kasten mit Text soll beispielsweise halb so breit sein wie der Artikel, in dem er steht.

Notieren Sie dafür zunächst das benötigte HTML:

```
<html>
<head>
    <title>Boxen mit relative Breiten</title>
</head>
<body>
    ①   <article>
        <h1>Boxen mit relativen Breiten</h1>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing
            elit...</p>
        <aside>
            ③   <h1>Weitere Informationen</h1>
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing
                elit...</p>
        </aside>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing
            elit...</p>
    </article>
</body>
</html>
```

- ① Ein Artikel beginnt.
- ② In den Artikel werden Zusatz-Informationen eingebettet.
- ③ Da Sie mittels aside-Element implizit einen hierarchisch eigenständigen Bereich definieren (wie section), können Sie diesen mit einer h1 überschreiben. Browser erkennen die Verschachtelung solcher Bereiche. In HTML5 ist es weiterhin möglich, aber nicht nötig, die Hierarchien mittels Überschriften h1 bis h6 mitzuteilen.

Mit zwei Zeilen CSS sorgen Sie dafür, dass die Zusatz-Informationen in einem vom restlichen Text umflossenen Kasten dargestellt werden, der halb so breit ist wie der Artikel:

```
aside {
    float: left;
    width: 50%; }
```

Mit ausreichend Abstand zum restlichen Text und einer anderen Schrift kann der Leser bereits erkennen, dass dieser Text nicht zum eigentlichen Artikel gehört.

Geben Sie dem Kasten nun einen Rahmen, wird dieser direkt an den enthaltenen Text anstoßen. Um das zu verhindern, wird zusätzlich zum Rahmen noch ein Innenabstand benötigt. Ebenso fügen Sie einen Außenabstand `margin` zum umgebenden Text hinzu (Außenabstände werden im nächsten Abschnitt erläutert).

```
aside {
    float: left;
    width: 50%;
    margin: 10px;
    border: 10px solid #eee;
    padding: 10px; }
```

Da alle Abstände und Rahmen zur Breite addiert werden, erhalten Sie eine Box, die deutlich größer ist als 50 % der Artikelbreite – der Außenabstand verringert den Platz, der dem Text für den eigentlichen Artikel zur Verfügung steht, zusätzlich.

Zwei Möglichkeiten bietet CSS, um dieses Problem zu lösen. Entweder Sie ändern das Box-Modell. Notieren Sie die folgende CSS-Deklaration, um dem Browser mitzuteilen, dass die Box inklusive Innenabständen und Rahmen halb so breit sein soll wie der Artikel:

```
box-sizing: border-box;
```

Boxen mit relativen Breiten

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Rerum adipisci in officiis asperiores accusamus ipsa iure eum, fugiat dolores necessitatibus consequuntur nam iste eos deleniti sed, culpa nihil aliquam ex!

Weitere Informationen

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ex ea blanditiis voluptate veritatis, repudiandae cumque natus. Harum laborum voluptatibus accusamus hic odio? Architecto officiis, officia. Vel autem possimus explicabo, saepe!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Mollitia asperiores, ipsam reiciendis dolor nam illo accusantium vitae. Sunt quod optio tenetur quos nesciunt. Similique quaerat aliquam eaque repudiandae qui soluta.

Die Box „Weitere Informationen“ ist zu breit, für den Haupttext bleibt wenig Raum (kap08\boxModellProblem.htm).

Der Standardwert für `border-box` lautet `content`. Mit dem Schlüsselwort `padding` sorgen Sie dafür, dass sich Größenangaben auf Inhalt und Innenabstand beziehen und nur Rahmen-dimensionen hinzugaddiert werden.

Die zweite Möglichkeit ist etwas komplizierter, dafür haben Sie mehr Möglichkeiten. Anstelle eines einfachen Wertes können Sie Zahlen vom Browser berechnen lassen.

Notieren Sie folgende Zeile in Ihrem CSS, um die Breite von Innenabständen (2×10 Pixel) und Rahmen (2×10 Pixel) von der Breite der Box abzuziehen:

```
width: calc(50% - 40px);
```

Notieren Sie jeweils ein Leerzeichen vor und nach dem Minus, damit der Browser die Werte mit Einheiten (50% und 40px) von den Rechenzeichen unterscheiden kann.

Da Sie so beliebige Berechnungen anstellen können, haben Sie mit dieser Methode mehr Möglichkeiten, die Ausmaße der Box zu beeinflussen. Auch Rechnungen mit geklammerten Operationen sind möglich:

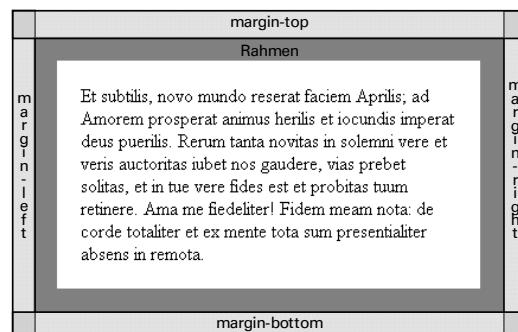
```
width: calc((2px + 2px) * 3)
```

Allerdings ist diese Methode aufwendiger und fehleranfälliger. Verwenden Sie daher `box-sizing: border-box` wo möglich und eine berechnete Breitenangabe wo nötig.

8.10 Außenabstand mit `margin`

In dem vorherigen Beispiel wird ersichtlich, dass der Rahmen immer im gleichen Abstand zum Rand des Browserfensters gesetzt wird. Ebenso sind die Abstände zwischen den einzelnen Absätzen immer gleich.

Das sind Außenabstände, die Sie mit der CSS-Eigenschaft `margin` ändern können. Damit variieren Sie den Außenabstand von Blockelementen. Weil eine freie Fläche um das Element entsteht, wird der Abstand auch Weißraum genannt.



```
margin: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ Zahlenangaben mit entsprechender Maßeinheit; standardmäßig wird die Maßeinheit Pixel (<code>px</code>) verwendet. ✓ Auch negative Werte sind möglich. ✓ Prozentuale Angaben zur Höhe und Breite des Browserfensters ✓ Schlüsselwort <code>auto</code>
Standardwert	Rechts/links: 0 px; oben/unten: browserabhängig, ca. 10 px bis 20 px
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Bei der Angabe des Schlüsselworts `auto` berechnet der Browser den Außenabstand. Alle aktuellen Browser setzen das Element, dem Sie `auto` zuweisen, mittig in das Elternelement. So können Sie Blockelemente horizontal zentrieren. Wenn Sie für den Außenabstand einen negativen Wert angeben, wird das Blockelement entweder über das vorherige Element oder über den Browserrand hinausgeschoben.

Bei der Angabe der Werte gilt das Gleiche wie bei den Rahmen und Innenabständen: Ein Wert steht für alle vier Seiten, alle anderen Werte werden nach dem bekannten Muster aufgeteilt. Außerdem können Sie Angaben darüber machen, auf welcher Seite der Abstand hinzugefügt werden soll.

```
margin-top:   Wert; /* oberer Außenabstand */
margin-right: Wert; /* rechter Außenabstand */
margin-bottom: Wert; /* unterer Außenabstand */
margin-left:  Wert; /* linker Außenabstand */
```

Beispiel: *kap08\margin.htm*

Verschiedenen Absätzen mit Rahmen und Innenabständen werden in diesem Beispiel verschiedene Außenabstände zugewiesen. Für die Wirkung der Abstände werden neben den Rahmen zusätzlich horizontale Trennlinien *<hr>* eingefügt.

```
<html>
<head>
<title>Außenabstände - margin</title>
<style>
①    p { font: 1rem Helvetica, Arial, Geneva , sans-serif;
      text-align: justify;
      border: 1px solid #cc0; padding: 5px; }
②    .abstandallg { margin: 10px; }
③    .abstandzwei { margin: 0.2in 20mm; }
④    .abstanddrei { margin: 12pt 10mm; margin-bottom: 0px; }
</style>
</head>
<body>
<h2>Außenabstände mit margin</h2><hr>
⑤    <p>Inhalt ohne Außenabstand;<br>Fortune plango
vulnera...</p><hr>
    <p class="abstandallg">margin: 10px;<br>oben, rechts, unten,
links 10 Pixel;
<br>Fortune plango vulnera stilantibus ocellis, quod sua
michi...</p><hr>
    <p class="abstandzwei">margin: 0.2in 20mm;<br>oben/unten:
0.2Inch; rechts/links: 20mm;<br>Fortune plango vulnera
stilantibus...</p><hr>
    <p class="abstanddrei">margin: 12pt 10mm; margin-bottom:
0px;<br>oben: 12 Points; rechts/links: 10mm; unten: 0 Pixel;
<br>...</p><hr>
</body>
</html>
```

- ① Für den Inhalt der Absätze formatieren Sie die Schriftart und -größe und setzen den Inhalt als Blocksatz. Um die Absätze setzen Sie einen 1 Pixel breiten, durchgehenden Rahmen. Der Abstand des Inhalts zum Rahmen beträgt auf allen vier Seiten 5 Pixel. Dies ist die Grundformatierung der Absätze für die gesamte Webseite.
- ② Die Einstellungen der Außenabstände nehmen Sie in separaten Klassen-Selektoren vor. Hier wird ein Abstand angegeben, sodass das Element auf allen vier Seiten 10 Pixel eingezogen wird.
- ③ In diesem Selektor legen Sie fest, dass der Außenabstand oben und unten jeweils 0,2 Zoll sowie rechts und links jeweils 20 mm betragen soll.
- ④ Der Selektor `abstanddrei` beinhaltet ebenfalls zwei Werteangaben. Der untere Abstand wird jedoch durch die Eigenschaft `margin-bottom` noch einmal separiert und auf 0 Pixel gesetzt.
- ⑤ Es folgen die einzelnen Absätze mit den zugewiesenen Klassen-Selektoren. Zum Vergleich wird der erste Absatz mit den Standardabständen dargestellt.

Collapsing Margins

Wenn Sie Absätzen einen oberen und unteren Abstand von 10 Pixeln zuweisen, werden diese 10 Pixel weit voneinander dargestellt, nicht 10 plus 10, also 20 Pixel. Steht über einem solchen Absatz eine Überschrift mit einem unteren Abstand von 15 Pixeln, wird der Absatz vom Browser 15 Pixel unter die Überschrift gesetzt.

Aufeinanderfolgende Außenabstände werden nicht addiert. Lediglich der größere Abstand bestimmt, wie weit die Elemente tatsächlich auseinanderliegen. Dieses Verhalten ist in den Standards so festgeschrieben und soll den Umgang mit Abständen erleichtern. In der Praxis können Sie so Abstände herstellen, ohne die Abstände aller umgebenden Elemente berücksichtigen zu müssen.

Möchten Sie das Überlagern der Abstände vermeiden, verwenden Sie an der entsprechenden Stelle ein Element mit einem Rahmen (`border`) oder Innenabstand (`padding`). Diese Eigenschaften führen dazu, dass Außenabstände addiert werden.

Außenabstände mit margin

```
Inhalt ohne Außenabstand:  
Fortune plango vulnera stiantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronde capillata, sed plerumque sequitur occasio calvata.
```

```
margin: 10px;  
oben, rechts, unten, links 10 Pixel;  
Fortune plango vulnera stiantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronde capillata, sed plerumque sequitur occasio calvata.
```

```
margin: 0.2in 20mm;  
oben/unten: 0.2inch; rechts/links: 20mm;  
Fortune plango vulnera stiantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronde capillata, sed plerumque sequitur occasio calvata.
```

```
margin: 12pt 10mm; margin-bottom: 0px;  
oben: 12 Points; rechts/links: 10mm; unten: 0 Pixel;  
Fortune plango vulnera stiantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronde capillata, sed plerumque sequitur occasio calvata.
```

*Verschiedene Außenabstände um die Absätze
(kap08\margin.htm)*

8.11 Übungen

Übung 1: Rahmen und Abstände

Level		Zeit	ca. 10 min
Ergebnisdatei	<i>kap08\uebung1-2.doc</i>		

1. Erklären Sie den Unterschied zwischen den Eigenschaften `border`, `padding` und `margin`.
2. Wie werden die Werte verteilt, wenn Sie für die Eigenschaften 1, 2 oder 4 Werte angeben?

Übung 2: Das Box-Modell anwenden

Level		Zeit	ca. 45 min
Ergebnisdatei	<i>kap08\uebung3.htm</i>		

1. Erstellen Sie die nachfolgende Webseite. Setzen Sie die folgenden Anforderungen um:
 - ✓ Absatz nach der Überschrift: fett formatierte Schrift; am unteren Rand eine 1 Pixel breite hellgraue Abgrenzung in Form einer gestrichelten Linie
 - ✓ Andere Absätze haben generell einen Innenabstand von 10 Pixeln.
 - ✓ Hervorgehobener Merksatz: Die erste Zeile wird fett und in der Farbe #800 dargestellt.
 - ✓ Hervorgehobener Merksatz: Der Außenabstand beträgt oben und unten 0 Pixel, rechts und links 50 Pixel; links besteht ein 10-Pixel-Innenabstand zum 20 Pixel breiten Rahmen.

Anwenden des Boxmodells

Fortune plango vulnera stilantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata. In Fortune solo sederam elatus, prosperitatis vario flore coconatus.

Quicquid enim florui felix et beatus, nunc a summo corruui gloria privatus. Fortune rota volvitur descendendo minoratus, alter in altum tollitur, nimis exaltatus rex sedet in vertice; caveat ruinam! Nam sub axe legimus hecubam redinam.

Merke:
Zephyrus nectareo spirans in odore, certiam pro bravia curramus in amore. Cytharizat cantico dulcis Philomena, flore rident vario prata iam serena, salit cetus avium silve per amena, iam gaudia millena.

Omnia Sol temperat purus et subtilis, novo mundo reserat faciem Aprilis; ad Amorem prosperat animus herilis et iocundis imperat deus puerilis. Rerum tanta novitas in solemni vere et veris auctoritas iubet nos gaudere, vias prebet solitas, et in tue vere fides est et probitas tuum retinere. Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.

Das erwartete Ergebnis der Übung (*kap08\uebung3.htm*)

Übung 3: Rahmenobjekte erstellen

Level		Zeit	ca. 30 min
Ergebnisdatei	<i>kap08\uebung4.htm</i>		

1. Erstellen Sie die folgenden Objekte mithilfe der Rahmeneinstellungen und der bisher erlernten Formatierungen.



Objekte, die aus Rahmendefinitionen erstellt wurden (kap08\uebung4.htm)

9

Tabellen

9.1 Die Eigenschaft `display` und Einführung in CSS-Tabellen

In HTML zählen Tabellen zu den Strukturierungselementen, um eine übersichtlichere Darstellung von tabellarischen Daten zu ermöglichen.

Mit folgenden Eigenschaften können Sie ausschließlich Tabellen gestalten.

- ✓ `caption-side`
- ✓ `border-collapse`
- ✓ `empty-cells`
- ✓ `table-layout`
- ✓ `border-spacing`

9.2 Darstellung von Elementen ändern

Bevor HTML-Tabellen behandelt werden, erfahren Sie hier, wie Sie Elemente dazu bringen, so auszusehen und sich so zu verhalten wie Tabellen. Anhand des Layoutbeispiels werden Sie sehen, wofür das nützlich sein kann.



Wenn Sie tabellarische Daten bereitstellen, sollten Sie diese auch korrekt auszeichnen, indem Sie die entsprechenden HTML-Elemente verwenden. CSS-Tabellen sind nur für Layoutzwecke gedacht.

Bisher haben Sie gelernt, dass der Browser Überschriften anders darstellt als Fließtext, Listen anders als Tabellen. Über die CSS-Eigenschaft `display` können Sie Elementen eine andere Darstellung zuweisen.

<code>display: Wert;</code>

Werte	<ul style="list-style-type: none"> ✓ none = keine Anzeige ✓ block = Darstellung als Block (wie Absätze) ✓ inline = fließt im Text mit (wie span) ✓ inline-block = Blöcke, die im Text mitlaufen (wie Bilder) ✓ list-item = Blöcke mit Aufzählungszeichen (wie Listeneinträge) ✓ run-in = ein Block- bzw. Inline-Element; abhängig vom Inhalt ✓ inline-table = Darstellung als Tabelle ohne Absatz ✓ table = Darstellung als Tabelle mit Absatz ✓ table-caption = Darstellung als Tabellenüberschrift ✓ table-cell = Darstellung als Tabellenzelle ✓ table-column = beschreibt eine Tabellenspalte ✓ table-column-group = Darstellung als Tabellenspalten ✓ table-footer-group = Darstellung als Fußzeile einer Tabelle ✓ table-header-group = Darstellung als Kopfzeile einer Tabelle ✓ table-row = Darstellung als Tabellenzeile ✓ table-row-group = Darstellung als Tabellenzeilen
Standardwert	inline
Vererbbar	Nein
Anwendbar auf	Alle Blockelemente



Bislang wird der Wert `run-in` von den großen Browersern nicht korrekt in die Darstellung umgesetzt.

Die Werte, die dazu führen, dass ein Element als Tabellenelement dargestellt wird, können Sie mit den entsprechenden HTML-Tags gleichsetzen.

```
table { display: table }
tr { display: table-row }
thead { display: table-header-group }
tbody { display: table-row-group }
tfoot { display: table-footer-group }
col { display: table-column }
colgroup { display: table-column-group }
td, th { display: table-cell }
caption { display: table-caption }
```

Elemente, denen die Eigenschaft `display` mit den Werten `table-column` oder `table-column-group` zugeordnet wird, sind nicht sichtbar. Sie können daher nur über das Attribut als Stilangabe für die entsprechenden Spalten genutzt werden.

Beispiel: kap09\display-table.htm

Im Folgenden formatieren Sie div-Elemente so, dass der Browser sie als Tabelle darstellt. Nutzen Sie dafür die Eigenschaft `display` mit den entsprechenden Werten.

```
<html lang="de">
<head>
<title>Tabelle mit display</title>
<style>
①    .tabelle { display: table;
            border: 1px solid;
            width: 500px;
        }
②    .zeile { display: table-row; }
③    .zelle1 { display: table-cell;
            border:1px solid;
            padding:5px;
            width: 100px;
        }
        .zelle2 { display: table-cell;
            border: 1px solid;
            padding: 5px;
        }
        .caption { display: table-caption;
            margin: 10px; "
        }
④    span { display: block;
            background: #f00;
            font-weight: bold;
        }
</style>
</head>
<body>
⑤ <div class="tabelle">
⑥   <div class="caption">Tabellenüberschrift</div>
⑦   <div class="zeile">
⑧     <div class="zelle1">Zelleninhalt</div>
⑨     <div class="zelle2">Der <span>Inhalt</span> einer
          Zelle</div>
    </div>
⑩   <div class="zeile">
     <div class="zelle1">Zellen<span>inhalt</span></div>
     <div class="zelle2">Zelleninhalt</div>
   </div>
</div>
</body>
</html>
```

- ① Das Element, das die Klasse `tabelle` zugewiesen bekommt, soll als Tabelle dargestellt werden und alle anderen Elemente beinhalten. Damit die Tabelle sichtbar ist, erhält sie einen Rahmen.
- ② Elemente mit der Klasse `zeile` fungieren als Tabellenzeile und sollen die Elemente beherbergen, die sich wie Tabellenzellen verhalten sollen.
- ③ Die simulierten Zellen werden über den Wert `table-cell` festgelegt. Auch hier werden zur Verdeutlichung der Abmessungen Rahmen eingesetzt. Damit der Zelleninhalt nicht direkt am Zellrahmen beginnt, fügen Sie einen Innenabstand von 5 Pixeln hinzu. Zellen mit der Klasse `zeile1` sind dabei auf eine maximale Breite von 100 Pixeln begrenzt.
- ④ Die Elemente `span` sollen wie Blockelemente dargestellt werden. Sie verhalten sich dann z. B. wie Absätze. Zur Verdeutlichung werden die entsprechenden Inhalte rot hinterlegt und fett formatiert.
- ⑤ Dem ersten `div` weisen Sie die Klasse `tabelle` zu und leiten somit die darzustellende Tabelle ein.
- ⑥ Dem nächsten Bereich weisen Sie die Klasse `caption` zu. Er kommt als Beschriftung einer Tabelle mit einem Abstand von 10 Pixeln zum Einsatz.
- ⑦ Dieses `div` wird wie ein `tr` behandelt, indem ihm der Selektor `zeile` zugewiesen wird.
- ⑧ Die erste von zwei Tabellenzellen.
- ⑨ Das zweite, als Tabellenzelle formatierte `div`-Element besitzt als Inhalt ein Element `span`.
- ⑩ Mit der erneuten Zuweisung der Formatierung `zeile` wird eine zweite Tabellenzeile mit zwei weiteren Tabellenzellen eingefügt.

Tabellenüberschrift	
Zelleninhalt	Der Inhalt einer Zelle
Zellen inhalt	Zelleninhalt

Eine CSS-Tabelle

Bedeutung in der Praxis

Eine echte Webseite werden Sie niemals nur aus `div`-Elementen zusammensetzen, um dann über CSS dafür zu sorgen, dass diese aussieht, als enthalte sie Tabellen, Listen und so weiter, denn das wäre weder semantisch korrekt, noch ist es effizient. In der Praxis geht man häufig den umgekehrten Weg. Man zeichnet seine Dokumente korrekt aus und legt über CSS gegebenenfalls eine andere Darstellung fest.

Mithilfe der Änderung von Darstellungseigenschaften können Sie zwei Dinge erreichen:

- ✓ Erstens zeichnen Sie die Webseite semantisch korrekt aus.
- ✓ Zweitens können Sie die Darstellung der Elemente unabhängig von ihrem Standard-Verhalten so beeinflussen, dass Sie sich wie die von Ihnen zur optischen Gestaltung gewünschten Elemente verhalten.

Beispiel: `kap09\list-as-table.htm`

Üblicherweise wollen Sie nicht, dass eine Aufzählung von Links (in aller Regel also ein Navigationsmenü) wie eine Liste aussieht.

Mit dem folgenden CSS-Beispiel erreichen Sie, dass eine Liste wie ein horizontales Menü dargestellt wird, das auch noch die gesamte Breite des Elternelementes einnimmt. Um die Berechnung der benötigten Breite für die einzelnen Menüeinträge kümmert sich der Browser.

```
<html lang="de">
<head>
    <title>Liste als Tabelle darstellen</title>
    <style>
        .tabelle { display: table; border: 1px solid; width: 100%; } ①
        .zeile   { display: table-row; }
        .zelle   { display: table-cell;
                    border: 1px solid;
                    padding: 5px;
                }
    </style>
</head>
<body>
    <div class="tabelle">
        <ul class="zeile">
            <li class="zelle"><a href="#">Ein Link</a></li>
            <li class="zelle"><a href="#">Ein Link</a></li>
            <li class="zelle"><a href="#">Ein Link</a></li>
        </ul>
    </div>
</body>
</html>
```

Die Tabelle sieht fast so aus wie die zuletzt erstellte, allerdings besteht sie aus nur einer Zeile und sie erstreckt sich über die gesamte Breite des Elternelementes (① 100%). Um die Aufteilung der Zellen kümmert sich der Browser – diese werden abhängig von ihrem Inhalt breiter oder schmäler dargestellt, so wie Sie es von HTML-Tabellen her kennen.

Liste als Tabelle darstellen

Ein Link	Ein Link	Ein Link
--------------------------	--------------------------	--------------------------

9.3 Tabellen beschriften

caption-side: Wert;

Werte	<ul style="list-style-type: none"> ✓ top = oben ✓ bottom = unten ✓ left = links ✓ right = rechts
Standardwert	top
Vererbbar	Ja
Anwendbar auf	table oder Elemente, die mit display:inline-table formatiert sind

9.4 Rahmen darstellen

Die Eigenschaft `border-collapse` erlaubt Ihnen, das Verhalten von Rahmen in Tabellen festzulegen.

<code>border-collapse:</code> Wert;
--

Werte	<ul style="list-style-type: none"> ✓ <code>collapse</code> = Zellen werden in einem Rahmen zusammengefasst ✓ <code>separate</code> = Zellen werden einzeln dargestellt
Standardwert	<code>separate</code>
Vererbbar	Ja
Anwendbar auf	table oder Elemente, die mit <code>display:inline-table</code> formatiert sind

Der Unterschied zwischen den beiden Werten wird in den Abbildungen ersichtlich. Mit dem Wert `separate` werden zwischen den einzelnen Zellen Zwischenräume eingefügt, bei `collapse` liegen die Zellen aneinander.

Ein Text, der nicht auf eine Zeile passt.	Ein_overlanges_Wort,_das_über_den_Rand_hinausläuft.
Ein kurzer Satz.	

border-collapse: separate

Ein Text, der nicht auf eine Zeile passt.	Ein_overlanges_Wort,_das_über_den_Rand_hinausläuft.
Ein kurzer Satz.	

border-collapse: collapse

kap09/border-collapse.htm

9.5 Rahmenabstand

Verwenden Sie `border-spacing`, um den Abstand zwischen Zellen festzulegen. Diese Eigenschaft kann jedoch nur aktiviert werden, wenn die Eigenschaft `border-collapse` den Wert `separate` besitzt.

<code>border-spacing:</code> Wert;

Werte	Zahlenangaben mit entsprechender Maßeinheit
Standardwert	0
Vererbbar	Ja
Anwendbar auf	table oder Elemente, die mit <code>display:inline-table</code> formatiert sind

9.6 Leere Tabellenzellen

Leere Tabellenzellen, die in der Form `<td></td>` angelegt sind, werden von den Browsern unterschiedlich dargestellt. Besonders bei eingeschalteter Umrandung fällt die unterschiedliche Darstellung auf. Mit der CSS-Eigenschaft `empty-cells` können Sie zwischen zwei Darstellungen wählen:

empty-cells: Wert;

Werte	<ul style="list-style-type: none"> ✓ <code>show</code> = die Umrandungen der leeren Zellen werden angezeigt ✓ <code>hide</code> = die Zellumrandungen der leeren Zellen werden nicht dargestellt
Standardwert	<code>show</code>
Vererbbar	Ja
Anwendbar auf	table oder Elemente, die mit <code>display:inline-table</code> formatiert sind

9.7 Tabellenlayout

Mit dieser Eigenschaft beeinflussen Sie die Breite der Tabellenzellen, sodass die Zellen bei einem zu breiten Inhalt automatisch vergrößert werden. Sie können auch festlegen, dass die Tabellenzellen ihre festgelegte Breite immer beibehalten müssen. Sollte ein Inhalt größer sein als die Tabellenzelle und nicht umbrochen werden können, wird er in den aktuellen Browsern über den Zellenrand hinaus dargestellt.

table-layout: Wert;

Werte	<ul style="list-style-type: none"> ✓ <code>auto</code> = wenn der Inhalt nicht in die Zelle passt, wird sie vergrößert ✓ <code>fixed</code> = wenn der Inhalt nicht in die Zelle passt, läuft er über den Rand hinaus
Standardwert	<code>auto</code>
Vererbbar	Nein
Anwendbar auf	table oder Elemente, die mit <code>display:inline-table</code> formatiert sind

Bei dem Wert `auto` muss der Browser die Breite einer Spalte immer wieder anhand der Zellinhalte überprüfen und notfalls die Zellen- und Tabellenbreite anpassen. Dadurch kann die komplette Tabelle erst angezeigt werden, wenn sie komplett geladen wurde. Der Wert `fixed` hingegen legt fest, dass die Breitenangaben der ersten Zeile für alle weiteren Zeilen gelten.

Beispiel: *kap09\table-layout.htm*

Erstellen Sie zwei Tabellen. Die erste soll keine zusätzliche CSS-Eigenschaft besitzen. Die zweite Tabelle formatieren Sie wie nachfolgend mit der Eigenschaft `table-layout` und dem Wert `fixed`.

```
<table style="table-layout:fixed; width:505px; border: 1px solid #000">
    <caption><b>CSS: table-layout: fixed</b></caption>
    <tr>
        <td width="200">
            Ein Text, der nicht auf eine Zeile passt.
        </td>
        <td width="300">
            Ein Text, der umgebrochen wird,
            weil er länger als das längste Wort ist.<br>
            Ein_überlanges_Wort,_das_nicht_in_die_Zelle_passen_wird.
        </td>
    </tr>
    <tr>
        <td>&nbsp;</td>
        <td>Ein kurzer Satz.</td>
    </tr>
</table>
```

In dem Beispiel sehen Sie zwei Tabellen, deren linke Zellen 200 Pixel breit sind und deren rechte Zellen eine Breite von 300 Pixeln aufweisen. Durch das überlange Wort kann der Browser in der ersten Tabelle keinen automatischen Zeilenumbruch generieren und muss die Zellenbreite vergrößern. Das hat zur Folge, dass die linken Zellen verkleinert werden. In der zweiten Tabelle wird die Breite des Textes ignoriert, sodass dieser bei einem überlangen Inhalt über den rechten Zellenrand hinausläuft. Die Breiten der Zellen werden in der definierten Größe beibehalten.

The screenshot shows a browser window with two tables. The title bar says "Tabellenlayout".

Top Table (CSS: table-layout: auto):

Ein Text, der nicht auf eine Zeile passt.	Ein Text, der umgebrochen werden wird, weil er länger als das längste Wort ist. Ein_überlanges_Wort,_das_nicht_in_die_Zelle_passen_wird.*
	Ein kurzer Satz.

Bottom Table (CSS: table-layout: fixed):

Ein Text, der nicht auf eine Zeile passt.	Ein Text, der umgebrochen werden wird, weil er länger als das längste Wort ist. Ein_überlanges_Wort,_das_nicht_in_die_Zelle_passen_wird.
	Ein kurzer Satz.

Unterschiedliche Tabellenformatierungen

9.8 Übungen

Übung 1: Mit Blockelementen arbeiten

Level		Zeit	ca. 5 min
Ergebnisdatei	<i>kap09\uebung1.doc, uebung1.htm</i>		

1. Mit welcher Eigenschaft können Sie Elementen ein anderes Verhalten zuweisen (um beispielsweise zu erreichen, dass sich ein Inline-Element wie ein Blockelement verhält)?

Übung 2: Tabellen mit CSS erstellen

Level		Zeit	ca. 20 min
Ergebnisdatei	<i>kap09\uebung2.htm</i>		

1. Wie lautet die Stylesheet-Definition, um die nachfolgend verschachtelten div-Bereiche in eine Tabellenform zu bringen?

```
<div>
  <div>
    <div>Zelle 1</div>
    <div>Zelle 2</div>
    <div>Zelle 3</div>
  </div>
  <div>
    <div>Zelle 4</div>
    <div>Zelle 5</div>
    <div>Zelle 6</div>
  </div>
</div>
```



Tabelle		
Zelle 1	Zelle 2	Zelle 3
Zelle 4	Zelle 5	Zelle 6

Übung 3: Tabellen mit CSS formatieren

Level		Zeit	ca. 45 min
Ergebnisdateien	<i>kap09\uebung3.htm, kap09\uebung4.htm</i>		

1. Erstellen Sie die nachfolgende Tabelle über CSS.

Die Vorgabe	
Erste Zeile mit hellblauem Hintergrund	
Zweite Zeile mit weißem Hintergrund	
HTML-Tabelle	

Verwenden Sie dazu das folgende Grundgerüst.

```
<table cellspacing="0">
  <caption>HTML-Tabelle</caption>
  <tr>
    <td>Erste Zeile mit hellblauem Hintergrund</td>
  </tr>
  <tr>
    <td>Zweite Zeile mit weißem Hintergrund</td>
  </tr>
</table>
```

Alle anderen Einstellungen, wie die Tabellenbreite von 75 %, die hellblaue Hintergrundfarbe #EEEBCF6 für die erste Zeile, die Zellinnenabstände von 10 Pixeln und den Tabellenrahmen, realisieren Sie komplett über die CSS-Eigenschaften.

2. Versuchen Sie, die HTML-Tabelle komplett über die in diesem Kapitel gelernten CSS-Formatierungen darzustellen. Verwenden Sie dazu nur die HTML-Tags `<div>` und `</div>`.

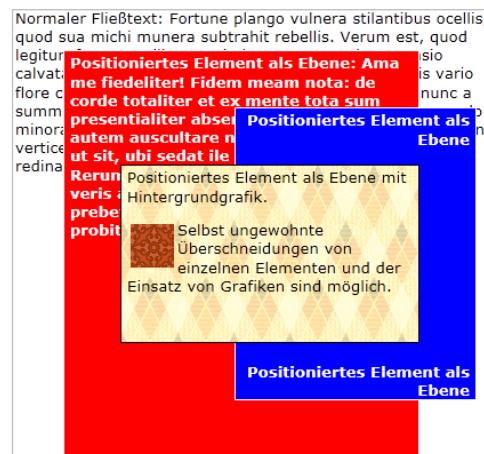
10

Positionieren mit CSS

In der nebenstehenden Abbildung können Sie vier Ebenen erkennen, die übereinanderliegen und sich dadurch teilweise überdecken.

Bisher erschienen alle Elemente, die Sie verwendet haben, im Browser in genau derselben Reihenfolge, in der Sie sie im Quelltext notiert haben.

Im Folgenden erfahren Sie, wie Sie Elemente unabhängig von ihrer Position im Quelltext an einer beliebigen Stelle des Browserfensters anzeigen lassen.



Vier übereinanderliegende, frei positionierte Elemente (kap10\screen.htm)

10.1 Elemente positionieren

Um ein Element zu positionieren, geben Sie seine genaue Position und seine Ausdehnung an.

Art der Positionierung

Sie haben über die CSS-Eigenschaft `position` vier verschiedene Möglichkeiten, ein Element zu positionieren.

```
position: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ <code>static</code> = Element positioniert sich im normalen Elementfluss (Standard) ✓ <code>relative</code> = Abweichung relativ zu der Position, die das Element im normalen Elementfluss eingenommen hätte ✓ <code>absolute</code> = Positionierung des Elements mit Bezug auf das Element <code>body</code> oder den letzten Vorfahren mit einer relativen Positionierung ✓ <code>fixed</code> = wie <code>absolute</code>, aber beim Scrollen der Webseite bleibt das Element fixiert ✓ <code>sticky</code> = wie <code>relative</code>, sobald ein festgelegter Schwellenwert überschritten wird, wie <code>fixed</code>
Standardwert	Static
Vererbbar	Nein
Anwendbar auf	Alle Elemente

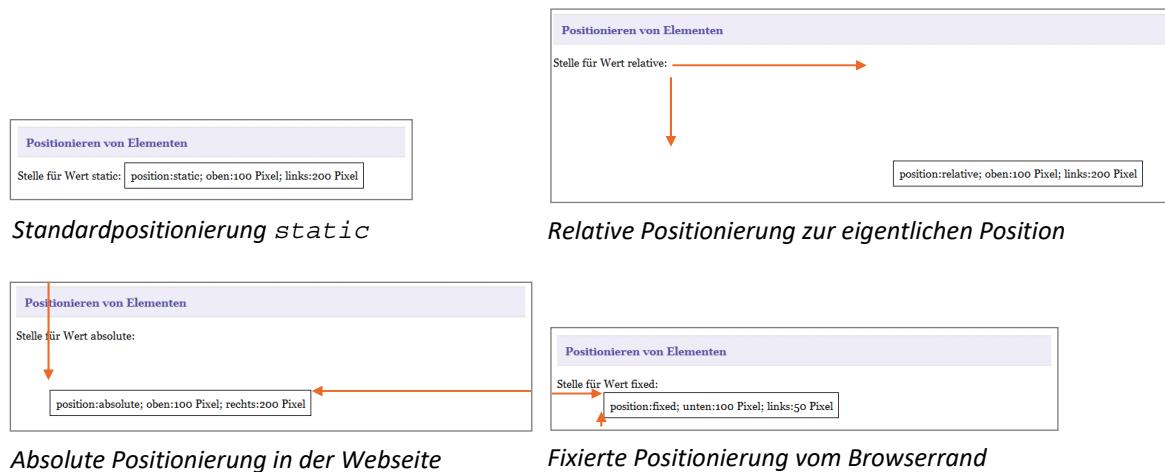
Startposition angeben

Die Angabe, wo Sie ein spezielles Element auf der Webseite platzieren, erfolgt über die vier Startpositionen und die dazugehörigen Zahlenwerte.

```
left: Wert;      /* Position von links */
top: Wert;       /* Position von oben */
right: Wert;      /* Position von rechts */
bottom: Wert;     /* Position von unten */
```

Werte	<ul style="list-style-type: none"> ✓ Zahlenangaben mit entsprechender Maßeinheit ✓ Prozentuale Angaben zur Höhe und Breite des Elternelements ✓ <code>auto</code> = automatische Positionierung
Standardwert	auto
Vererbbar	Nein
Anwendbar auf	Alle Elemente, denen mittels <code>position</code> ein Wert ungleich <code>static</code> zugewiesen wurde

Mit `left` und `top` geben Sie den Abstand von links und oben an. Bei Angabe von `right` beziehen sich die Werte auf die rechte Seite des Elternelements bzw. des Viewports. Haben Sie ein Element mit `position: static` oder keiner Positionsangabe, sind die Angaben der vier Eigenschaften unwirksam.



Beispiel: kap10\position.htm

Die gezeigten Beispiele mit den verschiedenen Positionierungsarten werden allesamt im folgenden HTML-Dokument angewendet. Dazu erstellen Sie vier verschiedene Absätze. In jeden Absatz fügen Sie ein Element span ein und weisen ihm die entsprechende Positionierung zu.

```

<html lang="de">
<head>
  <title>Positionieren von Elementen</title>
  ① <link rel="stylesheet" href="standard.css">
  ② <style>
    span { border:1px solid; padding:5px; }
  </style>
</head>
<body>
  <h1>Positionieren von Elementen</h1>
  ③ <p>Stelle für Wert static: <span style="position: static; top:100px; left:200px;"> position: static; oben:100 Pixel; links:200 Pixel </span></p>

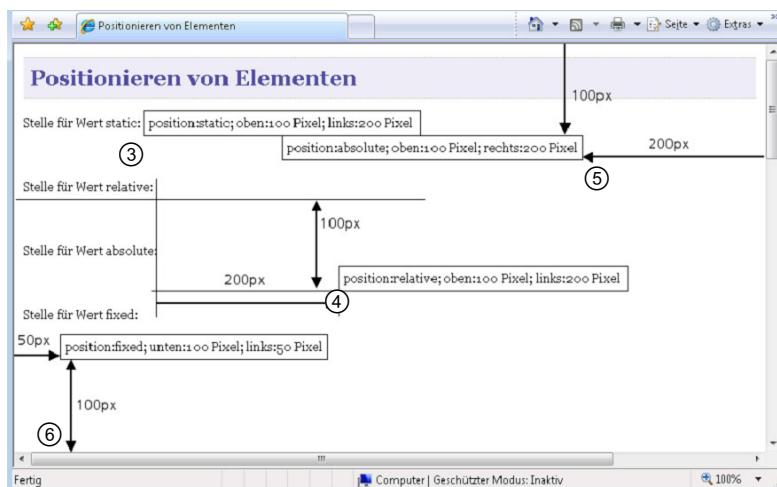
  ④ <p>Stelle für Wert relative: <span style="position: relative; top:100px; left:200px;"> position: relative; oben:100 Pixel; links:200 Pixel </span></p>

  ⑤ <p>Stelle für Wert absolute: <span style="position: absolute; top:100px; right:200px;"> position: absolute; oben:100 Pixel; rechts:200 Pixel </span></p>

  ⑥ <p>Stelle für Wert fixed: <span style="position: fixed; bottom:100px; left:50px;"> position: fixed; unten:100 Pixel; links:50 Pixel </span></p>
</body>
</html>
```

- ① Binden Sie eine CSS-Datei mit Formatierungen für die Elemente body und h1 ein.
- ② Damit Sie die Abmessungen der span-Elemente sehen können, erhalten diese jeweils einen Rahmen.
- ③ Im ersten und allen folgenden Absätzen definieren Sie mit ein Inline-Element. Dieses soll statisch positioniert werden (`position: static`). Die Position sei von oben 100 Pixel sowie 200 Pixel von links.
- ④ Einen relativen Abstand zur eigentlichen Position des Textes erreichen Sie über die Angabe von `position: relative`. Die Koordinaten `top` und `left` beziehen sich auf die Position, die das Element einnimmt, wenn man keine Angaben macht. Fügen Sie vor diesem Absatz weitere Absätze ein, verschiebt sich auch das Element entsprechend auf der Webseite, aber immer in dem festgelegten Abstand zur „normalen“ Position.
- ⑤ Völlig unabhängig von der bisherigen Position können Sie ein Element mit `position: absolute` ausrichten. Die Attributwerte von `top` und `right` sind dabei die Abstände der linken oberen Ecke von body oder des letzten Vorfahrenelementes mit `position: relative`. In dem Fall beginnt das Element 100 Pixel von oben. Der rechte Rand des Elements hat einen Abstand von 200 Pixeln zum rechten Rand der Webseite.
- ⑥ Mit `position: fixed` beziehen sich die Werte der Startpositionen nicht mehr auf den Webseitenrand, sondern auf den Rand des Browsers. Auch beim Scrollen einer Webseite oder beim Ändern der Fenstergröße bleibt das Element in demselben Abstand zum Browserrand. Nun ist der Bezugspunkt immer das Element `html` – was dem sichtbaren Bereich der Webseite (viewport) entspricht.

Das Ergebnis im Browser sieht folgendermaßen aus:



Visualisierung der verschiedenen Positionsarten (kap10\position.htm)

Das Inline-Element ③ wird mit einem Rahmen versehen. `static` ist der Standardwert, den der Browser auch anwendet, wenn Sie keine Positionsart angeben. Daher bleibt das Element im normalen Textfluss. Die Angaben für `top` und `left` werden ignoriert.

Das Element ④ wird relativ positioniert. Normalerweise wäre der Textfluss hinter der Zeile Stelle für Wert `relative`: fortgesetzt worden. Mit der gewählten Positionsart verschieben Sie das formatierte Inline-Element von dieser Position um 200 Pixel nach rechts und 100 Pixel nach unten.

Die absolute Positionierung ⑤ wird im Gegensatz zu den bisherigen Elementen völlig unabhängig von der eigentlichen Position betrachtet.

Das Inline-Element wird hier vom oberen rechten Rand des Elementes `body` positioniert. Wird die Webseite gescrollt, bewegt sich auch dieses Element mit.

Element ⑥ dagegen bewegt sich beim Scrollen nicht mit dem übrigen Text – es bleibt immer im sichtbaren Bereich. Wie bei der absoluten Positionierung wird auch die fixierte Positionierung unabhängig von der eigentlichen Position betrachtet. Die Positionsangaben beziehen sich hier auf den Rand des Browserfensters.

10.2 Größe und Seitenverhältnis eines Elements angeben

Breite festlegen

Blockelemente nehmen die gesamte Breite des Elternelementes ein. Die Breite von Inline-Elementen wird nur von der Menge des Inhalts bestimmt. Je mehr Inhalt, desto breiter das Element. Mit der CSS-Eigenschaft `width` können Sie die Breite des Elements festlegen.

```
width: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ Zahlenangaben mit entsprechender Maßeinheit ✓ Prozentuale Angaben relativ zur Breite des Elternelements ✓ Angaben in <code>vw</code> oder <code>vh</code> mit Bezug zum Viewport ✓ <code>auto</code> = automatische Festlegung (inklusive Innenabstand und Rahmen wird die volle Breite des Elternelementes ausgefüllt)
Standardwert	Auto
Vererbbar	Nein
Anwendbar auf	Alle Elemente außer Tabellenzeilen (<code>tr</code>)

Höhe festlegen

Neben der Breite eines Elements können Sie auch dessen Höhe bestimmen.

Legt man die Höhe und die Breite eines Elements fest, kann der Browser diese in keine Richtung eigenständig anpassen, wenn der Inhalt der Box einmal mehr Raum benötigt, als die Box bereitstellt. Daher sollten Sie dem Browser mit der Eigenschaft `overflow` (siehe unten) mitteilen, wie er mit Inhalten umgehen soll, die nicht in die Box passen.

Text kann aus zwei Gründen nicht in eine Box passen: einerseits, wenn Sie mehr Text in eine Box schreiben als hineinpasst. Dann haben Sie es in der Hand, die Größe der Box oder die Menge des Textes anzupassen. Andererseits können Nutzer die Schrift vergrößern, um diese besser zu lesen. Dann passt ein Text unter Umständen nicht mehr in eine Box. Das können Sie mit von der Schriftgröße abhängigen Einheiten wie `ex`, `em` und `rem` vermeiden. Wenn Sie diese verwenden, wächst und schrumpft die Box zusammen mit dem Text.

```
height: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ Zahlenangaben mit entsprechender Maßeinheit ✓ Prozentuale Angaben relativ zur Höhe des Elternelements ✓ Angaben in vw oder vh mit Bezug zum Viewport ✓ auto = automatische Festlegung (Element ist so hoch wie sein Inhalt)
Standardwert	Auto
Vererbbar	Nein
Anwendbar auf	Alle Elemente außer Inline-Elementen

Beispiel: *kap10\width-height.htm*

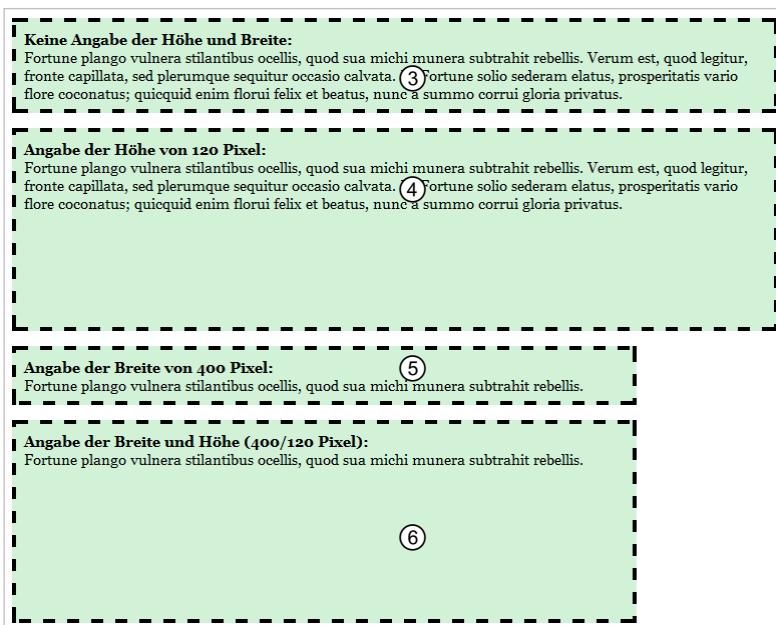
Im Folgenden wird ein Dokument mit vier Absätzen erstellt. Damit die Wirkungsweise der Eigenschaften für die Höhe und Breite eines Elements sichtbar wird, ordnen Sie jedem Absatz unterschiedliche Höhen und Breiten zu.

	<pre><html lang="de"> <head> <title>Höhen und Breiten von Elementen</title> ① <link rel="stylesheet" href="standard.css"> <style></pre>
②	<pre>p { position: relative; left: 10px; top: 10px; border: 3px dashed black; padding: 5px; margin-right: 20px; background-color: #D2F2D8; }</pre>
③	<pre></style> </head> <body></pre>
④	<pre><p>Keine Angabe der Höhe und Breite:
Fortune plango...</p></pre>
⑤	<pre><p style="height: 120px;">Angabe der Höhe von 120 Pixel:
Fortune plango vulnera stilantibus ocellis, quod sua michi munera...</p></pre>
⑥	<pre><p style="width: 400px;">Angabe der Breite von 400 Pixel:
Fortune plango vulnera stilantibus ocellis, quod sua michi munera...</p></pre>
	<pre><p style="width: 400px; height: 120px;">Angabe der Breite und Höhe (400/120 Pixel):
Fortune plango vulnera stilantibus ocellis,...</p> </body> </html></pre>

Unterschiedlich hohe und breite Absätze (*kap10\width-height.htm*)

- ① Die Standardformatierungen für das Dokument binden Sie über eine externe Datei ein.

- ② Damit die Höhen und Breiten der Absätze sichtbar sind, fügen Sie einen Rahmen und eine grünliche Hintergrundfarbe hinzu. Die Positionsangaben von `left` und `top` erfolgen relativ zur eigentlichen Position.
- ③ Der erste Absatz erhält keine Höhen- oder Breitenangaben. Die Ausmaße werden entsprechend der Browserfenstergröße und der Länge des Inhalts vom Browser festgelegt.
- ④ Weisen Sie diesem Absatz eine Höhe von 120 Pixeln zu. Auch wenn diese Höhe durch den Inhalt nicht erreicht wird, zeigt der Browser den Absatz in der Höhe an. Dadurch entsteht zusätzlicher Freiraum. Die Breite ist auf 100 % des Elternelementes festgelegt. Der Browser kann diesen Absatz nicht vergrößern. Befüllen Sie diesen Absatz mit mehr Text, wird dieser über den Rand der Box hinauslaufen und andere Elemente der Seite überlagern.
- ⑤ Der nächste Absatz erhält eine Beschränkung in der Breite. Egal, ob Sie das Browserfenster breiter oder schmäler ziehen, die Breite des Elements bleibt bestehen. Die Höhe wird wieder dem Inhalt angepasst, sodass dieser Absatz beliebig viel Text aufnehmen kann. Wenn das Browserfenster schmäler ist als dieser Absatz, sind Teile des Textes nicht mehr sichtbar.
- ⑥ Den letzten Absatz beschränken Sie in der Höhe und in der Breite. Dadurch nimmt dieser nur noch einen Teil der Bildschirmbreite ein und erhält aufgrund der Höhe einen zusätzlichen Freiraum. Auch hier kommt es zu denselben Problemen wie beim Absatz ④.



Maximalwerte festlegen

```
/* Element kann breiter, aber nicht schmäler sein */
min-width: Wert;

/* Element kann schmäler, aber nicht breiter sein */
max-width: Wert;

/* Element kann höher, aber nicht niedriger sein */
min-height: Wert;

/* Element kann niedriger, aber nicht höher sein */
max-height: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ Zahlenangaben mit entsprechender Maßeinheit; standardmäßig wird die Maßeinheit Pixel (px) verwendet. ✓ Prozentuale Angaben zur Höhe und Breite des Elternelements ✓ Angaben in vw oder vh mit Bezug zum Viewport ✓ none = keine Festlegung (für max-height und max-width)
Standardwert	none bzw. 0
Vererbbar	Nein
Anwendbar auf	Alle Elemente außer Tabellenelementen

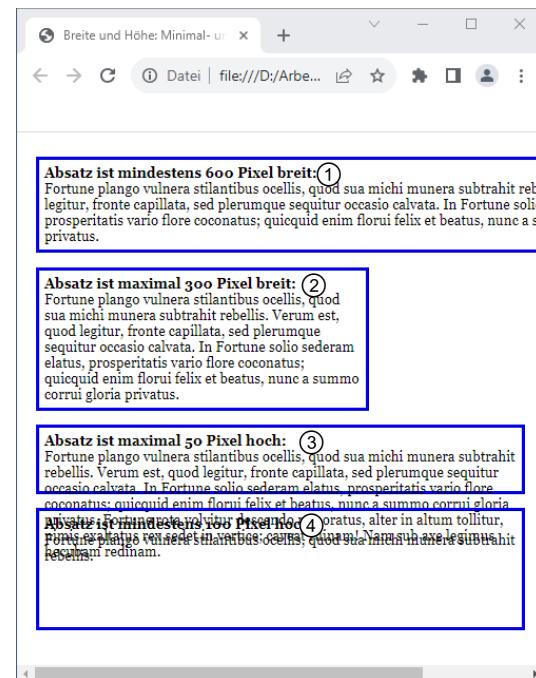
Wenn Sie das Beispiel *width-height.htm* etwas umwandeln und in den einzelnen Absätzen statt der Breiten und Höhen die Minimal- und Maximalwerte angeben, erhalten Sie das Ergebnis wie in der nebenstehenden Abbildung (*minmax.htm*).

Für den Absatz ① legen Sie mit `min-width: 600px` eine Breite von mindestens 600 Pixeln fest. Dies bedeutet, dass die Elementbreite nicht kleiner als 600 Pixel wird. **Achtung: Wenn das Browserfenster verkleinert wird, sind einige Teile nicht mehr im sichtbaren Bereich** (siehe auch nächster Abschnitt „Überlauf festlegen“). Ist das Browserfenster breiter als 600 Pixel, nimmt der Absatz die gesamte Breite ein, verhält sich also wie Absätze ohne Breitenangabe.

Absatz ② ist mit `max-width: 300px` so beschränkt, dass das Element nicht breiter als 300 Pixel werden kann. Eine automatische Verkleinerung durch den Browser ist erlaubt.

Der dritte Absatz ③ ist mit `max-height: 50px` in der Höhe beschränkt, sodass selbst bei größerem Inhalt das Element nur mit einer Höhe von 50 Pixeln dargestellt wird. **Achtung: Passt der Text nicht hinein, überlagert er nachfolgende Elemente** (siehe auch nächster Abschnitt „Überlauf festlegen“).

Mit dem Absatz ④ definieren Sie keine Breiten-, sondern eine Höhenbeschränkung. Die Angabe von `min-height: 100px` setzt die Höhe des Elements auf mindestens 100 Pixel. Benötigt der Inhalt mehr Platz, wird die Elementhöhe automatisch vergrößert.



Seitenverhältnis festlegen

Das Seitenverhältnis eines Elements legen Sie mit der Eigenschaft `aspect-ratio` fest.

```
aspect-ratio: Wert;
```

Werte	✓ Verhältniswert Breite/Höhe ✓ auto für die Übernahme der Größe von Bildern, Mediendateien und anderen ersetzen Elementen
Standardwert	auto
Vererbbar	Nein
Anwendbar auf	Alle Elemente

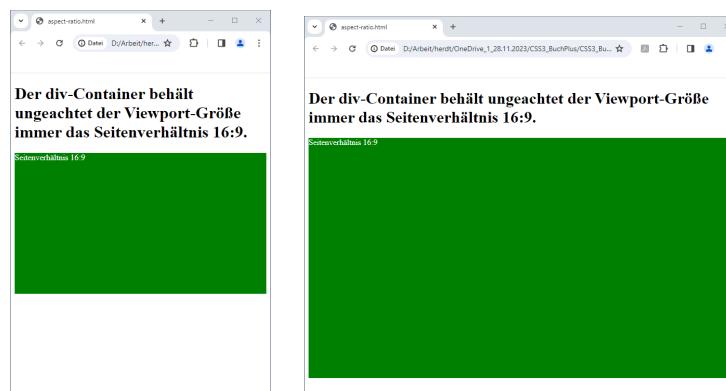
Beispiel: *kap10\aspect-ratio.htm*

Im Folgenden wird ein `<div>`-Container erstellt und mit dem Seitenverhältnis 16:9 versehen. Dieses wird ungeachtet der Abmessungen und Proportionen des Browserfensters beibehalten.

<pre> <html lang="de"> <head> <title>Seitenverhältnis von Elementen festlegen</title> <style> .container { background-color: green; color: white; aspect-ratio: 16 / 9; } </style> </head> <body> <h1> Der div-Container behält ungeachtet der Viewport-Größe immer das Seitenverhältnis 16:9. </h1> <div class="container"> <p>Seitenverhältnis 16:9</p> </div> </body> </html> </pre>

Container mit festem Seitenverhältnis definieren (kap10\aspect-ratio.htm)

- ① In der CSS-Stildefinition der Klasse `.container` wird unter anderem das Seitenverhältnis 16:9 definiert.
- ② Die Klasse `.container` wird dem `<div>`-Element im `<body>` des Dokuments zugewiesen.



10.3 Überlauf festlegen

In der Abbildung des Abschnitts „Maximalwerte festlegen“ weiter oben können Sie sehen, dass im vorletzten Absatz der Text über den Rahmen hinausläuft. Dieses Verhalten können Sie mit der Eigenschaft `overflow` beeinflussen.

overflow: Wert;

Werte	<ul style="list-style-type: none"> ✓ <code>visible</code> = der gesamte Inhalt ist sichtbar, unter Umständen außerhalb des Elementes ✓ <code>hidden</code> = überfließender Inhalt wird abgeschnitten und nicht dargestellt ✓ <code>auto</code> = überfließender Inhalt wird mittels Scrollbalken erreichbar; Bildlaufleisten werden nur eingeblendet, wenn der Inhalt größer ist als das Element ✓ <code>scroll</code> = horizontale und vertikale Bildlaufleisten werden permanent eingeblendet, auch wenn der Inhalt in die Box passt
Standardwert	<code>visible</code>
Vererbbar	Nein
Anwendbar auf	Alle Blockelemente

Beispiel: *kap10\overflow.htm*

Den vier Absätzen weisen Sie die unterschiedlichen Werte der Eigenschaft `overflow` zu, um die Auswirkungen auf die Darstellung sichtbar zu machen.

```

<html lang="de">
<head>
  <title>overflow: Überlauf festlegen</title>
  <link rel="stylesheet" href="standard.css">
  <style>
    ① p { position: relative; left: 10px; top: 10px; border: 3px
          ridge blue;
          padding: 5px; margin-right: 20px; background-color: #D6EDF4;
          height: 55px;
        }
    </style>
  </head>
  <body>
    <h1>Festlegen des Überlaufs</h1>
    ② <p style="overflow: visible"><b>Überlauf ist visible:</b><br>Fortune...</p>
  </body>
</html>

```

```

③ <p style="overflow: hidden"><b>Überlauf ist hidden:</b>
</p><br>Fortune...</p>

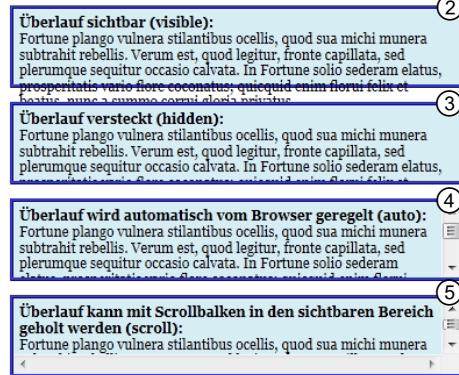
④ <p style="overflow: auto"><b>Überlauf ist auto:</b>
</p><br>Fortune plango...</p>

⑤ <p style="overflow: scroll"><b>Überlauf ist scroll:</b>
</p><br>Fortune... </p>
</body>
</html>

```

- ① Legen Sie die Formatierungen der Absätze fest. Als Höhe geben Sie 55 Pixel an.
- ② Mit `overflow:visible` legen Sie fest, dass der Inhalt des Absatzes immer sichtbar sein soll, unabhängig davon, wie lang dieser ist. Dabei kann der Text von nachfolgenden Elementen verdeckt werden oder selber andere Elemente überlagern und so unleserlich machen.
- ③ Der Wert `hidden` dagegen lässt den Browser überfließende Inhalte abschneiden. Nachfolgende Texte oder Bilder sind nicht zugänglich.
- ④ Mit dem Wert `auto` veranlassen Sie den Browser, eine Bildlaufleiste **bei Bedarf** einzublenden, wenn die Inhalte also nicht in die Box passen. Die im CSS definierten Angaben für Breite und Höhe werden vom Browser strikt eingehalten.
- ⑤ Bildlaufleisten blenden Sie immer mit dem Schlüsselwort `scroll` ein. Auch hier bleiben die definierten Dimensionen des Elements bestehen.

Festlegen des Überlaufs



Verschiedene Arten des Überlaufs
(kap10\overflow.htm)

Wie Sie den Beispielen entnehmen können, lassen sich Boxen sehr gut mit der Einheit Pixel so formatieren, dass die enthaltenen Texte nicht mehr hineinpassen. Boxen, die zu klein für ihre Inhalte sind, lassen sich sehr gut zur Demonstration der CSS-Eigenschaft `overflow` verwenden. Auf einer echten Webseite möchte man allerdings keine abgeschnittenen oder nur durch Scrollen erreichbaren Inhalte. Entsprechend ungeeignet sind alle Einheiten, die sich nicht auf die Textgröße beziehen, wie `vw`, `vh`, Prozent oder alle absoluten Einheiten.

Gut geeignet sind dagegen `em`, `rem` oder `ex`.

10.4 Textüberlauf beeinflussen

Nutzen Sie die Eigenschaft `text-overflow`, um einzeilige Texte zu begrenzen und deren Aussehen zu beeinflussen.

```
text-overflow: Wert;
```

Werte	✓ <code>clip</code> = überfließender Inhalt wird abgeschnitten und nicht dargestellt ✓ <code>ellipsis</code> = wie <code>clip</code> , aber abgeschnittener Inhalt wird durch drei Punkte repräsentiert
Standardwert	<code>clip</code>
Vererbbar	Nein
Anwendbar auf	Alle Blockelemente

Die Eigenschaft `text-overflow` können Sie nur auf einzeilige Texte anwenden. Daher sollten Sie mittels `white-space: nowrap` ungewollte Zeilenumbrüche verhindern, wenn Sie `text-overflow` einsetzen.

Beispiel: *kap10\text-overflow.htm*

Den vier Absätzen weisen Sie die unterschiedlichen Werte der Eigenschaft `overflow` zu, um die Auswirkungen auf die Darstellung sichtbar zu machen.

```
<html lang="de">
<head>
  <title>overflow: Überlauf festlegen</title>
  <link type="text/css" rel="stylesheet" href="standard.css">
  <style type="text/css">
    p {
      width: 100px;
      ① overflow: hidden;
      ② white-space: nowrap;
      ③ ④ text-overflow: ellipsis;
    }
  </style>
</head>
<body>
  <h1>Beeinflussen des Text-Überlaufs</h1>
  ⑤ <p>Donaudampfschiffahrtsgesellschaft</p>
</body>
</html>
```

- ① Beschränken Sie den verfügbaren Platz.
- ② Verstecken Sie überfließenden Inhalt.
- ③ Verhindern Sie, dass der Inhalt durch Zeilenumbrüche im sichtbaren Bereich gehalten werden könnte.
- ④ Notieren Sie `text-overflow: ellipsis`, um den abgeschnittenen Inhalt durch drei Punkte (...) zu ersetzen.
- ⑤ Der Absatz wird einzeilig dargestellt, der Inhalt beschnitten und es werden drei Punkte hinzugefügt.

⑤ **Donaudampfsch...**

(*kap10\text-overflow.htm*)

10.5 Element umfließen

Der normale Elementfluss sieht vor, dass Blockelemente in der Reihenfolge, wie sie im Quelltext auftauchen, aufeinanderfolgen. Das heißt, Absätze werden untereinander dargestellt.

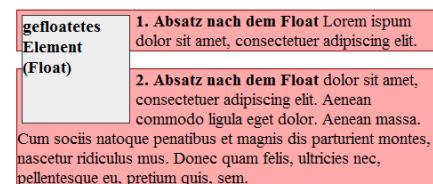
Mit der CSS-Eigenschaft `float` können Sie Elemente wie Bilder oder Absätze, aber auch komplexe Strukturen wie Menüs und Infoboxen, von den Inhalten der **nachfolgenden** Elemente umfließen lassen.

```
float: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ <code>left</code> = Element steht links und wird rechts umflossen ✓ <code>right</code> = Element steht rechts und wird links umflossen ✓ <code>none</code> = Element wird nicht umflossen
Standardwert	None
Vererbbar	Nein
Anwendbar auf	Alle, bis auf absolut positionierte Elemente

Damit die Eigenschaft `float` überhaupt eine **sichtbare** Auswirkung hat, müssen Sie für das Element eine Breite festlegen (z. B. 50%). Denn nur wenn neben dem Float Platz ist, können daneben weitere Inhalte stehen.

Wie die Abbildung zeigt, befinden sich die umfließenden Elemente (erster und zweiter Absatz) nicht neben dem Float, sondern **dahinter**.



Lediglich die Inhalte der Absätze werden verschoben: Der Float gibt den ihm zustehenden Platz komplett frei, die nachfolgenden Elemente rutschen entsprechend weit nach oben, die darin enthaltenen Texte und Bilder werden in den sichtbaren Bereich gerückt.

Um den normalen Elementfluss wiederherzustellen, verwenden Sie die Eigenschaft `clear`.

```
clear: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ <code>left</code> = stellt den normalen Elementfluss nach Verwendung von <code>float: left</code> wieder her ✓ <code>right</code> = stellt den normalen Elementfluss nach Verwendung von <code>float: right</code> wieder her ✓ <code>both</code> = stellt den normalen Elementfluss nach Verwendung von <code>float: left</code> oder <code>float: right</code> wieder her ✓ <code>none</code> = <code>float</code> behält seine Gültigkeit
Standardwert	None
Vererbbar	Nein
Anwendbar auf	Alle, bis auf absolut positionierte Elemente

Wenn Sie einem Element die Eigenschaft `clear` mit einem entsprechenden Wert zuweisen, ist es das erste Element, das die Höhe des Floats berücksichtigt und erst darunter angezeigt wird.

Beispiel: `kap10\float.htm`

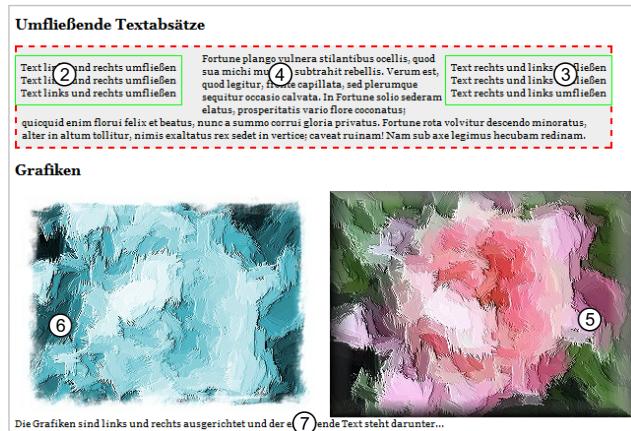
In dem Beispiel wird mehreren Elementen einer Webseite die Eigenschaft `float` zugewiesen. Das Beispiel ist als abgedruckter Quelltext leicht gekürzt.

```
<html lang="de">
<head>
    <title>Texte umfließen</title>
    <link rel="stylesheet" href="standard.css">
    <style>
        ① .absatz {
            border: 1px solid 0f0;
            padding: 5px;
            margin-right: 20px;
            background-color: #eee;
        }
    </style>
</head>

<body>
    <h1>Umfließende Textabsätze</h1>
    ② <p class="absatz" style="float: left">
        Text links und rechts umfließen<br>
        Text links und rechts umfließen<br>
        Text links und rechts umfließen
    </p>
    ③ <p class="absatz" style="float: right">
        Text links und rechts umfließen<br>
        Text links und rechts umfließen<br>
        Text links und rechts umfließen
    </p>
    ④ <p class="absatz">Fortune plango vulnera stilantibus ocellis,
        quod...</p>

        <h1>Grafiken</h1>
        <p>
            ⑤ 
            ⑥ 
            ⑦ <span style="clear: both">
                Die Grafiken sind links und rechts ausgerichtet und der
                erklärende Text steht darunter...
            </span>
        </p>
    </body>
</html>
```

- ① Damit Sie erkennen, wie sich Texte umfließen können, notieren Sie für die Klasse `absatz` einige Eigenschaften. Elemente mit dieser Klasse erhalten einen grünen Rahmen und eine hellgraue Hintergrundfarbe.
- ② Der erste Absatz erhält die Formatzuweisung des Selektors `.absatz`. Zusätzlich wird er links ausgerichtet (`float: left`). Die Inhalte der nachfolgenden Elemente umfließen ihn damit auf der rechten Seite.
- ③ Ein weiterer Absatz erhält die Formatierung, die Sie für `.absatz` notiert haben, und wird rechts ausgerichtet.
- ④ Die Absätze ② und ③ verzichten auf ihren Platz. Absatz ④ rutscht an deren Position, was anhand des roten, gestrichelten Rahmens deutlich wird. Der gesamte Text bleibt aber im sichtbaren Bereich, umfließt also die vorhergehenden Floats, sofern die Breite des Browserfensters dafür ausreicht.
- ⑤ Die Eigenschaft `float` kann auch auf Grafiken angewendet werden, hier eine rechts ausgerichtete Grafik.
- ⑥ Eine weitere Grafik erhält `float: left` und erscheint links.
- ⑦ Im selben Absatz zeichnen Sie Text mit `` aus. Normalerweise würde dieses Element wie in ④ zwischen den Grafiken hindurchfließen. Mit der Eigenschaft `clear` beenden Sie den Textfluss. Der Wert `both` legt dabei fest, dass beide Textflüsse zu beenden sind.



Verschiedene Elemente können umflossen werden
(kap10\float.htm)

! Die Eigenschaft `clear` funktioniert nur bei Block-Level-Elementen. Sie müssen also mittels `display: block` dafür sorgen, dass sich das Inline-Element `span` wie ein Blockelement verhält.

10.6 Elemente verbergen

Mit der CSS-Eigenschaft `visibility` können Sie beliebige Elemente sichtbar oder unsichtbar setzen. Mit `unsichtbar` ist gemeint, dass das Element transparent dargestellt wird und damit die Sicht auf eventuell dahinterliegende Elemente freigibt. Der Platz, der nötig wäre, um das Element darzustellen, wird **nicht** freigegeben. Das Element ist also weiterhin vorhanden und nimmt dieselbe Höhe und Breite für sich in Anspruch, als wäre es sichtbar.

```
visibility: Wert;
```

Werte	<ul style="list-style-type: none"> ✓ <code>visible</code> = Element ist sichtbar ✓ <code>hidden</code> = Element nicht sichtbar ✓ <code>collapse</code> = anwendbar auf Tabellenspalten und -zellen ✓ <code>inherit</code> = die Sichtbarkeit wird vom übergeordneten Element geerbt
Standardwert	Inherit
Vererbbar	Nein
Anwendbar auf	Alle Elemente

! Der Wert `collapse` soll eine Tabellenzeile oder -spalte komplett ausblenden, so als wäre `display: none` angewendet worden. Allerdings soll im Unterschied zu `display: none` die Aufteilung der restlichen Tabelle nicht neu berechnet werden. Es sind unterschiedliche Darstellungen in den verschiedenen Browsern bekannt. Wenn Sie es dennoch einsetzen möchten, sind umfangreiche Browsertests nötig.

Beispiel: *kap10\visibility.htm*

Es wird eine Webseite erstellt, die einen Absatz und eine Grafik enthält, die auf unsichtbar gestellt werden. Im zweiten Teil des Beispiels wenden Sie die Werte auf die Zeilen einer Tabelle an. Beachten Sie die unterschiedliche Darstellung in den verschiedenen Browsern.

```

<html lang="de">
<head>
  <title>Elemente ausblenden</title>
①  <link rel="stylesheet" href="standard.css">
</head>
<body>
  <h1>Elemente ausblenden</h1>
  <p>Der nachfolgende Leerraum ist nicht wirklich leer. Dies
  ist ein unsichtbarer Absatz mit einer separaten Grafik.</p>
② <p style="visibility:hidden">Text und Grafik, die beide nicht
  angezeigt werden.
③ 
</p>
<p>
  Die Sichtbarkeit der Grafik ist vom übergeordneten Absatz
  abhängig.
</p>

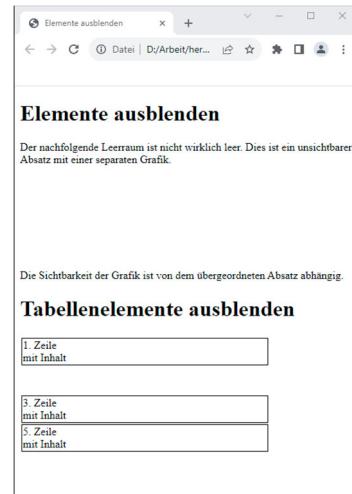
  <h1>Tabellenelemente ausblenden</h1>
  <table>
    <tr><td>1. Zeile</td></tr>
④  <tr style="visibility:hidden">
    <td>2. Zeile</td>
  </tr>
  <tr><td>3. Zeile</td></tr>

```

⑤

```
<tr style="visibility:collapse">
  <td>4. Zeile</td>
</tr>
<tr><td>5. Zeile</td></tr>
</table>
</body>
</html>
```

- ① Die grundlegenden Formatierungen der Webseite nehmen Sie durch die Einbindung der externen CSS-Datei *standard.css* vor.
- ② Im ersten Teil fügen Sie zwischen zwei Absätzen einen weiteren Absatz mit Text und Grafik ein. Über *visibility: hidden* setzen Sie ihn unsichtbar.
- ③ Auch die im Absatz enthaltene Grafik wird auf „nicht sichtbar“ gesetzt.
- ④ Innerhalb einer Tabelle können Sie auch über den Wert *hidden* eine Textzeile ausblenden. Das Problem ist, dass der eigentlich benötigte Platz der Zeile ungenutzt bleibt und somit als großer Zeilenabstand bestehen bleibt.
- ⑤ In Tabellen ist es daher besser, den Wert *collapse* zu verwenden, um Zeilen oder Spalten auszublenden. Der Vorteil ist, dass auch der sonst sichtbare Leerraum verschwindet.

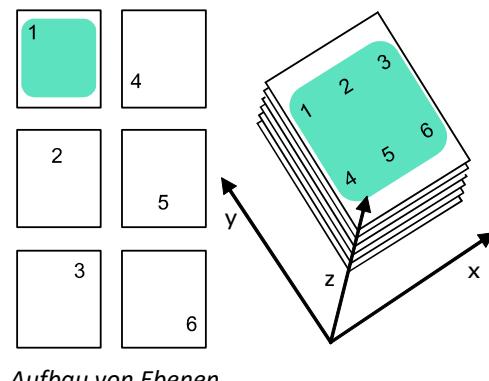


10.7 Ebenen in HTML-Dokumenten meistern

Stellen Sie sich die Ebenen wie Folien vor, die übereinandergelegt werden.

Die erste (unterste) Folie ist die 1. Ebene, die darüberliegende Folie ist die 2. Ebene usw. Da die Ebenen prinzipiell durchsichtig sind, sind die Inhalte der einzelnen Ebenen sichtbar.

Dabei ist auch der Effekt möglich, dass der Inhalt einer Ebene den Inhalt einer anderen Ebene überlagert.



Wenn Sie Elemente absolut oder fixiert positionieren, werden diese unabhängig von anderen Elementen ausgerichtet und können dadurch Elemente der darunterliegenden Ebenen verdecken. Die Reihenfolge der Ebenen können Sie mit der folgenden CSS-Eigenschaft beeinflussen.

```
z-index: Wert;
```

Werte	❖ Positive und negative Ganzzahlen ❖ auto = die Ebenen werden in der Reihenfolge der Definition festgelegt
Standardwert	auto
Vererbbar	Nein
Anwendbar auf	Alle positionierten Elemente

Stellen Sie sich das Positionieren der Elemente wie in einem dreidimensionalen Koordinatensystem vor (siehe Abbildung *Aufbau von Ebenen*). Der Wert von **z-index** beschreibt dabei die z-Koordinate und muss eine Ganzzahl sein. Dezimalzahlen (mit Nachkommastellen) sind nicht möglich. Elemente mit höheren Werten überdecken Elemente mit niedrigeren Werten.

Beispiel: *kap10\z-index.htm*

Verschiedene Block-Level-Elemente werden so angeordnet, dass sie sich teilweise überlappen. Die Reihenfolge der Elemente legen Sie abweichend von der automatischen Stapelung fest.

```
<html lang="de">
<head>
  <title>Ebenen mit z-index</title>
  <link rel="stylesheet" href="standard.css">
  <style>
    ① div {
      position: absolute;
      border: 1px solid #fff;
      color: #fff;
      font-weight: bold;
      padding: 2px 5px;
    }
    ② .nr1 {
      top: 50px;
      left: 50px;
      width: 350px;
      height: 275px;
      background-color: #f00;
      z-index: 10;
    }
    ③ .nr2 {
      top: 200px;
      left: 75px;
      width: 300px;
      height: 100px;
      background-color: #0f0;
      z-index: 30;
    }
  </style>
</head>
<body>
  <div>1</div>
  <div>2</div>
  <div>3</div>
</body>

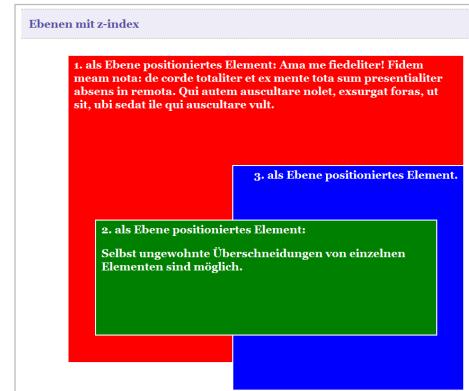
```

```

④ .nr3 {
    top: 150px;
    left: 200px;
    width: 200px;
    height: 200px;
    background-color: #00f;
    text-align: right;
    z-index: 20;
}
</style>
</head>
<body>
<h1>Ebenen mit z-index</h1>
⑤ <div class="nr1">
    1. als Ebene positioniertes Element: Ama me fiedeliter!
        Fidem meam nota: de corde totaliter et ex mente tota...
</div>
<div class="nr2">
    2. als Ebene positioniertes Element:
    <p>
        Selbst ungewohnte Überschneidungen von einzelnen
        Elementen sind möglich.
    </p>
</div>
<div class="nr3">3. als Ebene positioniertes Element.</div>
</body>
</html>

```

- ① Nach dem Einbinden der Standardformatierungen legen Sie fest, dass alle div-Elemente absolut zu positionieren sind. Zur besseren Hervorhebung erhalten alle div-Elemente einen weißen Rand und eine Hintergrundfarbe.
- ② Über die Klasse nr1 legen Sie die Position und die Höhe und Breite des ersten Elements fest. Die Hintergrundfarbe ist Rot. Das Element, das diese Klasse zugewiesen bekommt, erhält den Ebenenwert 10.
- ③ Für den Absatz Nummer zwei verwenden Sie die Klasse nr2. Dieser Absatz erhält einen grünen Hintergrund. Als Wert für den z-Index geben Sie 30 an. Damit wird dieses Element über das Element mit der Klasse nr1 gelegt.
- ④ Die letzte Definition bewirkt einen blauen Hintergrund. Der Text wird rechts ausgerichtet. Mit einem z-index-Wert von 20 liegt das formatierte Element genau zwischen den beiden Elementen mit den Klassen nr1 und nr2. Es überlappt dann zwar das Element mit der Klasse nr1, wird jedoch selbst teilweise vom Element mit der Klasse nr2 überdeckt.
- ⑤ Es folgt die Zuweisung der einzelnen Selektoren und somit der Formate.



*Einander überlagernde Ebenen
(kap10\z-index.htm)*

Gerade wenn Sie mit mehreren positionierten Elementen arbeiten, kann es sein, dass Sie später ein oder zwei weitere Elemente in den Ebenenstapel einfügen möchten. Dabei hilft es, wenn Sie wie hier im Beispiel Zehnerschritte zur Festlegung des z-index benutzt haben. So ist es einfach, zwischen dem Element mit dem z-index: 20 und dem mit dem z-index: 30 ein Element mit dem z-index: 25 hinzuzufügen. Bei fortlaufenden Werten wie 1, 2, 3 wäre das nur durch erneutes Durchnummerieren sowohl im CSS als auch im HTML möglich.



Absolute Positionierungen sind allgemein und bei responsiven Seiten im Besonderen eine große Herausforderung. Es ist nahezu unmöglich, für jede Bildschirmauflösung- und Schriftgrößen-Kombination sicherzustellen, dass sich Elemente nicht so überdecken, dass Teile der Webseite unerreichbar werden.

10.8 Übungen

Übung 1: Fragen zu Ebenen

Level		Zeit	ca. 7 min
Ergebnisdatei	kap10\uebung1-2.doc		

1. Nennen Sie die Unterschiede einer statischen, relativen, absoluten und fixierten Positionierung.
2. Welche Eigenschaftswerte stehen Ihnen zur Verfügung, um die Sichtbarkeit eines Elements zu beeinflussen?

Übung 2: Text um anderen Text fließen lassen

Level		Zeit	ca. 15 min
Ergebnisdatei	kap10\uebung3.htm		

1. Erstellen Sie die nachfolgende Webseite, bei der sich zwei Texte umfließen. Setzen Sie den linken Text zur Hervorhebung in einen separaten Rahmen. Die Breite beträgt 30 % des Browserfensters.

Textfluss als Übung

Omnia Sol temperat purus et subtilis, novo mundo reserat faciem Aprilis.

Qui autem auscultare nolet, exsurgat foras, ut sit, ubi sedat ile qui auscultare vult. Id nos Latine gloriosum dicimus. Qua adseditis causa in festivo loco, comoedias quam nos actuari sumus et argumentum et nomen vobis eloquar. Alazon Graece huic nomen est comoediae: Id nos Latine gloriosum dicimus. Flora veris leta facies mundo propinatur, hiemalis acies victa iam fugatur. In vestitu vario Flora principatur, nemorum dulcisono que cantu celebratur. Flore fusus gremio Phebus novo more risum dat, hic vario iam stipate flore.

Ein Textabsatz umfließt einen anderen Absatz ([kap10\uebung3.htm](#)).

11

Inhalte generieren: Zähler, Variablen und Berechnungen

11.1 Einführung in CSS-generierte Inhalte

Bis auf die Nummerierungen über Listen, bei denen der Browser die automatische Zählung der Einträge übernimmt, gibt es keine weiteren Möglichkeiten, Inhalte automatisch über die HTML-Syntax in Webseiten einzufügen zu lassen. So könnte beispielsweise unter Grafiken immer das Wort „Abbildung“ eingefügt werden, bevor der Name der Grafik erscheint. Außerdem könnten, wie in diesem Buch, einzelne Kapitel und Abschnitte automatisch durchnummierter werden. Automatische Nummerierung und mehr sind mit CSS möglich.

11.2 Anführungszeichen setzen

Über die HTML-Tags `<q>` und `</q>` können Sie Anfang und Ende von Zitaten markieren. Zitate werden dann in deutschen Texten automatisch mit Anführungszeichen gekennzeichnet. Die Kennzeichnung unterscheidet sich jedoch ein wenig, weil hierfür die typografischen Anführungszeichen („ „) verwendet werden. In anderen Ländern, wie in Frankreich oder der Schweiz, werden Anführungszeichen etwa im Format » « oder « » verwendet.

Das entsprechende Anführungszeichen können Sie in CSS über die Eigenschaft `quotes` festlegen.

```
quotes: Startzeichen Endezeichen;
```

Werte	<ul style="list-style-type: none">✓ Jedes beliebige Zeichen✓ Ein oder mehrere Paare von Anführungszeichen werden angegeben.✓ Getrennt werden sie durch Leerzeichen.✓ Das erste Zeichen ist das öffnende, das zweite das schließende Zeichen.✓ Jedes weitere Paar bestimmt die Anführungszeichen innerhalb der vorherigen.
Standardwert	Vom Browser abhängig

Vererbbar	Ja
Anwendbar auf	Alle Elemente

- ✓ Zusätzlich können Sie die Werte `open-quote` und `close-quote` mit der später erklärten CSS-Eigenschaft `content` verwenden, um z. B. ein öffnendes oder schließendes Anführungszeichen als zusätzliche Inhalte einzufügen.
- ✓ Einige Zeichen können nicht über die Tastatur eingegeben werden. Über den entsprechenden ISO-Code oder Unicode haben Sie eine Möglichkeit, jedes beliebige Zeichen anzugeben.

Anführungszeichen	Erklärung	ISO 8859-1	Unicode
"	Doppeltes Anführungszeichen	" ;	0022
'	Einfaches Anführungszeichen (Apostroph)	' ;	0027
< >	Angewinkelte Anführungszeichen links und rechts (französisch)	‹ ; (‹ ;) › ; (› ;)	2039 203A
« »	Angewinkelte doppelte Anführungszeichen links und rechts (französisch)	« ; » ;	00AB 00BB
‘ ’	Einfaches Anführungszeichen links und rechts (englisch)	‘ ; (‘ ;) ’ ; (’ ;)	2018 2019
“ ”	Doppeltes Anführungszeichen links und rechts (englisch)	“ ; (“ ;) ” ; (” ;)	201C 201E

Eine Unicode-Zeichtentabelle finden Sie unter <https://unicode-table.com/de/>.

Die aufgeführten Anführungszeichen sind jeweils von der Sprache abhängig. Über die zusätzliche Angabe der Sprache können Sie die landestypischen Zeichen anwenden.

:lang (Kürzel)

Die Pseudoklasse `lang` ist die Abkürzung für das englische Wort „language“ (Sprache). Damit geben Sie an, in welcher Sprache der Inhalt eines HTML-Elementes verfasst wurde.

Beispiel

Die Angabe von `q : lang (de) { . . . }` bedeutet, dass eine Formatierung des HTML-Elementes `q` vorgenommen wird, wenn es als „deutsch“ ausgezeichnet ist. Weitere Kürzel sind¹:

- ✓ en = englisch
- ✓ fr = französisch
- ✓ no = norwegisch

¹ Liste aller Sprachkürzel unter <https://wiki.selfhtml.org/wiki/Sprachkürzel>

Damit die sprachspezifische Formatierung angewendet werden kann, müssen Sie im HTML-Grundgerüst der Webseite die Sprache angeben. Dieses geschieht über die Angabe des Attributs lang.

```
<html lang="de">
[...]
</html>
```

Somit erkennt der Browser, dass die Webseite in der deutschen Sprache verfasst ist, und kann dementsprechende sprachspezifische Formatierungen über die CSS-Eigenschaft vornehmen.

Beispiel: kap11\quotes.htm

Zwei Sätze, die eine wörtliche Rede enthalten, sollen automatisch in Anführungszeichen gesetzt werden. Zudem werden betonte Wörter mit einfachen Anführungszeichen hervorgehoben.

```
<html lang="de">
<head>
    <link rel="stylesheet" href="../standard.css">
    <style>
        ①      q:lang(de) { quotes: '“”’ ’“”; }
        q:lang(fr) { quotes: "«”» ’“’ ’“”; }
    </style>
    <title>Anführungszeichen: quotes</title>
</head>
<body>
    ②      <h1>Anführungszeichen: quotes</h1>
            <p><q lang='de'>Das kann doch nicht wahr sein!</q>, schrie
                er plötzlich laut heraus. <q lang='de'>So
                plötzlich, ohne <q lang='de'>Vorankündigung</q>? Das
                kann nicht wahr sein...</q>
            </p>
            <p>oder</p>
    ③      <p lang="fr">
            <q><q>L'Etat, c'est moi.</q> Voilà sans doute la citation
                la plus célèbre de l'Histoire de France, attribuée à Louis
                XIV, et censée illustrer la monarchie absolue.[...]</q>
            </p>
    </body>
</html>
```

- ① Nach Einbinden der Datei *standard.css* legen Sie fest, dass die von den Tags `<q>` und `</q>` umschlossenen Inhalte in Anführungszeichen gesetzt werden sollen. Dabei werden die doppelten Anführungszeichen verwendet. Befindet sich innerhalb des Zitates ein weiteres Element `q`, wird der Inhalt von einfachen Anführungszeichen umgeben.
- ② Die direkte Rede wird mit doppelten Anführungszeichen gekennzeichnet. Das Wort Vorankündigung wird zudem in einfache Anführungszeichen gesetzt.

- ③ Das französische Zitat wird in französischen Anführungszeichen ausgegeben, das darin enthaltene Zitat von Ludwig XIV. (L'Etat, c'est moi.) am Beginn und Ende in doppelten oberen Anführungszeichen.

Anführungszeichen: quotes

„Das kann doch nicht wahr sein!“, schrie er laut heraus.
„So plötzlich, ohne ‚Vorankündigung‘? Das kann nicht wahr sein ...“

oder

«"L'Etat, c'est moi." Voilà sans doute la citation la plus célèbre de l'Histoire de France, attribuée à Louis XIV, et censée illustrer la monarchie absolue.[...]»

Bildschirmausgabe der Datei „quotes.htm“

11.3 Inhalte einfügen

Im Kapitel über Selektoren haben Sie bereits die Pseudoelemente `::before` und `::after` kennengelernt. Dort finden Sie auch Beispiele, wie man mittels der CSS-Eigenschaft `content` Texte (z. B. drei Punkte) oder Bilder einfügt.

Hier eine Liste der möglichen Werte für `content`:

`content: Wert;`

Werte	<ul style="list-style-type: none"> ✓ Angabe von Text in Anführungszeichen (String) ✓ <code>counter()</code> = Zähler ✓ <code>open-quote</code> oder <code>close-quote</code>, um ein öffnendes oder schließendes Anführungszeichen (quotes) einzufügen ✓ <code>no-open-quote</code> und <code>no-close-quote</code>, um die Anzahl der verschachtelten Anführungszeichen hoch- bzw. herunterzählen ✓ <code>attr()</code> = Attributwert des im Selektor angegebenen Elements ✓ <code>url()</code> = Grafik einfügen ✓ Kombinationen der oben angegebenen Werte sind möglich
Standardwert	-
Vererbbar	Nein
Anwendbar auf	Die Pseudoelemente <code>::before</code> und <code>::after</code>

! Alle Inhalte gehören ins HTML. Ihr HTML-Dokument muss auch bei abgeschaltetem CSS und daher ohne CSS-generierte Inhalte verständlich sein.

CSS ist für die **Präsentation** der Inhalte zuständig. Es spricht nichts dagegen, mittels `::before` und `::after` Symbole hinzuzufügen, um die Bedeutung wiederkehrender Elemente hervorzuheben.

Häufig werden die Pseudoelemente beispielsweise genutzt, um den Listenpunkt in Listen durch eigene Symbole zu ersetzen (z. B. ein SVG-Icon). Das lässt sich zwar auch durch `list-style-image` erreichen, aber wie Sie in dem Beispiel dazu sehen können (siehe Kapitel 7, „Listen gestalten“), haben Sie dann keinerlei Möglichkeiten, die Darstellung weiter zu beeinflussen. Im Kapitel über Listen steht die eigene Grafik etwas höher als der Text. Es wäre zwar möglich, die Grafiken mit einem Aufwand passend zu erstellen, aber wenn Sie eine der bestehenden Icon-Bibliotheken mit mehreren hundert Zeichen verwenden wollen, macht es mehr Sinn, eine andere Technik zu wählen.

Die Pseudoelemente lassen sich genau wie echte Elemente formatieren, also auch positionieren. So können Sie die Position des Icons pixelgenau festlegen.

Beispiel: *kap11\list-bullets.htm*

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>Inhalte einfügen: eigene Listenpunkte</title>
    <style>
      ① li {
        position: relative;
        list-style: none;
        margin: 1em;
      }
      ② li::before {
        content: url(images/lnr-diamond.svg);
        display: inline-block;
        position: absolute;
        left: -2em;
        top: 0;
      }
    </style>
  </head>

  ③ <body>
    <h1>Inhalte einfügen: eigene Listenpunkte</h1>
    <ul>
      <li>Lorem ipsum dolor sit amet, consectetur...
      <li>Praesentium sapiente ex saepe facere nobis...
      <li>Architect possimus voluptas error, tenetur...
      <li>In maiores, placeat minima architect harum...
      <li>Dolor quibusdam perspiciatis mollitia fuga nulla...
    </ul>
  </body>
</html>
```

- ① Da die Grafiken als Listenpunkte dienen sollen, werden die Listenpunkte entfernt und alle Listeneinträge relativ positioniert. Dadurch können die per `content` eingefügten Bilder an den äußeren Kanten der Listeneinträge per absoluter Positionierung ausgerichtet werden.
- ② Das Pseudoelement `:before` wird mit dem Diamant-Icon gefüllt. Dank der absoluten Positionierung wird es aus dem normalen Elementfluss herausgenommen.

So führt es nicht zu einer Einrückung des Textes. Um diesen nicht zu überlagern, wird es um 2em nach links aus dem Listeneintrag geschoben. Der Abstand von oben ist in Ordnung. Deshalb nehmen Sie keine vertikale Verschiebung vor.

- ③ Es folgt der Dokumentenrumpf mit der gestalteten Liste.

So wie in dem Beispiel mit der Liste kann die Gestaltung von Elementen dazu beitragen, dass Seitenbesucher die Bedeutung von Elementen verstehen können.

Dank der Pseudoelemente `:before` und `:after` können Sie beliebige Gestaltungselemente oder Texthinweise zu wiederkehrenden Elementen wie Zitaten oder Buttons hinzufügen, um den Sinn solcher Elemente zu verdeutlichen.



CSS-generierte Inhalte werden **in dem ausgewählten Element** ausgegeben (nicht vor oder nach dem Element, sondern vor oder nach dem bereits vorhandenen Elementinhalt). Für einen Link bedeutet das, dass die CSS-generierten Inhalte unterstrichen werden, wenn der Link unterstrichen ist.

In eine Grafik können Sie keinen Text einbetten. Daher können Sie Grafiken mittels `:before` und `:after` keine Inhalte hinzufügen.

Inhalte einfügen: eigene Listenpunkte

```
! Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptates vero eius harum fuga expedita similique veniam earum nesciunt deleniti animi, exercitationem aspernatur dicta nostrum assumenda asperiores molestiae sequi ipsum, quae.

! Praesentium sapiente ex saepe facere nobis, qui sed autem voluptatibus unde. Quod nihil sunt illum ex facere, itaque ratione dolor ea in! Ducimus inventore doloribus architecto quibusdam tempore magni ipsam.

! Repellat ut possumus voluptas error, tenetur tempore, quasi a dolor suscipit harum laboriosam, dicta reprehenderit ipsum. Minima, at eligendi, possimus iusto aliquam consectetur suscipit veritatis iure, itaque, nobis aut iste.

! In maiores, placeat minima architecto harum delectus, atque, adipisci ratione corporis nihil rem suscipit sit voluptate doloremque! Amet eligendi, quibusdam recusandae odit eos rerum aut voluptas, distinctio deserunt ipsa delenit.

! Dolor quibusdam perspicatis molitia fuga nulla tempora reprehenderit in vero ipsa atque. Ex voluptates magni ducimus, fugiat repellendus quia neque, voluptatibus nostrum doloremque adipisci, odio minus, eos assumenda. Atque, fugit.
```

Eigene, sauber ausgerichtete Listenpunkte eingefügt per Pseudoelement

11.4 Zähler einbinden

In der Spezifikation von CSS sind Definitionen für Zähler vorgesehen, z. B. zur automatischen Nummerierung von Kapitelüberschriften. Gezählt wird dabei das Auftreten bestimmter Elemente innerhalb des Dokuments. Somit können Sie z. B. anhand der verwendeten `h1`- und `h2`-Überschriften eine automatische Nummerierung im Stil von 1, 1.1, 1.2, 2, 2.1, 2.2 usw. festlegen.

```
content: counter (Zählvariable, Listenart);
```

Werte	<ul style="list-style-type: none"> ✓ Beliebiger Name der Zählvariable ✓ Als Listenwert sind möglich: decimal, upper-alpha, lower-alpha, upper-roman, lower-roman, decimal-leading-zero
Standardwert	none / decimal
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Die Definition eines Zählers erfolgt über das Schlüsselwort `counter` innerhalb der CSS-Eigenschaft `content`. Mit diesem legen Sie die zu verwendende Zählvariable fest. Der Wert für die Listenart ist optional und legt die Darstellung des Zählerelements fest. Grundsätzlich sind als Angaben die Werte der CSS-Eigenschaft `list-style-type` möglich.

Mit der Eigenschaft `counter-increment` können Sie den Wert der Zählvariablen um einen beliebigen Betrag erhöhen. Sollte es notwendig sein, einen Zähler zurückzusetzen, steht Ihnen die Eigenschaft `counter-reset` zur Verfügung.

```
counter-increment: Zählvariable Wert;
counter-reset (Zählvariable);
```

Werte	<ul style="list-style-type: none"> ✓ Name der Zählvariable ✓ Der Wert ist ein Betrag, um den die Zählvariable erhöht werden soll. ✓ <code>none</code> = Die Zählvariable wird nicht erhöht oder zurückgesetzt, sondern nur angezeigt.
Standardwert	None
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Beispiel: *kap11\counter.htm*

Es wird eine Webseite mit Überschriften erster und zweiter Ordnung erstellt. Diese sollen automatisch durchnummieriert werden.

①	<pre><html lang="de"> <head> <link rel="stylesheet" href="..../standard.css"> <style type="text/css"> body {counter-reset: Ordnung01;} h1:before { content: counter(Ordnung01, upper-roman) " "; counter-increment: Ordnung01; } </style> </head> <body> <h1>Überschrift 1</h1> <h2>Überschrift 2</h2> </body> </html></pre>
---	--

```

②    h1 { counter-reset: Ordnung02; }
      h2:before {
        content: counter(Ordnung01) "." counter(Ordnung02,
          decimal-leading-zero) " ";
        counter-increment: Ordnung02;
      }
    </style>
    <title>Zähler einfügen: counter()</title>
</head>
<body>
  ③ <h1>Überschrift vom Typ h1</h1>
  ④ <h2>Untergeordnete Überschrift vom Typ h2</h2>
  <p>Fortune plango vulnera stilitibus ocellis, quod sua michi
  ...</p>
  <h2>Untergeordnete Überschrift vom Typ h2</h2>
  <p>Omnia Sol temperat purus et subtilis, novo mundo reserat faciem
  ...</p>

  ⑤ <h1>Weitere Überschrift vom Typ h1</h1>
  <h2>Untergeordnete Überschrift vom Typ h2</h2>
  <p>Qui autem auscultare nolet, exsurgat foras, ut sit, ubi sedat
  ...</p>
  <h2>Untergeordnete Überschrift vom Typ h2</h2>
</body>
</html>

```

- ① Zunächst wird die Nummerierung auf null gesetzt. Die Nummerierung der Überschrift erster Ordnung soll vor dem Text der h1 erscheinen, deshalb wird das Pseudoformat :before eingesetzt. Als Inhalt wird der Zähler h1 in großen römischen Ordnungszahlen definiert, gefolgt von einem Leerzeichen. Der Zähler Ordnung01 soll bei jedem Auftreten der Überschrift h1 um den Wert 1 erhöht werden.
- ② Bei jedem erneuten Auftreten einer Überschrift erster Ordnung soll der Zähler Ordnung02 zurückgesetzt werden, damit die Überschriften zweiter Ordnung in jedem Kapitel neu gezählt werden. Mit h2 :before definieren Sie einen Inhalt am Beginn jeder h2.

Dieser besteht aus dem Wert des Zählers Ordnung01, einem Punkt, dem Zählerwert Ordnung02 sowie einem Leerzeichen. Auch hier wird bei jeder Formatierung der Zähler um eins hochgezählt.

I Überschrift vom Typ h1

① ③

1.01 Untergeordnete Überschrift vom Typ h2

②

Fortune plango vulnera stilitibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronde capillata, sed plerumque sequitur occasio calvata. In Fortune solio sederam elatus, prosperitatis vario flore coconatus; quicquid enim florui felix et beatus, nunc a summo corruji gloria privatus. Fortune rota volvitur descendere minoratus, alter in altum tollitur, nimis exaltatus rex sedet in vertice; caveat ruinam! Nam sub axe legimus hecubam redinam.

1.02 Untergeordnete Überschrift vom Typ h2

② ④

Omnia Sol temperat purus et subtilis, novo mundo reserat faciem Aprilis; ad Amorem prosperat animus herilis et iocundis imperat deus puerilis. Rerum tanta novitas in solemni vere et veris auctoritas iubet nos gaudere, vias prebet solitas, et in tue vere fides est et probitas tuum retinere. Anna me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.

II Weitere Überschrift vom Typ h1

① ⑤

2.01 Untergeordnete Überschrift vom Typ h2

②

Qui autem auscultare nolet, exsurgat foras, ut sit, ubi sedat ille qui auscultare vult. Id nos Latine gloriosum dicimus. Quia adseditis causa in festivo loco, comoediā quam nos actuari sumus et argumentum et nomen vobis eloquar. Alazon Graece huic nomen est comoedias: Id nos Latine gloriosum dicimus. Flora veris leti facies mundo propinatur, hiemalis acies victa iam fugatur. In vestitu vario Flora principatur, nemorum dulcisonio que cantu celebratur. Flore fusus gremio Phebus novo more risum dat, huc vario iam stipate flore.

2.02 Untergeordnete Überschrift vom Typ h2

②

- ③ Die erste Überschrift leitet den Zählerstart ein und erhält als Inhalt den Wert 1.
- ④ Die zweite Überschrift erhält den Wert 1 und leitet den Zählerstart für den zweiten Inhalts- teil ein. Somit wird die Zeichenkette "1 . 01 " vor die Überschrift gesetzt. Dies setzt sich mit "1 . 02 " fort.
- ⑤ Mit der erneuten Auszeichnung einer h1 wird automatisch der Wert h2 zurückgesetzt, sodass der folgenden h2 die Zeichenkette 2 . 01 vorangestellt wird.

11.5 Rechnen in CSS

Mitunter kann es nützlich sein, anstelle einer bestimmten Größe eine Gleichung anzugeben. In CSS steht Ihnen dafür die Funktion calc() zur Verfügung.

```
foo { width: calc(2em - 3px + 2vw) ; }
```

- ✓ Berechnungen sind nicht auf Breitenangaben beschränkt. Auch Schriftgrößen, Farbwerte und vieles anderes lassen sich berechnen. Sie können für Rechnungen sogar Variablen einsetzen.

```
foo { width: calc((2em - 3px) * (2vw / 1.5) ; }
```

- ✓ Sie können nicht nur Einheiten mixen. Auch feste Werte wie 1,5 können Teil der Berechnung sein. Außerdem dürfen Sie Klammern verwenden, um komplexere Kalkulationen durchzuführen.

```
div { width: calc(2em - 3px + 2vw) ; }
```

- ✓ In der Rechnung werden em, Pixelangaben und Anteile der Viewport-Breite verwendet. Diese Berechnung lässt sich nur ausführen, wenn die an dieser Stelle verwendete Schriftgröße und die aktuelle Breite des Viewports bekannt sind. Der Browser rechnet alle Angaben für die Darstellung auf dem Bildschirm in Pixel um und kann die so ermittelten Pixelwerte miteinander verrechnen.

Beispiel: *kap11\calc.htm*

①	<pre><html lang="de"> <head> <style> html { display: flex; background: white; } </style> </head> <body></pre>
---	---

```

②    body {
        width: calc(3px - 1em + 80vw);
        height: calc(3px - 1em + 80vmin);
        margin: auto;
        background: silver;
    }
</style>
</head>

<body>
</body>
</html>

```

Quelltext der Datei „kap11/calc.htm“

- ① Mittels `display: flex` wird eine Layoutmethode gewählt, die eine horizontale und vertikale Zentrierung erlaubt (Näheres im folgenden Kapitel „Layout mit CSS erstellen“).
- ② Höhe und Breite des Dokumentenrumpfes werden unter Nutzung verschiedenster Einheiten berechnet. Der Dokumenten-Rumpf wird grau hinterlegt, sodass seine Ausdehnung sichtbar wird.

① und ②
Die graue Fläche entspricht den berechneten Abmessungen.

11.6 Custom Properties (CSS-Variablen)

Variablen werden schon seit vielen Jahren von Entwicklern gewünscht und haben den Erfolg von Präprozessoren mit begründet – dort stehen sie von Beginn an zur Verfügung. Wie seit HTML5 üblich, werden beliebteste Features, die sich nur mit externen Hilfsmitteln realisieren ließen (z. B. JavaScript oder LESS) nach und nach in die offiziellen Standards übernommen. Seit einer ganzen Weile gehören CSS-Variablen zum offiziellen CSS-Standard. Inzwischen werden sie von allen aktuellen Browsern unterstützt.

Vorteile nativer Variablen gegenüber der Verwendung von Variablen in Präprozessoren sind die Einhaltung der Kaskade und der Vererbung. Auch in der `calc()`-Funktion können Sie Variablen nutzen.

Definition

```
element1 {
  --name: wert;
}
```

Verwendung

```
element2 {
  eigenschaft: var(--wert);
}
```

- ✓ CSS-Variablen werden für ein bestimmtes Element definiert. Sie gelten nur für dieses Element und alle seine Nachfahren.
- ✓ Wenn Sie die Variable nutzen möchten, weisen Sie die Variable als Wert einer Eigenschaft zu. Dazu verwenden Sie `var()` und geben in den Klammern den Namen der Variablen an.

Beispiel: *kap11\variable.htm*

```

<!DOCTYPE html>
<html lang="de">
<head>
  <style>
    ① :root {
      --main-color: #f00;
      --main-background-color: #fff;
    }
    ② body {
      color: var(--main-color);
      background-color: var(--main-background-color);
    }
  </style>
</head>
<body>
  <h1>Überschrift</h1>
  <p>Lorem ipsum...</p>
</body>
</html>

```

Definition und Verwendung von CSS-Variablen im Beispiel „kap11/variable.htm“

- ① Die Variablen werden definiert, indem einem beliebigen Namen zwei Minus-Zeichen vorangestellt werden. Den eigentlichen Wert weisen Sie wie gewohnt nach einem Doppelpunkt zu. Die Definition einer Variablen ist wie jede andere Deklaration zu notieren und muss daher ebenfalls mit einem Semikolon abgeschlossen werden.
- ② Verwenden Sie die Variable, indem Sie als Wert einer Eigenschaft `var (- - name)` notieren.

 **Lernvideo:** CSS-Variablen anwenden

11.7 Mauszeiger anpassen

So wie sich im Betriebssystem bei bestimmten Aktionen der Mauszeiger ändert, können Sie dies auch auf Webseiten verwenden. Dafür steht Ihnen die CSS-Eigenschaft `cursor` mit verschiedenen Werten zur Verfügung.

cursor: Wert;

Im Folgenden sehen Sie eine Auflistung der laut Spezifikation erlaubten Werte:

Werte	<ul style="list-style-type: none"> ✓ auto = automatisch (vom System vorgegeben) ✓ default = Standard-Mauszeiger (plattformunabhängig) ✓ none = kein Mauszeiger sichtbar ✓ context-menu = kein Mauszeiger sichtbar ✓ help = Hilfesymbol ✓ pointer = Zeiger (meist ein ausgestreckter Zeigefinger) ✓ progress = Fortschrittssymbol (meist wie das Wartesymbol) ✓ wait = Wartesymbol (meist wie das Fortschrittssymbol) ✓ cell = Symbol für Tabellenzellen ✓ crosshair = Fadenkreuz ✓ text = Textcursor ✓ vertical-text = vertikaler Textcursor (Textcursor um 90° gedreht) ✓ alias = Verknüpfungssymbol ✓ copy = Kopiersymbol ✓ move = bewegbares Objekt ✓ no-drop = Symbol für ein Ablegen-Verbot ✓ not-allowed = Symbol für eine verbotene Aktion ✓ grab = zupackende Hand ✓ grabbing = zupackende Hand ✓ e-resize = Pfeil nach rechts (e = East) ✓ n-resize = Pfeil nach oben (n = North) ✓ ne-resize = Pfeil nach rechts oben (ne = Northeast) ✓ nw-resize = Pfeil nach links oben (nw = Northwest) ✓ s-resize = Pfeil nach unten (s = South) ✓ se-resize = Pfeil nach rechts unten (se = Southeast) ✓ sw-resize = Pfeil nach links unten (sw = Southwest) ✓ w-resize = Pfeil nach links (w = West) ✓ nesw-resize = Pfeil nach rechts oben und links unten ✓ nwse-resize = Pfeil nach links oben und rechts unten ✓ col-resize = Größenänderung einer Spalte ✓ row-resize = Größenänderung einer Zeile ✓ all-scroll = Scroll-Symbol ✓ zoom-in = Vergrößerungssymbol (z. B. Lupe mit Plus) ✓ zoom-out = Verkleinerungssymbol (z. B. Lupe mit Minus) ✓ url() = Angabe einer eigenen Cursorgrafik
Standardwert	Auto
Vererbbar	Nein
Anwendbar auf	Alle Elemente

Es können, analog zu Schriftangaben, mehrere Cursor angegeben werden. Wenn der Browser den ersten Cursor nicht darstellen kann, versucht er den nächsten usw.

Beispiel: *kap11\cursor.htm*

Verschiedenen Element-Blöcke weisen Sie unterschiedliche Cursordarstellungen zu. Sobald der Mauszeiger über einen dieser Blöcke bewegt wird, soll er sich verändern.

```
<html lang="de">
<head>
    <link rel="stylesheet" href=".../standard.css">
    <style type="text/css">
        ①   div {
            display: inline-block;
            width: 6.5em;
            margin: .5rem;
            border: 1px solid green;
            padding: .5rem;
            text-align: center;
            background: #eee;
        }
        ②   # auto      { cursor: auto; }
            # default  { cursor: default; }
            # none     { cursor: none; }

        [...]

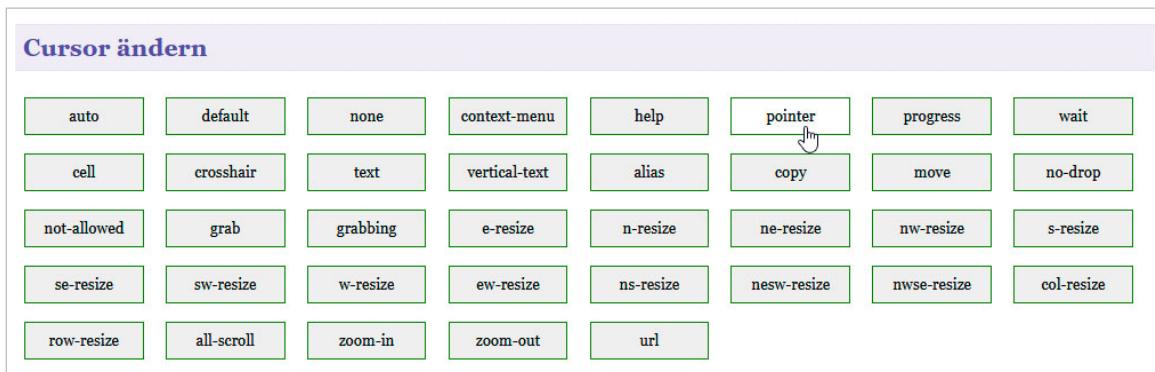
        ③   #url { cursor: url(images/lupe.cur), pointer; }
    </style>
    <title>Cursor ändern</title>
</head>
<body>
    ④   <h2>Cursor ändern</h2>
        <div id="auto">auto</div>
        <div id="default">default</div>
        <div id="none">none</div>

        [...]

        <div id="url">url</div>
</body>
</html>
```

- ① Die Elemente mit den unterschiedlichen Mauszeigern werden so formatiert, dass sie gut zu erkennen sind.
- ② Es folgt die Festlegung der einzelnen Mauszeiger. Dazu legen Sie verschiedene ID-Selektoren an, deren Formatierungen Sie später über die Angabe der ID den einzelnen Elementen zuweisen können.

- ③ Damit Sie einen individuellen Cursor einblenden können, binden Sie die externe Datei *lupe.cur* ein, die sich im Unterverzeichnis *images* befindet.
- ④ Im Rumpfbereich der Webseite legen Sie die verschiedenen Blöcke fest und weisen über die Angabe des Attributs *id* die entsprechenden Formatierungen zu.



Beim Überfahren der einzelnen Blöcke ändert sich der Mauszeiger.

11.8 Übungen

Übung 1: Absatzzähler ändern

Level		Zeit	ca. 10 min
Übungsdatei	<i>kap11\counter.htm</i>		
Ergebnisdatei	<i>kap11\uebung1.htm</i>		

1. Ändern Sie den Zähler im Beispiel der Datei *kap11\counter.htm* so, dass bei den Überschriften der zweiten Ordnung *h2* das Wort Kapitel und der zweite Inhaltsteil als Buchstabe angezeigt wird, zum Beispiel Kapitel 1.a, Kapitel 1.b, Kapitel 1.c usw.

Kapitel 1 Überschrift vom Typ h1

Kapitel 1.a: Untergeordnete Überschrift vom Typ h2

Fortune plango vulnera stulantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronde capillata, sed plerumque sequitur occasio calvata. In Fortune solio sedem elatus, prosperitas vario flore coconatus; quicquid enim florui felix et beatus, nunc a summo corrui gloria privatus. Fortune rota volvitur descendendo minoratus, alter in altum tollitur, nimis exaltatus rex sedet in vertice; caveat ruinam! Nam sub axe legimus hecubam redinam.

Kapitel 1.b: Untergeordnete Überschrift vom Typ h2

Omnia Sol temperat purus et subtilis, novo mundo reserat faciem Aprilis; ad Amorem prosperat animus herilis et iocundis imperat deus puerilis. Rerum tanta novitas in solemnii vere et veris auctoritas iubet nos gaudere, vias prebet solitas, et in tue vere fides est et probitas tuum retinere. Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.

Geänderte Zählung der Kapitelüberschriften

Übung 2: Mauszeiger testen

Level		Zeit	ca. 10 min
Übungsdatei	Kap11/cursor.htm		

Die CSS-Spezifikation sieht 36 Schlüsselwörter verschiedene Cursor vor. Außerdem haben Sie die Möglichkeit, beliebige eigene Grafiken einsetzen zu können.

Nicht alle Browser zeigen zu jedem Schlüsselwort eine Grafik an.

1. Öffnen Sie die Datei *cursor.htm* mit verschiedenen Browsern, wenn möglich auch auf unterschiedlichen Betriebssystemen (Mac, Windows, Linux), und überprüfen Sie, ob und welche Grafiken beim Überfahren der Elemente angezeigt werden.

Übung 3: Mauszeiger manipulieren

Level		Zeit	ca. 20 min
Ergebnisdatei	kap11\uebung3.htm		

1. Erstellen Sie eine Webseite mit mehreren Absätzen. Beim Überfahren der Absätze soll der Mauszeiger als Fadenkreuz erscheinen. Der zweite, umrandete Absatz soll als einziger Absatz ein Zeiger-Symbol `pointer` erhalten.

Mit unterschiedlichen Mauszeigern arbeiten

Fortune plango vulnera stilantibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronte capillata, sed plerumque sequitur occasio calvata. In Fortune solio sederam elatus, prosperitatis vario flore coconatus.

Merke:

Quicquid enim florui felix et beatus, nunc a summo corru gloria privatus. Fortune rota volvitur descendit minoratus, alter in altum tollitur, nimis exaltatus rex sedet in vertice; caveat ruinam! Nam sub axe legimus hecubam redinam.

Zephyrus nectareo spirans in odore, certiam pro bravia curramus in amore. Cytherizat ~~h~~^tico dulcis Philomena, flore rident vario prata iam serena, salit cetus avium silve per amena, ~~n~~^m gaudia millena.

Omnia Sol temperat purus et subtilis, novo mundo reserat faciem Aprilis; ad Amorem prosperat animus herilis et iocundis imperat deus puerilis. Rerum tanta novitas in solemni vere et veris auctoritas iubet nos gaudere, vias prebet solitas, et in tue vere fides est et probitas tuum retinere. Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.

Geänderte Mauszeigerdarstellung

12

Layout mit CSS erstellen

In diesem Kapitel werden Ihnen verschiedene Layouts nähergebracht und Schritt für Schritt erklärt. Die vorgestellten Layoutmethoden sind **responsiv**, das heißt, sie funktionieren sowohl auf großen Bildschirmen als auch auf Smartphones und Tablets.

12.1 Zweispalter mit Kopfbereich

Das „mobile first“-Prinzip

Webseiten werden in Deutschland mittlerweile ebenso häufig – weltweit sogar häufiger – auf mobilen Geräten (Smartphone und Tablet) betrachtet wie auf großen Monitoren. Tendenziell verlieren die großen Monitore weiter an Bedeutung. **Obwohl Sie Ihre Webseite wahrscheinlich an einem großen Monitor erstellen, sollten Sie nie vergessen, dass das nicht mehr die Standard-Darstellung für sehr viele Ihrer Besucher ist.**

Bevor Sie mit der Erstellung eines Layouts beginnen, hat Ihre Webseite schon eine Darstellung, die auf jedem Gerät funktioniert (schwarzer Text auf weißem Grund, der am Browserrand umbricht).

Fügen Sie erst nur eigene Farben und Schriften hinzu. Alle Elemente stehen weiter untereinander und lassen sich durch Scrollen erreichen.



*Standard-Darstellung plus eigene Farben
(kap12\2spalte.htm)*

Beispiel: HTML-Aufbau für ein Zweispalten-Layout

```
① <!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <title>Layout: Zweispalter mit Titelzeile</title>
    <link rel="stylesheet" href="twoColumns.css">
  </head>

  <body>
```

```

② <h1>
    <a href="index.htm">
        
    </a>
    <span>Überschrift</span>
</h1>

③ <div id="container">
    <nav>
        <ul>
            <li><a href="#">Link</a></li>
            <li><a href="#">Link</a></li>
            <li><a href="#">Link</a></li>
        </ul>
    </nav>

    <main>
        <h2>Überschrift des Artikels</h2>
        <p>Fortune ...</p>
        <p>Omnia ...</p>
    </main>
</div>
</body>
</html>

```

Struktur der Datei „kap12/2spalter.htm“

- ① Am Anfang der Datei finden Sie die gewohnten Angaben wie Dokumenten-Typ, Sprachangabe, Zeichenkodierung, Titel und natürlich die Verknüpfung mit einer CSS-Datei (siehe nächstes Listing).
- ② Die Überschrift enthält das Logo und einen Text in einem eigenen `span`-Element. So können diese getrennt voneinander formatiert und ausgerichtet werden.
- ③ Ein Container-Element gruppiert Navigation und Hauptbereich der Seite, sodass diese in diesem Container ausgerichtet werden können.
- ④ Navigation und Hauptbereich stehen zunächst untereinander. Dies ist das natürliche Verhalten von Blockelementen.

Beispiel: Mittels CSS Farben für die Beispielwebseite in der CSS-Datei angeben

```

body {
    font-family: Calibri, Helvetica, Arial, Geneva, sans-serif;
    background-color: #ddd;
    margin: 1em auto;
}

h1>*, #container>* {
    padding: 0.5em;
}

h1>a {
    display: inline-block;
    background-color: #4e95dc;
}

```

```

h1>span {
    display: block;
    background-color: #eee;
}
nav {
    background-color: #5fa6ed;
}
main {
    background-color: #fff;
}

```

- ✓ In der CSS-Datei werden zunächst nur Farben notiert. Das Ergebnis ist schon recht ansehnlich, obwohl Sie auf einer echten Webseite eher mit weniger unterschiedlichen Hintergründen arbeiten würden. Diese verursachen, dass das Beispiel sehr kleinteilig und zerstückelt wirkt. Es hilft hier, die einzelnen Bereiche zu unterscheiden, und soll im Beispiel daher beibehalten werden.
- ✓ Die Angabe von margin: 1em auto; bewirkt, dass oben und unten ein Rand von 1em frei gelassen wird, rechts und links soll automatisch gleich viel Rand zugewiesen werden. Da sich Blockelemente über die gesamte Seitenbreite erstrecken, wirkt sich diese Angabe nur aus, wenn Sie eine Breitenangabe notieren und der Bildschirm breiter ist als das Element. Diese Angabe spielt nur auf größeren Bildschirmen eine Rolle und wird im nächsten Abschnitt wichtig.

Mehrspaltiges Layout mit `display:table`

Nach den Farben legen Sie nun die Aufteilung der Seite in der Datei `twoColumns.css` fest.

```

① @media screen and (min-width:40em) {
②   body {
        display: table;
        width: 40em;
    }
③   h1, #container {
        display:table-row;
    }
④   #container>* {
        display: table-cell;
    }
⑤   h1>a,
    h1>span {
        display:table-cell;
        vertical-align: middle;
    }
    nav {
        vertical-align: top;
    }
    article {
        vertical-align: top;
    }
⑥ }

```

Schnell erledigt: Dank der Tabellendarstellung reichen für das Layout wenige Zeilen CSS.

- ① Mit der in Kapitel 1 erläuterten @media-Regel weisen Sie den Browser an, die folgenden Regeln nur umzusetzen, wenn die genannten Bedingungen erfüllt sind. Alles innerhalb der geschweiften Klammern soll also nur auf Bildschirmen (screen) mit einer Mindestbreite von 40em angewendet werden. Dadurch stellen Sie sicher, dass das Layout auf kleinen Bildschirmen (z. B. Smartphones) einspaltig bleibt – zwei Spalten haben hier nebeneinander nicht genug Platz.

Logo	Überschrift
<ul style="list-style-type: none"> • Link • Link • Link 	<p>Überschrift des Artikels</p> <p>Fortune plango vulnera stolidibus ocellis, quod sua michi munera subtrahit rebellis. Verum est, quod legitur, fronde capillata, sed plerumque sequitur occasio calvata. In Fortune sollio sederam elatus, prosperitatis vario flore coconatus; quicquid enim florui felix et beatus, nunc a summo corru gloria privatus. Fortune rota volvitur descendere minoratus, alter in altum tollitur, nimis exaltatus rex sedet in vertice; caveat ruinam! Nam sub axe legimus hecubam redinam.</p> <p>Omnia Sol temperat purus et subtilis, novo mundo reserat faciem Aprilis; ad Amorem prosperat animus herilis et iocundis imperat deus puerilis. Rerum tanta novitas in solemni vere et veris auctoritas iubet nos gaudere, vias prebet solitas, et in tue vere fides est et probitas tuum retinere. Ama me fideliter! Fidem meam nota: de corde totaliter et ex mente tota sum presentialiter absens in remota.</p>

Ein paar Zeilen CSS sorgen für nebeneinanderliegende Spalten.

- ② Um zwei Spalten bilden zu können, wird der Browser angewiesen, die Seite als Tabelle darzustellen. Die Breite wird auf 40em festgelegt. Bereits vorher wurde mit margin: 1em auto; festgelegt, dass weiterer Platz links und rechts gleichmäßig verteilt werden soll, wodurch die Inhalte zentriert werden.
- ③ Nun bestimmen Sie, welche Elemente wie Tabellenzeilen dargestellt werden sollen. Im vorliegenden Fall bietet sich die Überschrift h1 für die erste Zeile an, für die zweite Zeile benutzen Sie das Container-div.
- ④ In der ersten Zeile (h1) befinden sich zwei Elemente (span und a), die als Zellen fungieren können. So haben Sie eine Tabellenzeile, die zwei Zellen enthält. Im nachfolgenden Container gibt es wiederum zwei Kindelemente: nav und main. Weisen Sie auch diesen die Eigenschaft display:table-cell zu, um in dieser zweiten Zeile ebenfalls zwei Zellen zu schaffen. Damit haben Sie ein zweispaltiges Layout, das in allen aktuellen Browsern funktioniert.
- ⑤ Da die Zellen im Beispiel alle eindeutig voneinander unterscheidbar sind (nav, main, h1>a, h1>span), können Sie für jede eigene Styles definieren: Schriftarten, Farben, Rahmen usw. lassen sich für jeden Bereich einzeln vergeben. Elemente darin können Sie mittels Kind- oder Nachfahren-Selektor gezielt auswählen und gestalten. **Zusätzliche Klassen sind hier nicht nötig.**
- ⑥ Denken Sie daran, die Klammer der @media-Regel wieder zu schließen!

Mit wenigen Zeilen CSS konnten Sie ein mehrspaltiges Layout umsetzen. Das so entstandene Beispiel lässt sich um beliebig viele Spalten erweitern. Fügen Sie dafür einfach pro Zeile ein oder mehrere Kindelemente hinzu. Aufgrund der Selektoren h1>* bzw. #container>* erhalten diese automatisch die Eigenschaft display:table-cell. **Sie sollten darauf achten, dass alle Zeilen gleich viele Zellen enthalten.**

Den gesamten Code finden Sie in der Beispiel-Datei *kap12/twoColumns.css*.

12.2 Flexbox – flexible, responsive Layouts ohne @media-Regel

Mit Flexbox steht eine Methode bereit, mit der Sie responsive Layouts ohne @media-Regel umsetzen können. Der größte Unterschied einer Webseite zu Dokumenten aus Papier ist, dass Webseiten keine vorhersehbaren Dimensionen besitzen. Selbst auf großen Bildschirmen kann ein Fenster klein sein. Oder jemand, der schlecht sehen kann, vergrößert Schriften so stark, dass auch in einem breiten Fenster nur wenige Inhalte nebeneinander passen.

Zwar berücksichtigt das Tabellenlayout schon einige dieser Aspekte, doch musste mit der @media-Regel ein fester Breakpoint bestimmt werden.

Diesen zu setzen, erfordert Erfahrung und trotzdem geht es nicht immer ohne viele Tests, bis sich eine Webseite wie gewünscht verhält.

Flexboxen spielen hier ihre Stärke aus: Kombiniert mit einer Basisangabe für die Breite wird „überflüssiger“ Platz vom Browser gleichmäßig auf alle Elemente einer Zeile aufgeteilt.

Passen ein oder mehrere Elemente nicht in eine Zeile, werden diese umgebrochen und auch in dieser neuen Zeile wird überflüssiger Platz wieder gleichmäßig verteilt.

Dieses Verhalten können Sie über unterschiedlichste Angaben fein steuern, sodass immer ein ordentliches Aussehen mit bündigen Kanten erreicht wird.

Zu den vielen Vorteilen gehört auch, dass Sie sich bei Flexboxen nicht darum kümmern müssen, wie viele „Zellen“ in einer Zeile stehen.

Um das zweispaltige Layout mit Flexbox umsetzen zu können, benötigen Sie keine andere HTML-Datei, auch die Farb- und Schriftangaben aus dem ersten Beispiel können erhalten bleiben.

Nur die Layoutangaben lauten anders:

```

① body {
    max-width: 40em;
    margin: 1em auto;
}

② h1, #container {
    display: flex;
    flex-wrap: wrap;
    margin: 0;
}

```



Flexbox – Darstellung auf einem Smartphone-Bildschirm

```

③ h1 > *,
#container > * {
  flex: 2 1 22rem;
}

④ h1 > :first-child,
#container > :first-child {
  flex: 1 2 11rem;
}

```

Die Layoutangaben aus der Datei „kap12/twoColumns-flex.css“

- ① Wie zuvor wird das Layout auf 40em Breite beschränkt, diesmal aber mittels max-width, was ein Schrumpfen des verfügbaren Platzes zulässt.
- ② Die Container, die zuvor die Spalten beinhaltet haben, werden angewiesen, sich wie Flexboxen zu verhalten. flexwrap: wrap erlaubt ein Umbrechen der nebeneinander-liegenden Inhalte und margin:0 sorgt dafür, dass der Außenabstand der Überschrift verschwindet und keine Lücke zwischen h1 und div#container entsteht.
- ③ Den Inhalten der Flexboxen geben Sie mit flex drei Werte mit. Diese stehen für **Wachsen**, **Schrumpfen** und **Basisgröße** (meist Breite). Die Angabe für Wachsen und Schrumpfen geben das Verhältnis zu benachbarten Boxen an. Die Basisgröße wird auf 22em gesetzt.
- ④ Abschließend erhält die erste „Spalte“ eine Basisbreite von 11rem, also die Hälfte der folgenden Spalten. Bei zwei Spalten ergibt sich dadurch eine Aufteilung von 1 zu 2. Steht genügend Platz zur Verfügung, sollen die Spalten auch im Verhältnis 1 zu 2 wachsen (jeweils die erste Angabe der Eigenschaft flex).

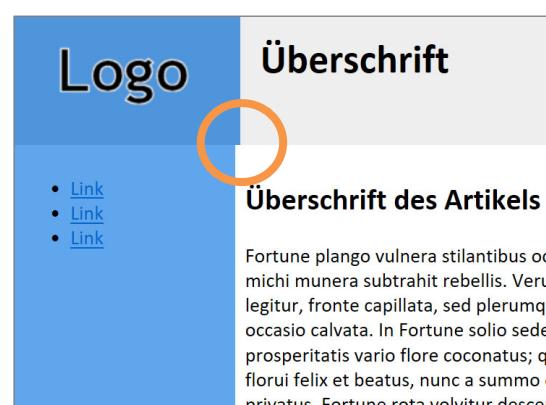
Das Verhältnis wird beibehalten, weil die Angaben zum Wachsen und Schrumpfen immer das **Verhältnis** angegeben, in dem freier Platz auf die Boxen verteilt wird. Daher werden keine Einheiten benötigt. Um die komplette Berechnung brauchen Sie sich nicht zu kümmern, das übernimmt der Browser und die Seite sieht immer ordentlich aus.

Wird das Fenster kleiner als 33rem (11rem + 22rem), werden die Boxen untereinander dargestellt (siehe Smartphone-Darstellung am Anfang dieses Abschnitts).

Ein Problem gibt es noch mit dem Logo. Dessen feste Breite sorgt bei Bildschirmen, die gerade noch groß genug für zwei Spalten sind, für eine unschöne Ecke (siehe Kreis im Bildschirmfoto).

Das können Sie lösen, indem Sie dem Bild eine Breite von maximal 100 % mitgeben:

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



Den kompletten Quelltext finden Sie in der Datei *kap12/twoColumns-flex.css*.

12.3 Eine responsive Navigation mit Flexbox umsetzen

Wie beim Flexbox-Layout gesehen, können sich Flexboxen untereinander oder nebeneinander anordnen, je nachdem, ob der Platz dafür ausreicht. Eine interessante Option für eine responsive Navigation.

Folgender Code sorgt zunächst dafür, dass die Navigationsliste wie ein Menü aussieht:

```

① nav ul {
    padding: 0;                  /* Listeneinrueckung */
    list-style-type: none;        /* Listenpunkte entfernen */
}
② nav li {
    margin: 0.125rem;           /* Abstände zwischen den li */
}
③ nav a {
    display: block;             /* Links füllen Elternelement aus */
    padding: 0.5rem;            /* Abstände um die Links */
}
④ nav a {
    background: white;          /* Hintergrund der Links */
}

```

Die Navigationsliste als Menü gestalten

- ① Die listentypische Darstellung wird entfernt.
- ② Zwischen den Listeneinträgen wird etwas Abstand hinzugefügt.
- ③ Die Links bekommen eine größere Fläche: Als Blockelemente füllen sie nun das Elternelement in voller Breite aus.
- ④ Um die klickbare Fläche zu kennzeichnen, bekommen die Links einen weißen Hintergrund.

Mit einem ähnlichen Code wie bei der Seitenaufteilung können wir die Menüpunkte veranlassen, nebeneinander oder untereinander zu stehen.



Die Liste sieht nun aus wie ein typisches Navigationsmenü.

In unserem Layout hat die Navigation eine Breite von `11rem` (abzüglich von Abständen). Wenn jeder Menüpunkt diese Breite bekommt, stehen alle untereinander. Da Flexboxen automatisch auf die gesamte Breite gezogen werden (überflüssiger Platz wird an alle Elemente komplett vergeben), kann auch eine geringere Breite angegeben werden. Wenn Sie sicherstellen wollen, dass niemals zwei Elemente nebeneinanderstehen, achten Sie lediglich darauf, dass zwei Elemente zusammen nicht weniger breit sind als das Elternelement. Andererseits sollten die Elemente breit genug für ihren Inhalt sein. Eine Breite von `9rem` reicht im vorliegenden Fall aus.

Der obige Code muss nur um drei Zeilen CSS ergänzt werden:

```

① nav ul {
    display: flex;          /* Flexbox-Container wird definiert */
    flex-wrap: wrap;        /* Umbrechen der Flexboxen erlauben */
    padding: 0;              /* Listeneinrueckung */
    list-style-type: none;   /* Listenpunkte entfernen */
}

② nav li {
    flex: 1 0 9rem;         /* grow shrink base */
    margin: 0.125rem;       /* Abstaende zwischen den li */
}

③ nav a {
    display: block;
    padding: 0.5rem;
    background: white;
}

```

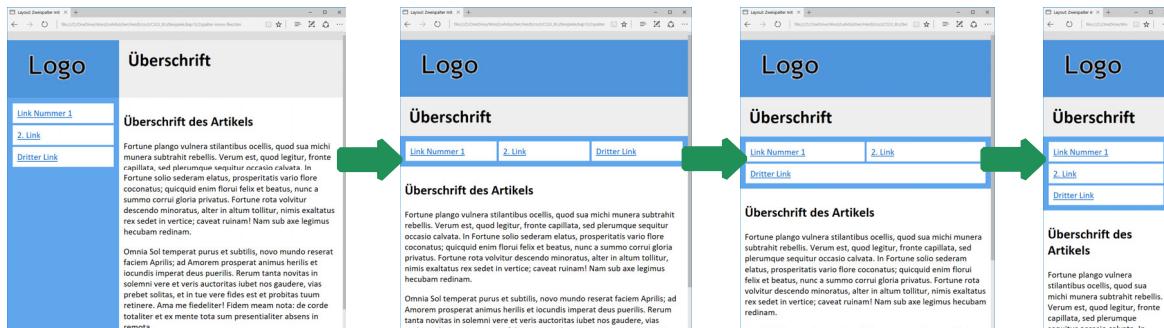
Nur drei Zeilen CSS sorgen für eine flexible Darstellung der Navigation.

- ① Weisen Sie das Listenelement `ul` an, als Flexbox-Container zu fungieren. Damit die Listeneinträge `li` bei wenig Platz untereinander stehen können, erlauben Sie das Umbrechen mittels `flex-wrap: wrap`.
- ② Mit `flex: 1 0 9rem` weisen Sie den Browser an, dass die Listeneinträge wachsen dürfen (1), aber nicht schrumpfen dürfen (0 – damit sie nicht kleiner werden als der enthaltene Text). Die Basisgröße geben Sie mit `9rem` an. Das ist breit genug für die Menüeinträge in diesem Beispiel – in Ihrer eigenen Seite könnte ein anderer Wert notwendig sein oder besser aussehen.
- ③ Der Rest bleibt wie gehabt.

Der entscheidende Unterschied wird erst bei geänderter Bildschirmbreite sichtbar. Auf großen Bildschirmen sieht die Navigation nämlich wie bisher aus, da in die schmale linke Spalte nach wie vor keine zwei Links nebeneinander passen.

Auf schmaleren Bildschirmen werden Kopfbereich, Navigation und Inhaltsbereich aber untereinander dargestellt. Jetzt haben die Menüpunkte genügend Platz, um nebeneinanderzustehen.

Auf sehr kleinen Bildschirmen (z. B. Smartphone) reicht der Platz dafür nicht mehr und die Listeneinträge stehen wieder untereinander.



Unterschiedliche Darstellung auf verschiedenen Displays

12.4 Benutzerfreundliche Menüs erstellen

Die Gestaltung der Navigationsbereiche ist von großer Bedeutung für die Nutzbarkeit Ihrer Webseite. Eine gute Navigation führt Ihre Besucher zielsicher auf kurzem Weg zu gesuchten Inhalten. Eine undurchdachte Navigation kann den Besucher verwirren. Daher sollten Sie Ihre Navigation nachvollziehbar gestalten: Einerseits sollte die Navigation auffallen und sofort erkennbar sein und sich andererseits optisch harmonisch in das Design Ihrer Seite einfügen.

Obwohl die Gestaltung einer Webseite in vielen Punkten von persönlichen Interessen und Vorlieben abhängt, gibt es einige Punkte, die gerade unerfahrenen Entwicklern helfen können, den Besucher nicht vor Rätsel zu stellen.

- ✓ **Halten Sie Menüs kurz** – pro Liste sollten nie mehr als sieben Einträge enthalten sein
- ✓ **Vermeiden Sie Untermenüs**. Wenn das nicht möglich ist, stellen Sie nicht mehr als fünf Menüpunkte pro Ebene dar. Eine zweite oder gar dritte Hierarchie-Ebene muss sich nicht in der Hauptnavigation befinden. Sie können dafür einen weiteren Navigationsblock zum Beispiel in der linken Seitenleiste anbieten.
- ✓ **Machen Sie es wie die anderen**. Eine Navigation befindet sich meistens direkt über oder unter dem Kopfbereich. Da wird sie gefunden und als solche erkannt.
- ✓ **Erfüllen Sie die Erwartungen**. Auf einer Webseite möchte man in aller Regel etwas erledigen: eine E-Mail schreiben, ein Formular ausfüllen, etwas herunterladen oder eine bestimmte Information finden. Dafür sind klare, deutliche Beschriftungen wichtig. Wenn man zum Beispiel auf einen Link „Kontakt“ klickt, möchte man gesammelt möglichst alle Kontaktmöglichkeiten auf einen Blick - von der Anfahrtsskizze über eine E-Mail-Adresse und Telefonnummer bis zur Postanschrift und tatsächlichen Adresse. Nur ein Kontaktformular ist zu wenig.
- ✓ **Unterstützen Sie den Nutzer**. Sicher bieten Sie auch Navigationsmöglichkeiten außerhalb des Menüs an. Klickbare Bereiche auf Ihrer Seite sollten sich als solche ausweisen. Von anderen Seiten sind Ihre Besucher es gewöhnt, wenn beim Überfahren von Schaltflächen etwas passiert: Farben ändern sich, Bildchen werden eingeblendet, ein anderer Mauszeiger wird verwendet. Links werden sofort als solche erkannt, wenn sie unterstrichen sind. Wenn Sie die Unterstreichung aus ästhetischen Gründen entfernen, sollten Sie unbedingt eine ähnlich **intuitive und deutliche Kennzeichnung** hinzufügen.

Besonders mit der ersten Empfehlung „Halten Sie Menüs kurz“ tut sich mancher zu Recht schwer. Soll eine Navigation mit nur wenigen Einträgen einen Nutzer sinnvoll führen – womöglich unter Verzicht einer weiteren Menüebene – muss eine Webseite gut durchdacht sein.

Ein Ansatz ist es, Beiträge zu verschlagworten und Übersichten aller Artikel zu bestimmten Schlagworten an **passenden** Stellen anzubieten (z. B. über, unter oder neben verwandten Artikeln).

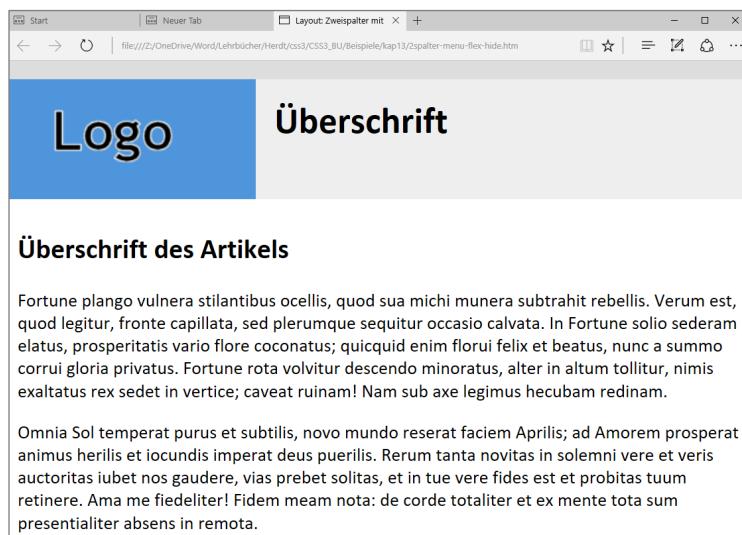
Trotzdem können in der Hauptnavigation schnell mehr Einträge sein, als auf einem Smartphone gut wäre: Schließlich sollen Seitenkopf (Logo und Name des Angebotes) plus Menü nicht das Einzige sein, was ein Besucher zu sehen bekommt.

Als kompakte Darstellung für Suche und Menü hat es sich durchgesetzt, diese zu verbergen und erst beim Antippen eines Buttons anzuzeigen. Da sich diese Methode weit verbreitet hat, kann sie verwendet werden. Sie erfüllt die anderen oben genannten Anforderungen an benutzerfreundliche Menüs.

Die Umsetzung lässt sich mit reinem CSS realisieren.

Hierfür benötigen Sie tatsächlich eine Medien-Abfrage, da Sie das Menü nur auf besonders kleinen Bildschirmen verbergen wollen und nur dann, wenn es sich wirklich nicht vermeiden lässt.

Zunächst wird das Menü ausgeblendet. Dazu gibt es mehrere Methoden. Die hier vorgestellte Methode lässt das Menü nur optisch verschwinden, sodass es für Software (z. B. Vorleseprogramme für Blinde) weiterhin zugänglich ist.



CSS

```
nav ul {
    nav {
        border: 0; /* Rahmen entfernen */
        clip: rect(0 0 0 0); /* Größe beschneiden auf 0 */
        height: 1px; /* Größe begrenzen */
        margin: -1px; /* Um 1px aus dem sichtbaren Bereich verschieben */
        overflow: hidden; /* Größere Inhalte verbergen */
        padding: 0; /* Innenabstand entfernen */
        position: absolute; /* Oben links positionieren */
        width: 1px; /* Größe begrenzen */
    }
}
```

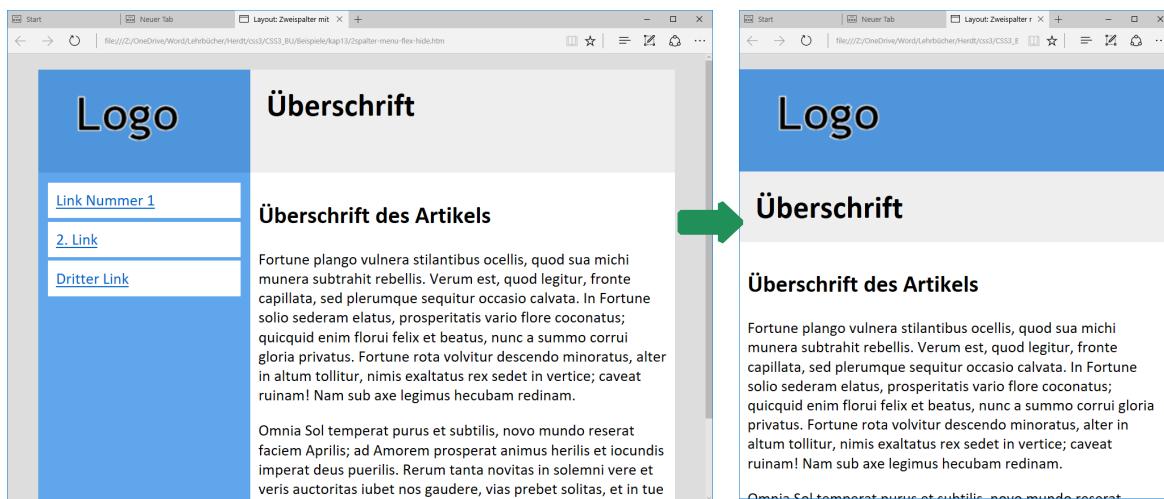
- ✓ Das Menü ist nicht mehr zu sehen, da es aus dem sichtbaren Bereich verschoben wurde. Software, wie Vorleseprogramme, kümmern sich nicht um die Position und geben es weiterhin aus.

Als Nächstes stellen Sie sicher, dass das Menü auf großen Bildschirmen nicht ausgeblendet wird. Dazu nutzen Sie eine Medienabfrage:

CSS

```
@media screen and (max-width: 30rem) {
    nav ul {
        [...]
    }
}
```

- ✓ Die gemachten Angaben (die zum Verbergen des Menüs führen) sollen nur für Bildschirme gelten, die maximal `30rem` breit sind. Das entspricht bei normaler Schriftgröße 480 Pixeln. Sowohl auf Tablets als auch auf Desktop-Monitoren wird das Menü angezeigt – wenn die Fenster breit genug sind.



Da das Menü für Smartphone-Nutzer nun gar nicht mehr erreichbar ist, muss eine weitere, vom Benutzer beeinflussbare Bedingung hinzugefügt werden, um das Menü gezielt einblenden und ausblenden zu können.

Eine Technik, die Checkbox-Hack genannt wird, bietet sich hierfür an. Sie macht geschickten Gebrauch von CSS-Selektoren:

CSS

```
:checked ~ ul
```

- ✓ Hier werden zwei Selektoren miteinander kombiniert. Die Pseudoklasse `:checked` fragt ab, ob eine Checkbox oder ein Radiobutton ausgewählt ist. Mit `~ ul` ist eine Liste gemeint, die der Checkbox folgt (also ein später im Quelltext auftauchendes Geschwister). Verwenden Sie dies statt nur `ul`, so wird die Navigation wieder auf Smartphones angezeigt, da sich vor der Liste keine (ausgewählte) Checkbox befindet.

Diese fügen Sie im nächsten Schritt zur HTML-Datei hinzu:

```
<nav>
  <input type="checkbox">
  <ul>
    <li><a href="#">Link Nummer 1</a></li>
    [...]
```

- ✓ Eine Checkbox vor der Navigation erlaubt das gezielte Ein- und Ausschalten des Menüs.

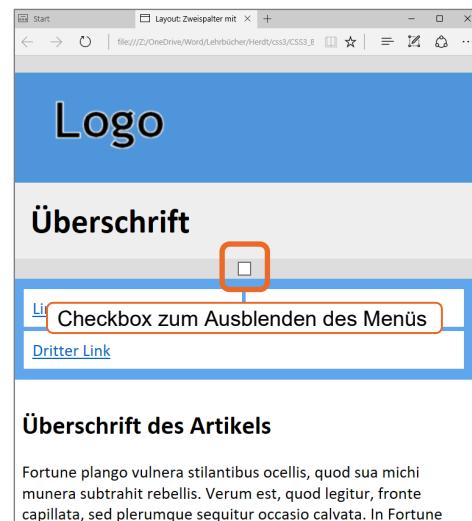
Jetzt können Sie durch Setzen des Häkchens die Navigation gezielt anzeigen oder verbergen. Da Platz gespart werden soll, verbergen Sie die Navigation standardmäßig, indem Sie der Checkbox das Attribut `checked` mitgeben:

HTML

```
<input type="checkbox" checked>
```

Jetzt muss der Nutzer eingreifen, um das Menü anzeigen zu lassen. Es bleibt aber ein Problem: Er weiß nicht, wozu das Kästchen dient. Es wird Zeit, es zu beschriften. Hierzu dient das Wort „Menü“, umgeben von einem Rahmen.

Das lässt sich erreichen, indem Sie ein ohnehin noch fehlendes Element in das HTML-Dokument einfügen: eine Beschriftung für die Checkbox.



HTML

```
<input type="checkbox" id="menu-switch">
<label for="menu-switch">Menü</label>
```

- ✓ Beachten Sie, dass Sie der Checkbox eine ID hinzufügen müssen, um die Beschriftung dieser Checkbox mittels `for`-Attribut zuweisen zu können

Das bringt Sie in eine komfortable Lage, denn ein Klick auf die Beschriftung funktioniert genau wie ein Klick auf die Checkbox selbst. Das heißt, Sie müssen die Checkbox nicht mehr anzeigen lassen. Sie funktioniert trotzdem:

CSS

```
nav [type="checkbox"] {
    opacity: 0;           /* checkbox wird transparent und */
    position: absolute;   /* aus dem Elementfluss genommen */
}
```

- ✓ Die Checkbox wird komplett durchsichtig und dank `position: absolute` beansprucht sie auch keinen Platz mehr für sich.

Nun können Sie sich der Gestaltung des Schriftzuges „Menü“ widmen. Das wiederum geschieht in der CSS-Datei:

CSS

```
label {
    border: 1px solid currentColor; /* Rahmen in Schriftfarbe */
    padding: 0 1em;                 /* Innenabstand */
    color: #0004E8;                /* Dunkelblaue Schriftfarbe */
    background: #eee;               /* Hellgrauer Hintergrund */
    cursor: pointer;               /* Mauszeiger als Hand */
}
```

Nun funktioniert das Menü und es ist verständlich, wie es aufgeklappt werden kann. Ein paar Dinge sind dennoch zu tun. Um Benutzern klarzumachen, dass das Wort „Menü“ klickbar ist, sollte der Button beim Überfahren mit der Maus und beim „Antabben“ sein Aussehen ändern, z. B. Vordergrund- und Hintergrundfarbe tauschen:

CSS

```
label:hover {  
    color: #eee;  
    background: #0004E8;  
}
```

Wie eingangs erwähnt steht das Menü blinden Menschen, die ein Vorleseprogramm nutzen, immer zur Verfügung. Sie benötigen den Menü-Button also nicht. Ergänzen Sie daher im öffnenden `label`-Tag das Attribut `aria-hidden`:

HTML

```
<label for="menu-switch" aria-hidden="true"Menü</label>
```

Den vollständigen Code (CSS und HTML) finden Sie im Beispielordner in den Dateien `kap12/twoColumns-menu-flex-hide.css` und `kap12/2spalter-menu-flex-hide.htm`



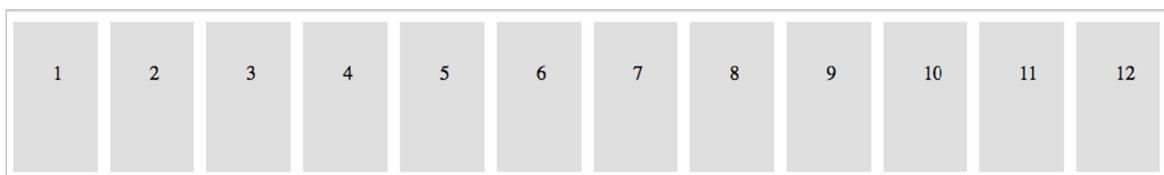
Lernvideo: *Flexibles einblendbares Menü*

12.5 CSS-Grid

CSS-Grid ist die erste Technik, die explizit für das Layouten von Webseiten erdacht wurde. Wann immer möglich, sollten Sie diese für das Layout Ihrer Website verwenden.

Mit CSS-Grid haben Sie die Möglichkeit, mit einer einzigen Zeile CSS ein komplettes Grid (Raster) zu erstellen.

In der Webseitengestaltung versteht man unter einem Raster die Aufteilung der Seite in unsichtbare „Spalten“, an denen man die verschiedenen Elemente ausrichtet, damit diese bündig zueinander sind.



Ein zwölfspaltiges Raster mit sichtbaren Spalten ([/kap12/2spalter-grid/2spalter-grid-visibleGrid.htm](#))

Dabei können Sie die Breiten in den bekannten Einheiten wie beispielsweise `vw`, `em` oder `%` angeben. Mit Einführung von CSS-Grid ist eine neue Einheit „Bruchteil“ (engl.: fraction) hinzugekommen. `1fr` verteilt den verfügbaren Platz gleichmäßig auf alle definierten Spalten.

Zahlen vor dieser Einheit stehen also für das Verhältnis der Spaltenbreiten zueinander. In aller Regel möchte man gleich breite Spalten. Um das zu erreichen, geben Sie einfach für alle zwölf Spalten jeweils `1fr` an.

Um ein zwölfspaltiges Grid anzulegen, reichen zwei Zeilen CSS:

```
body {
    ① display: grid;
    ② grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr
                            1fr 1fr 1fr;
}
```

Definition eines zwölfspaltigen Rasters

- ① `display: grid` macht aus `body` den Grid-Container, wodurch alle Kinder automatisch zu Grid-Items werden.
- ② mit `grid-template-columns` legen Sie die Rasterspalten an. Für jeden Wert, den Sie nach dem Doppelpunkt angeben, wird eine Spalte angelegt. Für zwölf Spalten müssen Sie entsprechend zwölf Werte angeben. Wenn die Spalten alle gleich breit sein sollen, brauchen Sie also zwölfmal denselben Wert.

Um ein Raster wie auf dem letzten Screenshot zu erhalten, müssen Sie noch Abstände zwischen den Spalten hinzufügen. Das ist wichtig, damit später nebeneinanderliegende Elemente nicht aneinanderstoßen. Dazu benötigen Sie eine zusätzliche Zeile in Ihrem Stylesheet.

```
① body {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr
                          1fr 1fr 1fr;
    ② grid-gap: 10px;
}
```

Angabe der Spaltenzwischenräume

- ① Die Definition des Grids bleibt unverändert.
- ② Die Eigenschaft `grid-gap` erlaubt Ihnen die Angabe der gewünschten Zwischenräume. Der Platz wird bei der Berechnung von `1fr` automatisch berücksichtigt, sodass die zwölf Spalten weiterhin die gesamte Seitenbreite ausfüllen und auch passen. Drei Zeilen CSS reichen zur Definition des oben gezeigten zwölfspaltigen Rasters.

Presentational Markup vermeiden

Für herkömmliche Layouts waren oft Container nötig, um zusammengehörige Elemente zu gruppieren, ohne dass dies semantisch Sinn macht. Im zweispaltigen Layout gab es bisher dafür ein `div` mit der ID „container“. Mit CSS-Grid kann solch semantisch überflüssiges HTML (Presentational Markup) oft entfallen, denn Sie können Elemente direkt auf dem Grid anordnen.

Die Abstände zwischen den Spalten waren in früheren Grids (meist auf der Basis von floats) eine echte Herausforderung. Das Problem: Damit ein Element sich über zwei Rasterspalten erstrecken konnte, also mit den beiden Spalten bündig vorne und hinten abschloss, musste es so breit sein wie die beiden Rasterspalten zuzüglich des Zwischenabstandes. Ein Element über drei Rasterspalten muss so breit sein wie drei Rasterspalten zuzüglich zwei Zwischenabständen und so weiter.

Daher war es nötig, bei einem händisch erstellten Grid die Breiten für Elemente zu definieren, die sich über ein, zwei, drei oder mehr Spalten erstrecken – bis hin zu Elementen über die gesamte Breite von 12 Rasterspalten und 11 Zwischenräumen.

Da Elemente nicht immer linksbündig stehen sollen, muss man auch mögliche Abstände vor dem ersten Element von 1 bis 11 Rasterspalten zuzüglich der Zwischenräume bereitstellen.

Ziemlich viel Rechnerei und viele Zeilen CSS-Code sind für ein solches Grid-System nötig. Mit CSS-Grid hat sich das vollkommen geändert, denn der Browser übernimmt die gesamte Berechnung automatisch. Sie brauchen nur zwei Dinge zu tun. Sie geben Elementen beliebige, sprechende Namen wie „Kopfbereich“, „Navigation“, „Hauptbereich“ und „Fußbereich“ und teilen dem Browser mit, wo sich diese Bereiche befinden sollen.

Dabei sind Sie nicht auf Spalten beschränkt. Ein CSS-Grid ist grundsätzlich zweidimensional. Sie können also Elemente horizontal und vertikal im Raster platzieren.

Für unser zweispaltiges Ein-Drittel-zu-Zwei-Drittel-Layout sind eigentlich nur zwei Rasterspalten nötig. Eine über `1fr`, und eine doppelt so breite, also `2fr`. In diesem Beispiel arbeiten wir aber mit drei Spalten, um zu sehen, wie die Elemente im Grid angeordnet werden.

```

① h1 {
    grid-area: header;
}
main {
    grid-area: content;
}
nav {
    grid-area: navigation;
}
② body {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr; /* 3 Spalten */
    grid-gap: 10px;
}
③ grid-template-areas:
④ "header     header     header"
    "navigation content   content";
}

```

Ein Grid anzulegen, ist aufgrund sprechender Namen sowie geschickter Zeilenumbrüche und Leerzeichen eine leicht nachvollziehbare Sache.

- ① CSS-Grid stellt mit `grid-area` eine Möglichkeit bereit, Elemente zu „Bereichen“ im Grid zu machen und diese zu benennen.
- ② Das bestehende Raster wird auf drei Spalten vereinfacht. So ist es zwar noch möglich, zu sehen, wie Elemente/Bereiche im Raster über mehrere Rasterspalten platziert werden, andererseits wird das Beispiel übersichtlicher als bei zwölf Spalten.

- ③ Die Eigenschaft `grid-template-areas` dient zur Anordnung der Bereiche, über die sich Elemente ausdehnen sollen. Das geschieht zweidimensional. Das heißt, die Elemente/Bereiche können horizontal und vertikal platziert werden (über eine oder mehrere Spalten und über eine oder mehrere Zeilen des Rasters).
- ④ Wenn Sie Zeilenumbrüche und Leerzeichen geschickt setzen, können Sie das Grid im Editor „nachbauen“. Schreiben Sie am besten so, dass Sie das Raster wiedererkennen können. Eine Zeile setzen Sie jeweils in Anführungsstriche. Achten Sie darauf, am Ende der Definition das Semikolon zu setzen.

Die Seitenaufteilung steht. Die Bereiche sind sauber zueinander ausgerichtet (Rahmen zur Kennzeichnung hinzugefügt). Allerdings fallen das Logo und die Schrift innerhalb der Überschrift noch aus dem Raster. Diese sind auch nicht direkt auf dem Raster platziert. Für die anderen Layouts per CSS-Tabellen und Flexbox wurde die `h1` als Container für den Seitentitel und den Startseitenlink benötigt. Dank CSS-Grid ist das nicht nötig. Die beiden Elemente können ebenfalls direkt auf dem Grid platziert werden.

Dazu wird das HTML-Element etwas umgebaut.



Zweispaltiges Layout mit CSS-Grid

```
<a href="index.htm" id="homeLink"></a>
<h1 id="title">Überschrift</h1>
```

- ✓ Das verlinkte Logo und der Seitentitel müssen dank CSS-Grid nicht länger gruppiert werden. Sie erhalten jedes eine eigenständige ID.

Nun können diese Elemente ebenfalls benannt und im Grid als eigenständige Bereiche platziert werden.

```
① a#homeLink {
    grid-area: homeLink;
}
h1#title {
    grid-area: title;
    margin:0;
}
② main {
    grid-area: content;
}
nav {
    grid-area: navigation;
}
body {
    display: grid;
    grid-gap: 10px;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-areas:
```

```
(3) "homeLink" title title"
    "navigation" content content";
}
```

Startseitenlink und Seitentitel werden eigenständige Grid-Bereiche und platziert.

- ① Der Link zur Startseite und der Seiten-
titel werden als Grid-Bereiche benannt.
- ② Der Rest des CSS bleibt identisch.
- ③ Mit einer Ausnahme: Statt einen Header
über drei Spalten platzieren Sie den
Startseitenlink und den Seitentitel
direkt im Grid.

Im Gegensatz zum Flexbox-Layout gilt diese Festlegung für alle Bildschirme. Daher ist es für diese beispielhafte Umsetzung eines Grid-Layouts nötig, mittels @media-Regel dafür zu sorgen, dass das Layout nur auf ausreichend großen Bildschirmen zweispaltig wird.



kap12/2spalter-grid/2spalter-grid-visibleGrid.htm

Das vollständige CSS und HTML für dieses Beispiel finden Sie in den Dateien *kap12/2spalter-grid/2spalter-grid.htm* und *kap12/2spalter-grid/twoColumns-grid.css*.

12.6 Unterschiede der Layoutmethoden

CSS-Tabellen

Das Tabellenlayout eignet sich vor allem für einfache Layouts und ist besonders leicht zu erlernen. Flexboxen sind komplexer und haben entsprechend mehr Möglichkeiten. Tabellen-Layouts sind starr und sehen immer gleich aus – egal wie viele Spalten Sie anlegen: diese bleiben immer erhalten, egal wie breit ein Bildschirm ist.

Flexbox

Flexboxen sind dagegen sehr viel flexibler. Sie spielen gerade im Web ihr Potenzial aus. Ihre Webseite sieht immer gut aus und wirkt wie für genau die gerade benutzte Seitenbreite gemacht – und das ohne Breakpoints. Mit dem Einsatz von Flexboxen erstellen Sie Webseiten, die so flexibel sind, dass es egal ist, mit welchem Browser Besucher auf Ihre Webseite kommen. Ihre Webseite passt sich an.

Flexboxen benötigen eine neue Denkweise. Begriffe wie „oben“ und „unten“, „rechts“ und „links“ spielen keine Rolle mehr, da sich die Boxen je nach Bildschirmbreite über- oder untereinander anordnen. Das gilt allerdings immer für responsive Webseiten und ist kein Nachteil, sondern gewollt.

Flexboxen machen insofern ein Umdenken beim Konzept eines Layouts erforderlich. Wenn Sie Flexboxen gekonnt einsetzen, werden Sie dafür mit Möglichkeiten belohnt, die Ihnen kein anderes Layoutkonzept bietet: Das meiste wird einfacher, vieles überhaupt erst möglich.

CSS-Grid

Mit CSS-Grid steht eine weitere Technik für die Umsetzung von Seitenaufteilungen bereit. Damit ist es besonders komfortabel möglich, ein Raster zu erstellen. Dazu benötigen Sie lediglich ein Container-Element (in der Regel dürfte `body` dafür reichen). Alle Kinder eines Grid-Containers werden automatisch zu Grid-Items. Diese können Sie mittels `grid-area` zu Grid-Bereichen mit sprechenden Namen machen. Diese lassen sich mittels `grid-template-areas` frei auf dem Raster platzieren. Das geht sehr intuitiv und ist gut lesbar, was der Wartungsfreundlichkeit zugutekommt.

Zwar müssen Sie ggf. mehrfache `@media`-Regeln für unterschiedliche Bildschirmbreiten in Ihr CSS einfügen. Aber dafür brauchen Sie nur die Anordnung per `grid-template-areas` anzupassen. Weil der Rest des CSS nicht geändert werden muss, können viele unterschiedliche Layouts mühelos angelegt werden.

Die Spezifikation des CSS-Grid-Moduls ist besonders umfangreich (<https://www.w3.org/TR/css-grid-1/>). Hier im Beispiel wurde lediglich eine leicht nachvollziehbare Möglichkeit gezeigt, wie Sie eine Seitenaufteilung realisieren können. Es ist auch möglich, Layouts zu erstellen, die wie Flexbox ohne `@media`-Regel auskommen. Diese funktionieren in allen modernen Browsern. Fallbacks bei älteren Browsern sind mit den anderen hier beschriebenen Techniken möglich.

Viele weitere Beispiele finden Sie auf der Seite *CSS Grid by example* von Rachel Andrew (<https://gridbyexample.com/>).

Layouts mittels `position`, `float` und `inline-block`

Nicht beschrieben wurden hier Layouts unter Verwendung von absoluter Positionierung, `float` oder `display: inline-block`. Alle sind technisch sehr simpel, benötigen aber Feintuning bei der Berechnung von Breiten und einen geschickten Einsatz von `@media`-Queries, um responsive Layouts bereitzustellen.

Insbesondere mit `position: absolute` werden Sie keine Webseiten erstellen können, die auf allen Geräten funktionieren.

Die anderen beiden haben nur einen Vorteil gegenüber den moderneren Techniken: Sie werden von sehr alten Browsern beherrscht. Wenn Sie also in Ausnahmefällen noch Internet Explorer 8 oder Ähnliches unterstützen müssen, können Sie sich mit diesen Methoden auseinandersetzen.

Solche alten Browser spielen im normalen Web aber keine Rolle mehr und daher können Sie außer für sehr exotische Projekte mit den anderen Methoden leichter und besser zu vernünftigen Ergebnissen kommen. Sie werden von allen modernen Browsern unterstützt; zudem zeigen Browser, die eine Layouttechnik wie Flexbox oder Grid nicht unterstützen, alle Inhalte untereinander an. Damit sind Seiten weiterhin bedienbar und verständlich.

Aus all diesen Gründen ist es sinnvoll, sich mit den neueren, besseren Layouttechniken auseinanderzusetzen. Daher wird auf die veralteten Techniken hier nicht im Einzelnen eingegangen.

12.7 Übungen

Übung 1: Flexbox-Verhalten in verschiedenen Viewport-Breiten überprüfen

Level		Zeit	ca. 10 min
Übungsdatei	<i>kap12\2spalter.htm, kap12\twoColumns-flex.css</i>		
Ergebnisdatei	<i>kap12\2spalter-flex.htm</i>		

1. Öffnen Sie die Datei in einem Editor.
2. Passen Sie den Link zur CSS-Datei so an, dass die Datei *twoColumns-flex.css* verwendet wird.
3. Vergrößern und verkleinern Sie das Browserfenster und beobachten Sie das Verhalten der Flexboxen. Führen Sie dasselbe bei starker Vergrößerung oder Verkleinerung der Inhalte durch.

Übung 2: Machen Sie sich mit CSS-Grid vertraut

Level		Zeit	ca. 15 min
Übungsdateien	<i>kap12\2spalter-grid.htm, kap12\twoColumns-grid.css</i>		
Ergebnisdateien	<i>kap12\2spalter-grid-ergebnis.htm, kap12\twoColumns-grid-ergebnis.css</i>		

1. Benutzen Sie die Datei *2spalter-grid.htm* als Ausgang für ein 3-spaltiges Layout. Fügen Sie dem Hauptbereich eine 3. Spalte hinzu.
2. Fügen Sie außerdem noch eine Fußzeile hinzu, die sich über die gesamte Seitenbreite erstreckt.
3. Der Hauptbereich soll doppelt so groß sein wie die Navigation und die Seitenleiste (also genauso breit wie beide zusammen). Das Ergebnis soll so aussehen wie auf dem Screenshot.



Dreispalter mit festgelegten Spaltenbreiten
(*kap12\uebung3.htm* und *kap12\uebung3.css*)

13

Formulare gestalten

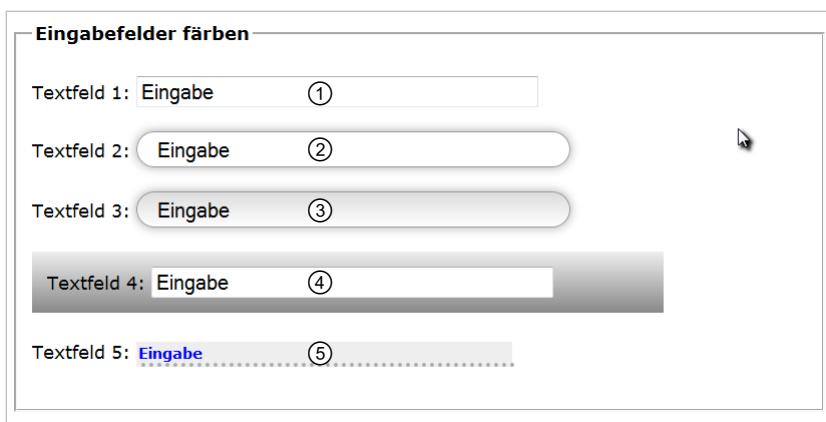
13.1 Einführung in die Gestaltung von Formularen

Wenn Sie nichts anderes angeben, werden Formularelemente in den Browsern grau und weiß dargestellt. Mit CSS haben Sie die Möglichkeit, Formulare individuell zu gestalten.

Mit HTML5 und CSS3 haben Sie darüber hinaus die Möglichkeit, auf Benutzereingaben zu reagieren. So können Sie beispielsweise Felder als Pflichtfeld markieren und abhängig von ihrer Befüllung gestalten. Zum Beispiel können Sie falsch befüllte Felder mit einem roten Rahmen versehen.

13.2 Einzelige Textfelder formatieren

Am weitesten verbreitet sind Eingabefelder für Text. In der folgenden Abbildung sehen Sie Möglichkeiten der Gestaltung. Das Feld ① zeigt die Darstellung ohne CSS-Formatierungen. Beim Feld ② wurden die Ecken abgerundet und ein leichter Schatten wurde eingeblendet. Feld ③ wurde wie Feld ② formatiert, aber mit einem zusätzlichen Farbverlauf versehen. Das vorletzte Feld ④ dagegen wurde gar nicht formatiert, sondern dem umgebenden Element `label` wurde ein Farbverlauf zugewiesen. Im untersten Feld ⑤ wurde der Rahmen bis auf eine gepunktete untere Linie entfernt, eine Hintergrundfarbe zugewiesen und zusätzlich das Format des eingegebenen Textes verändert.



Gestaltete Eingabefelder

Beispiel: *kap13\inputFormatiert.htm*

In diesem Beispiel wird das oben abgebildete HTML-Dokument erstellt. Das Element `fieldset`¹ gruppiert zusammengehörige Formularelemente und mittels `legend` können Sie dazu eine Beschreibung angeben. Das Element `label` dient dazu, eine logische Beziehung zwischen der Beschriftung und dem Eingabefeld herzustellen. Dieser Bezug kann vom Browser oder von anderen Programmen (z. B. Screenreadern) ausgewertet werden. Browser setzen `label` so um, dass eine Beschriftung klickbar wird. So lassen sich vor allem kleine Formularfelder wie Checkboxen leichter aktivieren oder deaktivieren. Im vorliegenden Beispiel führt ein Klick auf die Beschriftung dazu, dass das zugehörige Feld den Fokus erhält. Das bedeutet, der Cursor wird in das Feld gesetzt und dieses kann mit Text gefüllt werden.

```
<body>
<form>
  <fieldset>
    <legend>Eingabefelder formatieren</legend>
    <label>
      Textfeld 1:
      <input id="x1" size="40">
    </label>
    <label>
      Textfeld 2:
      <input id="x2" size="40">
    </label>
    <label>
      Textfeld 3:
      <input id="x3" size="40">
    </label>
    <label>
      Textfeld 4:
      <input id="x4" size="40">
    </label>
    <label>
      Textfeld 5:
      <input id="x5" size="40">
    </label>
  </fieldset>
</form>
</body>
```

¹ `fieldset` macht Probleme im Zusammenspiel mit CSS-Grid und Flexboxen. Formularfelder sollten Sie dennoch sinnvoll gruppieren. Wenn Sie die genannten Techniken verwenden möchten, können Sie zur Gruppierung beispielsweise ein `div` verwenden und diesem die Rolle `group` zuweisen. Mehr über Rollen, Zustände oder Landmarks im HERDT-Buch *Das Web barrierefrei gestalten – Leitfaden für Entwickler*.

Als Nächstes erstellen Sie die Formatierungsanweisungen.

```
<style>
    form { width:18em; }
    legend,
    label { font: 1rem Helvetica, Arial, Geneva, sans-serif; }
    legend { font-weight: bold; }
    input#x2, input#x3 {
        border-radius: 12px;
        border: 1px solid #999;
        padding: 3px 12px;
        box-shadow: 0px 0px 6px #aaa;
    }
    input#x3 {
        background-image: linear-gradient(to bottom, #ddd, #fff);
    }
    label:nth-last-child(2) {
        background-image: linear-gradient(to bottom, #eee, #888);
    }
    input#x5 {
        border: 0px;
        border-bottom: 3px dotted #aaa;
        background-color: #eee;
        font: bold 10px Verdana, Arial, helvetica, sans-serif;
        color: #0000FF;
        width: 250px;
    }
</style>
```

- ① Zu Beginn legen Sie die Formatierung für die Gruppenüberschrift und die Beschriftungen der Eingabefelder fest.
- ② Den Textfeldern mit den IDs x2 und x3 geben Sie abgerundete Ecken und einen leichten Schatteneffekt.
- ③ Das dritte Eingabefeld erhält zusätzlich einen Hintergrundverlauf.
- ④ Der vorletzten Beschriftung (die das vierte Formularfeld enthält) weisen Sie einen Hintergrundverlauf zu und geben zusätzlich Innenabstände und Breite an. Das Formularfeld selbst formatieren Sie nicht.
- ⑤ Neben dem Unterstreichen und dem Einfärben des Hintergrunds ändern Sie die Schriftart und die Farbe des eingegebenen, sichtbaren Textes. Über die Angabe der Eigenschaft width definieren Sie eine pixelgenaue Breite des Eingabefeldes.

Selbstverständlich können Sie auch andere Eigenschaften wie Außenabstände, Hintergrundbilder und mehr einsetzen und miteinander kombinieren. **Achten Sie bei der Gestaltung darauf, dass die Eingabefelder für den Besucher Ihrer Seiten als Eingabefelder erkennbar bleiben!**

13.3 Mehrzeilige Textfelder, `legend` und `fieldset` formatieren

Beispiel: *kap13\textarea.htm*

Aus dem letzten Beispiel übernehmen Sie das HTML, entfernen aber alle Felder bis auf eines und fügen ein Element `textarea` hinzu:



Mehrzeiliges Textfeld, legend und fieldset sind passend zu den Eingabefeldern gestaltet.

```
<form>
  <fieldset>
    <legend>Mehrzeilige Eingabefelder formatieren</legend>
    <label>
      Textfeld 1:
      <input id="x1" size="40">
    </label>
    <label>
      Textfeld 2:
      <textarea cols="30" rows="10"></textarea>
    </label>
  </fieldset>
</form>
```

Zu den Formatierungsanweisungen für das Element `input` kommen nun Deklarationen für `textarea` hinzu – auch `fieldset` und `legend` sollen passend zu den Eingabefeldern gestaltet werden, sodass ein Formular entsteht, das so aussieht wie in der Abbildung.

Beginnen Sie mit der Formatierung von `textarea`.

```
① input, textarea {  
    border-radius:12px;  
    border:1px solid #999;  
    box-shadow: 0px 0px 6px #aaa;  
    background-image: linear-gradient(top, #dddddd, #ffffff);  
}  
② input {  
    padding: 3px 8px;  
    width: 20em;  
}  
textarea {  
    padding: 3px 11px;  
    resize: vertical;  
    width: 20em;  
}
```

- ① Die aus dem letzten Beispiel bekannten Formate sollen auch für mehrzeilige Textfelder gelten.
- ② In den Dimensionen unterscheiden sich die Felder etwas. Um das auszugleichen, müssen Sie die Werte für `padding` getrennt angeben.

Im nächsten Schritt bekommt `fieldset` als Hintergrund einen Farbverlauf. An diesem Beispiel sehen Sie, wie Sie **ältere Browerversionen** dazu bringen, den Hintergrundverlauf korrekt darzustellen.

```
① fieldset {  
    border:1px solid #999;  
    width:18em; border-radius: 12px;  
}  
② background-image:  
    linear-gradient(to bottom, #eeeeee, #888888) #eee;
```

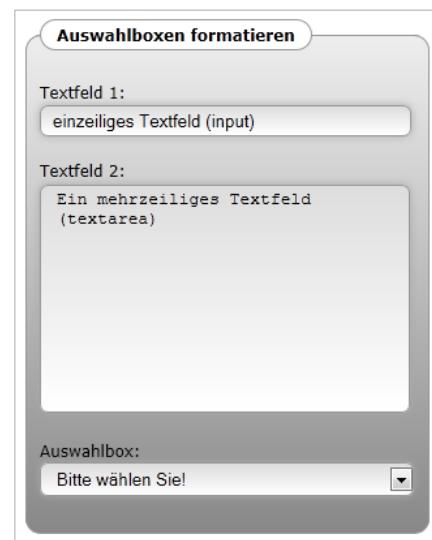
- ① Zunächst selektieren Sie `fieldset` und machen Angaben zur Breite und zur Rundung der Ecken. Hier sehen Sie, dass bereits drei Angaben für die Ecken nötig sind, um alle Browser abzudecken.
- ② Für ganz alte Browser geben Sie zunächst eine Hintergrundfarbe an, um auch die Browerversionen zu unterstützen, die mit Verläufen gar nichts anfangen können.

13.4 Auswahlboxen formatieren

Neben ein- und mehrzeiligen Eingabefeldern können Sie auch Auswahlboxen formatieren. Diese werden auch Kombinationsfelder, Dropdown-Listen oder Pop-up-Menüs genannt. Um sie zu erstellen, notieren Sie das Element `select` in Ihrem HTML-Dokument. Neben dem Ändern der Schrift- und Hintergrundfarbe können Sie auch die Schriftart und die Schriftgröße anpassen. Auch die einzelnen Auswahlmöglichkeiten einer Auswahlbox können Sie farblich unterschiedlich formatieren.

Allerdings unterstützen nicht alle Browser jede Formatierungsmöglichkeit. Testen Sie die Darstellung daher in unterschiedlichen Browsern und Versionen. Je mehr Anpassungen Sie vornehmen, desto größer werden die Unterschiede. Das ist nicht unbedingt ein Problem. Achten Sie allerdings auf eine verständliche Darstellung in allen Browsern, dann können Sie Browsern mit mehr Möglichkeiten eine weiter verbesserte Gestaltung mitgeben (progressive enhancement).

Im Verlauf dieses Abschnittes werden Sie das vorherige Beispiel so erweitern, dass es aussieht wie die nebenstehende Abbildung. In der Abbildung entspricht die Datei `select.htm` der Datei `textarea.htm`, wurde jedoch um eine Auswahlbox erweitert.



Beispiel: kap13\select.htm

Für dieses Beispiel dient die Datei `kap13\textarea.htm` als Ausgangspunkt. Fügen Sie ihr die Elemente `select` und `option` hinzu, um eine Auswahlbox zu erhalten:

```
<form>
  <fieldset>
    <legend>Auswahlboxen formatieren</legend>
    <p>
      <label>
        Textfeld 1:<br>
        <input id="x1" type="text" name="x1" size="40">
      </label>
    </p>
    <p>
      <label>
        Textfeld 2:<br>
        <textarea cols="30" rows="10"></textarea>
      </label>
    </p>
    <p>
      <label>Auswahlbox:<br>
        <select id="x3">
          <option>Bitte wählen Sie!</option>
          <option>Erster Eintrag</option>
        </select>
      </label>
    </p>
  </fieldset>
</form>
```

```

<option>Weiterer Eintrag</option>
<option>Dritter Eintrag</option>
</select>
</label>
</p>
</fieldset>
</form>

```

Zunächst sorgen Sie dafür, dass die Auswahlbox so aussieht wie die Elemente `input` und `textarea`. Dazu erweitern Sie den entsprechenden Selektor einfach.

```

input, textarea, select {
/* Hier stehen die Formate aus dem letzten Beispiel */
}

```

Das weitere Aussehen legen Sie per CSS wie folgt fest:

```

① select {
    width:21.5em;
    padding: 3px 0 3px 8px;
}

② select option {
    background-color: #ddd;
}

③ select option:nth-child(2n-1) {
    background-color: #fff;
}

```

- ① Auch hier müssen Sie wieder an der Breite und den Rändern einige Anpassungen vornehmen, damit dieses Formularelement bündig mit den anderen abschließt.
- ② Geben Sie eine Hintergrundfarbe für die Einträge der Auswahlliste an.
- ③ Allen ungeraden Auswahlmöglichkeiten geben Sie eine andere Hintergrundfarbe, sodass ein Streifenmuster entsteht (funktioniert nicht in allen Browsern).



Unterschiedlich unterlegte Auswahlmöglichkeiten (kap13\select.htm)

Angaben zu verschiedenen Schriftgrößen, Schriftarten und Umrandungen sind ebenfalls möglich und werden auch von allen Browsern umgesetzt.

Einige Formularelemente wie `select`, Radiobuttons oder Checkboxen bringen ihr Browser- oder betriebssystem-spezifisches Design mit. Es gibt für die unterschiedlichen Elemente verschiedene Hacks zur Gestaltung (bekannt ist beispielsweise der Checkbox-Hack, bei dem das eigentliche Bedienelement unsichtbar gemacht wird und ein eigenes grafisches Element an seiner Statt mittels `:before` eingefügt wird; ein Praxisbeispiel finden Sie in Abschnitt 12.4, „Benutzerfreundliche Menüs erstellen“).

Für Elemente wie `select` können Sie mittels `appearance: none` die Standarddarstellung entfernen. Das funktioniert in allen modernen Browsern.



Auswahlbox mit appearance: none: Die Erkennungsmerkmale werden entfernt und können durch eigene ersetzt werden.

13.5 Schaltflächen gestalten

Auch für andere Schaltflächen gilt: Sie werden in den verschiedenen Browsern unterschiedlich dargestellt und beschriftet.

Die nebenstehende Abbildung zeigt die Standardschaltflächen der verschiedenen Browser unter Windows 11 in folgender Reihenfolge:

- ① Mozilla Firefox
- ② Opera
- ③ Microsoft Edge
- ④ Google Chrome



Auf weiteren Betriebssystemen (neuere oder ältere Windows-Versionen, Linux, macOS, mobile Geräte) werden wiederum andere Darstellungen genutzt.

Mit CSS können Sie auch das Aussehen von Schaltflächen gestalten und dadurch für ein einheitliches Design sorgen.

! Beachten Sie, dass Sie mit jeder Anpassung am Aussehen von Bedienelementen eine Darstellung ändern, die der Benutzer von seinem Betriebssystem kennt und an die er sich gewöhnt hat. Alle Formularelemente sollten Sie so formatieren und beschriften, dass dem Benutzer auch nach der Anpassung noch klar ist, um was für Bedienelemente es sich jeweils handelt und welchem Zweck sie dienen.

Um einige Möglichkeiten zu demonstrieren, werden im nächsten Abschnitt den Schaltflächen verschiedene Formatierungen zugewiesen. Das etwas umfangreichere Beispiel wird so aufgeteilt, dass im ersten Formular die Beschriftungen der Schaltflächen geändert werden. Danach werden die Schaltflächen eingefärbt und zum Schluss werden ihnen Grafiken zugewiesen.

Das oben begonnene Beispiel wird im übernächsten Abschnitt mit einem zum restlichen Design passenden Absende-Knopf versehen.

Beispiel: kap13\buttons.htm**Erstes Formular**

```

<h1>Schaltflächen gestalten</h1>
<form action="#" name="formular1">
    <h2>Schaltflächen mit unterschiedlicher Schrift</h2>
①    <input type="submit" value="Abschicken">
②    <input
        style="font-size:1.125em;"
        type="submit"
        value="Abschicken">
③    <input
        style="
            color: #0f0;
            font-weight: bold;"
        type="submit"
        value="Abschicken">
</form>

```

- ① Zum Vergleich bleibt die erste Schaltfläche unformatiert. Sie wird in Abhängigkeit von den Formatvorgaben des jeweiligen Browsers dargestellt.
- ② Die Beschriftung wird auf 1.125em vergrößert. Damit vergrößert sich auch automatisch die Schaltfläche.
- ③ Bei der dritten Schaltfläche soll die Schrift grün und fett werden.

Zweites Formular

```

<form action="#" name="formular2">
    <h2>Schaltflächen mit Hintergrundfarbe und Rahmen</h2>
④    <input style="background-color:rgb(32,132,255); color:#fff"
        type="submit"
        name="x1" value="Abschicken">&nbsp;
⑤    <input style="border:3px dotted f00" type="submit" name="x2"
        value="Abschicken">&nbsp;
⑥    <input style="border:0px; border-top:3px solid #f00;
        border-bottom:3px solid #f00; color:green; font-weight:
        bold;"
        type="submit" name="x3" value="Abschicken">
</form>

```

- ④ Damit die Schaltfläche farbiger wird, weisen Sie ihr über die Eigenschaft background-color einen blauen Hintergrund zu. Damit die Beschriftung trotz dunkleren Hintergrunds lesbar bleibt, wechseln Sie die Schriftfarbe auf Weiß.
- ⑤ Auch eine veränderte Umrandung ist möglich. In diesem Fall setzen Sie mit der border-Angabe einen 3 Pixel breiten „Schmuckrand“.
- ⑥ Verfeinern können Sie die Umrandung, indem Sie für jede Seite der Schaltfläche den Rahmen separat definieren. Somit sind auch mehrfarbig umrandete Schaltflächen möglich.

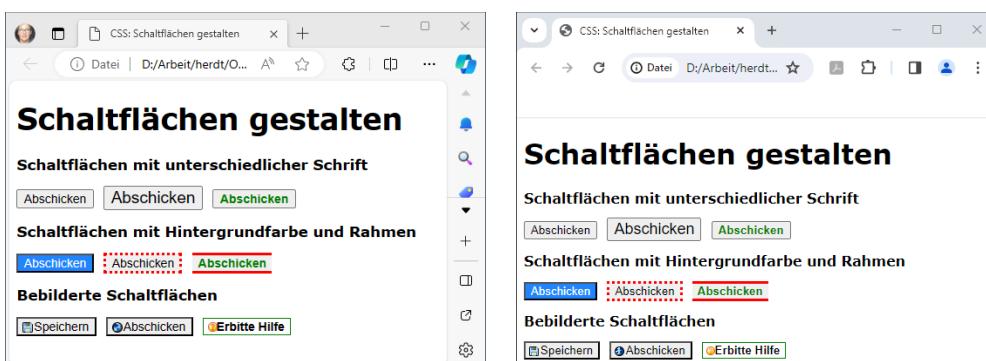
Drittes Formular

```

<form action="#" name="formular3">
    <h2>Bebilderte Schaltflächen</h2>
    ⑦ <input
        style="
            background-color:#eee;
            background-repeat:no-repeat;
            background-image:url(..../images/disk.gif);
            background-position:4px 4px;">
        type="submit" value="Speichern">
    ⑧ <input
        style="
            background-color:#eee; background-repeat:no-repeat;
            background-image:url(..../images/remote_a.gif);
            background-position:4px 4px;">
        type="submit" value="Abschicken">
    ⑨ <input
        style="
            background-color:#fff;
            background-repeat:no-repeat;
            background-image:url(..../images/help_a.gif);
            background-position:4px 4px;
            border:1px solid #0f0;
            color:#000;
            font-weight:bold;">
        type="submit" value="Erbitte Hilfe">
</form>

```

- ⑦ Zuerst setzen Sie die Hintergrundfarbe der Schaltfläche auf ein helles Grau. Damit das Hintergrundbild, das als Schaltflächenbild benutzt werden soll, nicht immer wieder angezeigt wird, unterbinden Sie die Wiederholung mit der CSS-Eigenschaft `background-repeat` und dem Wert `no-repeat`. Um die Grafik etwas einzurücken, setzen Sie einen linken und oberen Abstand von vier Pixeln.
- ⑧ Das zweite Beispiel erhält ein animiertes Bild im GIF-Format, das ebenfalls genutzt werden kann.
- ⑨ Die letzte Schaltfläche wird so verändert, dass sie nicht mehr wie eine Schaltfläche wirkt. Der Hintergrund wird auf Weiß gesetzt und verschmilzt dadurch mit dem Hintergrund der Webseite. Damit eine Abtrennung ersichtlich ist, setzen Sie um die Schaltfläche einen grünen Rahmen. Die Grafik binden Sie als Hintergrundbild ein. Die Beschriftung formatieren Sie fett und in schwarzer Farbe.



Die drei Formulare in Microsoft Edge und Google Chrome (kap13\buttons.htm)

Standardmäßig ist die Breite einer Schaltfläche abhängig von der Länge des angegebenen Wertes. Einheitliche Breiten erhalten Sie über die Stylesheet-Eigenschaft width.

```
<input type="submit" value="Abschicken" style="width: 12.5rem">
```

Beispiel: *kap13\absenden.htm*

Fügen Sie nun dem bereits begonnenen Beispielformular eine passende Schaltfläche zum Absenden des Formulars hinzu.

Die Beschriftung lassen Sie zunächst auf dem vom Browser vergebenen Standardwert.

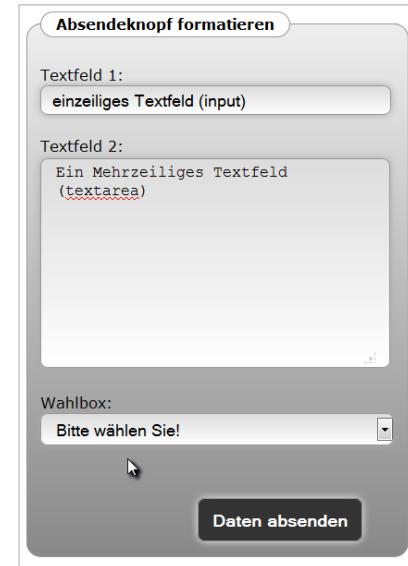
Die Änderungen, die Sie dazu am HTML-Dokument vornehmen müssen, sind wieder schnell gemacht. Die eigentliche Arbeit steckt wie immer in der CSS-Datei.

Fügen Sie vor dem schließenden Tag </fieldset> die folgende Zeile in Ihr Dokument ein:

```
<input type="submit">
```

Mittels CSS ändern Sie Hintergrundfarbe, Schriftfarbe, Abstände und mehr.

```
input[type="submit"] {
    float: right;
    width: 10em;
    padding: 8px;
    color: #fff;
    background-color: #333;
    background-image: none; /* Entfernt den zugewiesenen Verlauf */
    font-weight:bold;
    margin: 2em 2em 0 0;
    box-shadow: 0px 0px 6px #eee; }
```



Jedes Formular benötigt einen Button zum Versenden.

13.6 HTML5 und CSS3: Formulartypen und Pseudoklassen

HTML5 bietet verschiedene Input-Typen. Diese können Sie nutzen, um Ihre Formulare zu gestalten.

HTML5-Input-Typen

Die verschiedenen Input-Typen geben Ihnen die Möglichkeit, dem Browser mitzuteilen, welche Daten Sie in dem jeweiligen Formularfeld erwarten.

Beispiel: *kap13\html5Inputs.htm*

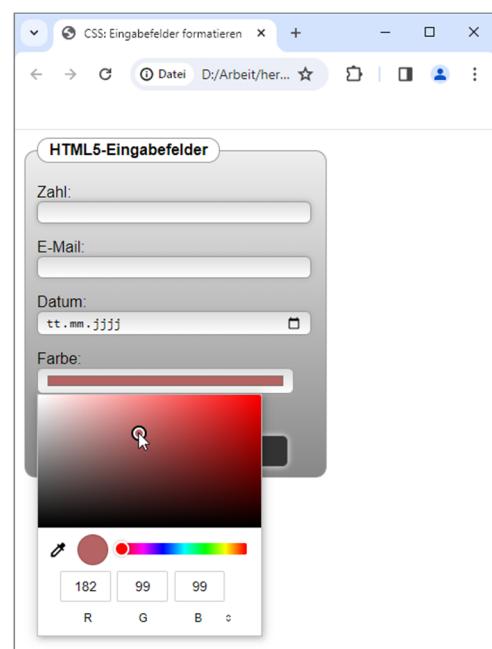
Hier die wichtigsten HTML5-Input-Typen:

- ① <input type="number">
- ② <input type="email">
- ③ <input type="date">
- ④ <input type="color">

- ① In diesem Feld sind nur Zahlen erlaubt.
- ② Hier soll eine E-Mail-Adresse eingegeben werden.
- ③ Diese Angabe definiert ein Feld für die Eingabe eines Datums.
- ④ Dieses Feld erwartet eine Farbangabe, die über ein Farbfeld ausgewählt werden kann.

Die Browser kontrollieren von sich aus, ob die Eingabe korrekt war, z. B. bei der Angabe der Mail-Adresse.

Da diese Felder weitgehend wie Textfelder formatiert werden, ergeben sich bei der Formatierung keine neuen Möglichkeiten. Allerdings machen einige davon Probleme. So wie die bekannten Felder für den Datei-upload oder Auswahlboxen kommen hier zusätzliche Schaltflächen zum Einsatz, deren Formatierungen sich nicht anpassen lassen. Sie müssen diese mittels appearance: none vollständig entfernen und durch eigene ersetzen. Je nach Browser kommen mehrere Bedienelemente pro Feld zum Einsatz. Im Kalender beispielsweise können die Nutzer ein Datum auswählen oder über die Pfeile nach oben und unten blättern.



Google Chrome zeigt alle neuen Input-Typen. Die Darstellung bestimmt der Browser.

Pseudoklassen für Formulare

Für die Gestaltung spielen die unterschiedlichen Typen auch eine wichtige Rolle, weil diese mit dem Attribut-Selektor ausgewählt werden können, ohne dass Klassen nötig wären.

```
[type="foo"]
```

Zusätzlich stehen Pseudoklassen zur Verfügung, um Zustände wie „benötigtes Feld“ oder „falsches Eingabeformat“ gestalten zu können.

:required	Felder, die ausgefüllt werden müssen und im HTML entsprechend mit dem Attribut required gekennzeichnet wurden
:optional	Felder, die nicht als required markiert sind
:valid	Felder, die korrekt ausgefüllt wurden
:invalid	Felder, die nicht korrekt ausgefüllt wurden

Beispiel: *kap13\kontakt.htm*

Aus dem bisherigen Beispiel erstellen Sie im nächsten Schritt ein Kontaktformular. Der Kopf mit den CSS-Angaben bleibt dabei unverändert.

Dieses Beispiel soll die Möglichkeiten der aktuellen Browser demonstrieren. In aller Regel benötigen Sie das Geburtsdatum und das Alter des Nutzers nicht. Auch die Anrede ergibt sich in aller Regel aus dem Namen. Fragen Sie nur unbedingt notwendige Daten ab.

Je länger ein Formular ist, desto seltener wird es ausgefüllt.

Im ersten Schritt stellen Sie das HTML-Dokument um und erweitern es um ein paar Eingabefelder. Dies sind ein Feld zur Angabe der E-Mail-Adresse und ein weiteres für das Geburtsdatum. Außerdem werden Pflichtfelder mit dem Attribut required gekennzeichnet.

Das Ergebnis sehen Sie in der Abbildung rechts.

The screenshot shows a contact form titled "Kontakt". The form fields are as follows:

- "Ihre Nachricht an uns" (Your message to us) - A dropdown menu asking "Wie werden Sie angesprochen? Bitte wählen Sie eine Anrede".
- "Wie heißen Sie?" (What is your name?) - A text input field asking "Bitte geben Sie Ihren Namen an".
- "Wie alt sind Sie?" (How old are you?) - A dropdown menu asking "Bitte wählen Sie Ihre Altersgruppe aus".
- "Wie lautet Ihre E-Mail-Adresse? *" (What is your email address? *) - A text input field asking "Sagen Sie uns, wie wir Sie erreichen".
- "Wann wurden Sie geboren?" (When were you born?) - A dropdown menu asking "Bitte wählen Sie Ihr Geburtsjahr aus".
- "Was möchten Sie uns mitteilen? *" (What do you want to tell us? *) - A text area asking "Ihre Nachricht an uns".

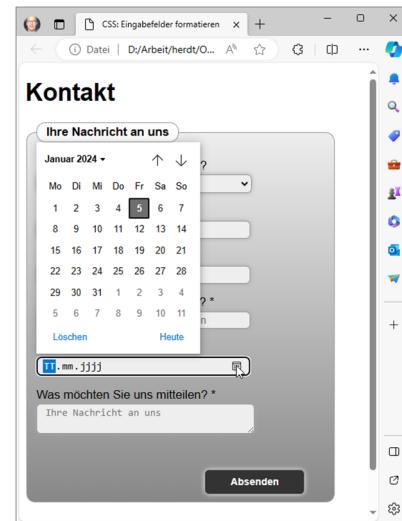
At the bottom right is a "Senden" (Send) button.

kap13/kontakt.htm

Das HTML-Dokument dazu sieht folgendermaßen aus:

```
<h1>Kontakt</h1>
<form action="formular.htm">
  <fieldset>
    <legend>Ihre Nachricht an uns</legend>
    <p>
      <label>
        ①       Wie werden Sie angesprochen?<br>
        <select id="x3" name="x3">
          <option>Bitte wählen Sie eine Anrede</option>
          <option>Frau</option>
          <option>Herr</option>
        </select>
      </label>
    </p>
    <p>
      <label>
        ②       Wie heißen Sie? *<br>
        <input type="text" placeholder="Vorname Nachname" required>
      </label>
    </p>
    <p>
      <label>
        ③       Wie alt sind Sie?<br>
        <input type="number">
      </label>
    </p>
    <p>
      <label>
        ④       Wie lautet Ihre E-Mail-Adresse? *<br>
        <input type="email"
               placeholder="Sagen Sie uns, wie wir Sie erreichen können."
               required>
      </label>
    </p>
    <p>
      <label>
        ⑤       Wann wurden Sie geboren?<br>
        <input type="date">
      </label>
    </p>
    <p>
      <label>
        ⑥       Was möchten Sie uns mitteilen? *<br>
        <textarea placeholder="Ihre Nachricht an uns"
                  required></textarea>
      </label>
    </p>
    <input type="submit">
```

- ① Beginnen Sie mit der Auswahlbox für die Anrede.
- ② Dann fragen Sie nach dem Namen des Kontakt suchenden. Da Sie diesen benötigen, um später mit dem Kontakt suchenden kommunizieren zu können, verwenden Sie hier ein Pflichtfeld. Um den Nutzer bei der Eingabe seiner Daten zu unterstützen, können Sie mit dem Attribut `placeholder` einen Text angeben, der verschwindet, wenn der Nutzer den Fokus auf das Feld setzt, es also zum Beispiel anklickt.
- ③ Hier verwenden Sie das erste Feld mit einem HTML5-Input-Typ. Der Typ `number` erlaubt nur die Angabe von Zahlen. Der Browser überprüft dies beim Absenden.
- ④ Ohne E-Mail-Adresse könnten Sie mit dem Interessenten nicht in Kontakt treten – also markieren Sie dieses Feld als Pflichtfeld. Dank des Typs `email` überprüft der Browser vor dem Versenden, ob die angegebene Adresse plausibel ist, also zum Beispiel, ob sie das Zeichen @ enthält.
- ⑤ Felder vom Typ `date` erlauben nur Datumsangaben im DIN-Format JJJJ-MM-TT – vorgesehen ist allerdings, dass der Nutzer das nicht schreiben muss, sondern dass er den Tag mit der Maus aus einem kleinen Kalender auswählen kann. Bisher unterstützt dies nur der Browser Opera (siehe Abbildung rechts). Alle anderen zeigen ein normales Textfeld.
- ⑥ Als letztes Eingabefeld geben Sie dem Nutzer in einem mehrzeiligen Textfeld die Möglichkeit, Ihnen eine Nachricht zu schreiben.
- ⑦ Mit einem Input-Feld vom Typ `submit` kann er das Formular abschicken. In neueren Browsern bewirkt dies vor dem Versand eine Überprüfung, ob die Felder korrekt ausgefüllt wurden und ob alle Pflichtfelder Zeichen enthalten.



Die Datei „kap13\kontakt.htm“ in Microsoft Edge mit geöffnetem Datumswähler

! Die Überprüfung ist wichtig und praktisch – sie kommt allerdings schnell an ihre Grenzen. So kann sie nicht feststellen, ob die Einträge sinnvoll sind. Es wird nur geprüft, ob sie plausibel, also syntaktisch korrekt sind. Aber nicht einmal darauf können Sie sich verlassen, weil ältere Browser diese Funktion nicht besitzen und jedes Formular abschicken. Wenn Sie Daten noch genauer prüfen wollen, müssen Sie dies im Browser mittels JavaScript oder auf dem Server mit einer geeigneten Programmiersprache erledigen.

Einige Input-Typen erlauben die Verwendung des Attributes `placeholder` nicht. In unserem Beispiel betrifft das die Felder vom Typ Nummer und Datum. Daher besitzen die Felder für das Alter und das Geburtsdatum keine Beispieltexte. Dennoch würden die Beispieltexte in vielen Browsern korrekt angezeigt. In der Praxis werden Sie häufiger Webseiten antreffen, die nicht valide sind, aber in allen Browsern wie gewünscht funktionieren. Dennoch sollten Sie generell darauf achten, validen Code zu erzeugen. Sollten Sie gute Gründe haben, ausnahmsweise auf Validität zu verzichten, müssen Sie unbedingt in allen Browsern überprüfen, ob Sie das gewünschte Ergebnis erhalten, und müssen diese Tests in allen neu erscheinenden Browserversionen wiederholen. Da die Browser die Standards immer besser unterstützen, sind Sie mit standardkonformem Code für die Zukunft besser gewappnet.

13.7 Komplettes Formular

Beispiel: *kap13\formular.htm*

Das Kontaktformular, das Sie im letzten Abschnitt erstellt haben, enthält die gebräuchlichsten Felder. Jetzt könnten Sie noch Schalter hinzufügen. HTML bietet zwei Arten von Schaltern:

- ✓ Checkboxen können mit einem Häkchen versehen werden. Aus einer Gruppe von mehreren Checkboxen können beliebig viele oder gar keine ausgewählt werden.
- ✓ Radiobuttons (siehe Abbildung) bieten dem Nutzer an, genau eine von mehreren Möglichkeiten auszuwählen.

Das Aussehen beider Schalter können Sie kaum beeinflussen. Zudem sehen diese in den verschiedenen Browsern auch nicht gleich aus. Die Unterschiede sind allerdings gering und fallen den Nutzern in der Regel nicht auf. Schließlich benutzen die Besucher Ihrer Website in der Regel immer denselben Browser.

Weil an den Schaltern kaum Formatierungen vorgenommen werden können, soll als Beispiel ein Satz von Radiobuttons genügen. Fügen Sie den folgenden Code vor der Absenden-Schaltfläche ein:

```
<p>
<label>
    <input type="radio" name="newsletter" checked value="JA">
        Ich möchte den Newsletter bestellen.
</label><br>
<label>
    <input type="radio" name="newsletter" value="nein">
        Nein, ich möchte ihn nicht bekommen.
</label>
</p>
```

Ein wenig CSS ist noch nötig, um Formate wieder zu entfernen, die an den Radiobuttons stören würden:

```
input[type="radio"] {
    width: 1em;
    margin-right: 1em;
    padding: 0; }
```

Radiobuttons sollen nicht so breit wie normale Textfelder sein. Hier genügt eine Breite von `1em`. Der rechte Außenabstand `margin` sorgt für Abstand zwischen dem Schalter und seiner Beschriftung. Weitere Abstände sind hier nicht nötig, weshalb die Innenabstände `padding` entfernt werden.

The screenshot shows a contact form titled "Kontakt". It includes fields for "Wie werden Sie angesprochen?", "Wie heißen Sie?", "Wie alt sind Sie?", "Wie lautet Ihre E-Mail-Adresse?", "Wann wurden Sie geboren?", "Was möchten Sie uns mitteilen?", and a radio button group for newsletter preferences. A "Senden" button is at the bottom.

Das Formular mit allen Feldern
(*kap13\formular.htm*)

13.8 Übung

Formularelemente im Aussehen verändern

Level		Zeit	ca. 30 min
Übungsdatei	<i>kap13\formular.htm</i>		
Ergebnisdateien	<i>kap13\uebung1.htm, kap13\uebung2.htm, kap13\uebung3.htm</i>		

1. Verändern Sie das Aussehen des Kontaktformulars, indem Sie fokussierten Eingabefeldern eine gelbe Hintergrundfarbe geben.
2. Ändern Sie zusätzlich die Schriftart des eingegebenen Textes. Die Schriftgröße soll vom Elternelement übernommen (geerbt) werden.
3. Geben Sie der Schaltfläche einen blauen Verlauf.

14

Wie geht es weiter?

In diesem Kapitel erfahren Sie, was Sie mit den hier beschriebenen Kenntnissen anfangen können, wo Sie sich weiterbilden können und welche Techniken zusätzlich sinnvoll sind.

14.1 CSS ist umfangreich und komplex

Auch wenn es auf den ersten Blick nicht so scheint: Ein CSS-Dokument ist nichts weiter als ein Satz von Regeln, mit denen Elementen Eigenschaften und Werte zugewiesen werden. So weit, so simpel. Allerdings benötigen insbesondere große Projekte sauber strukturierte und aufgebaute CSS-Dateien.

Wenn Sie Ihr CSS ohne Konzept aufbauen, werden große Projekte unwartbar.

Ein paar Tipps:

- ✓ Strukturieren Sie Ihre CSS-Dateien.
Fassen Sie zusammengehörende Styles unter (Kommentar-)Überschriften zusammen.
- ✓ Seien Sie bei der Wahl der Selektoren so unspezifisch wie möglich.
- ✓ Notieren Sie die spezifischsten Selektoren zuletzt.
- ✓ Notieren Sie hilfreiche Informationen.
 - ✓ Ein Inhaltsverzeichnis ermöglicht es, Worte aus Überschriften zu markieren, zu kopieren und in die Suche einzufügen, sodass Sie bestimmte Bereiche des Stylesheets direkt anspringen können.
 - ✓ Stellen Sie das Datum der letzten Änderung an den Anfang der Datei.

Da Ihnen in CSS keine echten Überschriften zur Verfügung stehen, müssen Sie sich mit Kommentaren behelfen.

- ✓ Beispiele für mögliche CSS-„Überschriften“. Da es hierfür keine Vorschriften gibt, können Sie diese „gestalten“, wie Sie möchten. Es sind keine echten Überschriften.
- ✓ Nummerierungen können viel Arbeit sein, vor allem, wenn Sie Abschnitte umstellen möchten. Wenn Sie tatsächlich durchnummerieren wollen, was hilfreich sein kann, sollten Sie das erst ziemlich am Ende des Projektes machen (wenn die Version 1.0 absehbar ist). Dann werden Sie die Struktur kaum noch umwerfen und für ein späteres Bearbeiten haben Sie dennoch eine leicht nachvollziehbare Struktur.

```
/* =====
   1. Beispiel für eine Überschrift
===== */
/* -----
   1.1. Beispiel für eine Zwischenüberschrift
----- */
/* 1.1.1. Weitere Überschriftenebene
- - - - - - - - - - - - - - - - - - - - */
/* Erläuterungen und Erklärungen (normaler Kommentar) */
```

Hilfsmittel

Editor

Die Wahl des persönlichen Favoriten hängt von der eigenen Arbeitsweise und Denkart ab. Wir alle funktionieren anders, denken anders, organisieren uns anders. Bei einem Editor ist es daher entscheidend, dass er uns in unserem Vorgehen optimal unterstützt (das betrifft natürlich auch die anderen Werkzeuge, aber mit dem Editor verbringt man die meiste Zeit).

Ein ausgezeichneter Editor speziell für die Frontend-Techniken HTML, CSS und JavaScript ist **Brackets** von Adobe. Da die Oberfläche sehr aufgeräumt ist, erschließen sich die mächtigen Funktionen nicht alle auf den ersten Blick. Es gibt (wie zu den anderen Editoren) zahlreiche Videos und Tutorials im Web, die die Leistungsfähigkeit demonstrieren.

Andere beliebte Editoren sind **Sublime Text** und das kostenlose Pendant **Atom** – beide sind extrem anpassbar. **vim** – der Urahn aller Editoren – scheint niemals aus der Mode zu kommen.

Wer den Einarbeitungsaufwand nicht scheut, den belohnen komplettete Entwicklungsumgebungen wie **Eclipse** oder **Dreamweaver** mit einer einheitlichen Oberfläche für alle anfallenden Aufgaben.

Mächtig und doch vergleichsweise einsteigerfreundlich ist das ebenfalls beliebte **Visual Studio Code** von Microsoft.

Wenn Sie den passenden Editor gefunden haben, müssen Sie sich weniger auf die Programmbedienung konzentrieren und haben den Kopf frei für das Projekt. Sie arbeiten dann schneller, besser und mit mehr Freude. Probieren Sie auch gelegentlich neue Editoren aus. Es lohnt sich!

Änderungen nachverfolgen und wiederherstellen (Versionsverwaltung)

Als **Versionsverwaltung** hat sich **git** weitgehend gegenüber dem früher sehr beliebten **svn** durchgesetzt. Es gibt zahlreiche kostenlose und kostenpflichtige grafische Oberflächen; die Integration in Ihren Lieblingseditor erfolgt entweder nativ oder als Plug-in.

Unter <https://github.com> finden Sie zahlreiche Open-Source-Projekte (auch Module für die eigene Website wie Bilderkarusselle, Aufklapp-Navigationen usw.). Wenn Sie git installiert haben, können sie diese Projekte „**klonen**“, also eine Kopie auf Ihren eigenen Rechner ziehen.

Sie können sich aber auch in die Weiterentwicklung einbringen. GitHub ist mehr als nur eine „Softwarehalde“. Sie können mit anderen Entwicklern kommunizieren, Tickets (Fehlermeldungen) zu Ihren eigenen Projekten abarbeiten – oder Fehler in anderen Projekten melden.

GitHub lässt sich wohl am besten mit einem sozialen Netz für Entwickler vergleichen, die sich nicht nur austauschen, sondern etwas produzieren wollen.

Backups erstellen

Kein Backup, kein Mitleid! – so hart werden Hilfsanfragen bei Datenverlust in Foren mitunter beantwortet.

Was auch immer Sie mit Ihrem Computer, Tablet oder Smartphone erschaffen. **Sorgen Sie dafür, Ihre Daten nicht zu verlieren.**

Backup-Software ist für jedes System reichlich vorhanden. Sinnvoll ist es, mehrere Backups anzulegen, die nicht alle am selben Ort aufbewahrt werden.

Bei einem Einbruch können schließlich Computer und Backup-Medium entwendet werden. Auch Blitzschlag, Wasser, Feuer und andere Ereignisse können sämtliche an einem Ort befindliche Kopien mit einem Mal zerstören.

Eine veröffentlichte Webseite liegt zusätzlich mindestens noch auf einem Webserver. Doch ein unbeabsichtigtes Hochladen eines unfertigen Zwischenstandes kann die öffentlich verfügbare Fassung ebenfalls zerstören.

Hier helfen die bereits genannten Versionierungssysteme, einen früheren Stand wiederherzustellen.

Komfortabel (aber meist kostenpflichtig) sind Cloud-Backups, bei denen die Daten auch gleich an einem anderen Ort gespeichert werden.

Ein Backup ist keine Kopie der Daten. Ein Backup ist ein Konzept. Überlegen Sie genau, wie Sie welche Daten sichern. Webseiten liegen wie gesagt bereits an einem zweiten Ort (dem Rechenzentrum des Hosting-Anbieters), doch für die Originale der verwendeten Bilder trifft das wahrscheinlich nicht zu.

Bedenken Sie, dass Cloud-Speicherdiene, die automatisch Daten synchronisieren, auch Daten auf anderen Computern zerstören können. Wenn ein Verschlüsselungstrojaner auf einem Ihrer Geräte Ihre Daten unbrauchbar macht, werden **Dropbox**, **OneDrive** und andere die geänderten Daten brav in die Cloud spiegeln, von wo aus sich die anderen Geräte wiederum bedienen (häufig bewahren solche Dienste – genügend Speicherplatz vorausgesetzt – ältere Versionen der Dateien auf. Das müssen Sie aber vorher abklären).

Ein Teil Ihres Backup-Konzeptes sollte also sicherstellen, an einem entfernten Ort frühere Stände Ihrer Daten bereitzuhalten.

Präprozessoren

Präprozessoren erweitern die Möglichkeiten von CSS um Schleifen und aus Programmiersprachen bekannte Features, die Code leichter wartbar machen und weniger Tipparbeit bedürfen. So können Sie mit Präprozessoren Code-Schnipsel wiederverwenden oder Selektoren verschachteln. Für viele Entwickler war einer der Hauptgründe, Präprozessoren zu verwenden, die Möglichkeit, CSS um Variablen zu erweitern. Diese Option bietet CSS mittlerweile mit den Custom Properties jedoch selbst (siehe Abschnitt 11.6).

Die so entstandenen Dateien können Browser nicht verarbeiten, daher müssen diese erst in normales CSS übersetzt werden.

Darüber hinaus gibt es eine ganze Reihe von Tools speziell für die Arbeit mit den großen Präprozessoren: Browser-Plug-ins, Editor-Plug-ins, **Grunt**-Plug-ins oder eigenständige Anwendungen wie **CodeKit** für den Mac, die auch gleich den Browser automatisch aktualisieren und andere Aufgaben übernehmen können.

Die bekanntesten Präprozessoren sind **LESS** und **SASS/SCSS**.

Die Verwendung eines Präprozessors allein ist noch kein Konzept, aber der Präprozessor ist oft zusammen mit Coding-Guidelines Teil eines Kooperations- und Code-Strukturierungskonzeptes.

Die nativen CSS-Funktionen sind in mancher Hinsicht mächtiger als die von Präprozessoren (siehe Kapitel „*Inhalte generieren*“).

Postprozessoren

Postprozessoren wollen Ihnen ebenfalls Tipparbeit abnehmen. Auch hierfür gibt es vorgefertigte Regelsätze, beispielsweise zum Minifizieren Ihrer Dateien.

Task-Runner

Da Präprozessoren kein gültiges CSS produzieren und Postprozessoren keine ausführbaren Programme sind (es sind im Grunde nur Regeln, die auf den geschriebenen Code angewendet werden sollen), benötigt man Task-Runner (dt. Aufgaben-Ausführer), welche erst die eigentliche Umwandlung Ihres Codes in ein gewünschtes Endergebnis vornehmen.

Dazu zählt das oben bereits genannte Grunt. Ebenfalls sehr bekannt und weit verbreitet ist **Gulp**. Für Mac gibt es das ebenfalls bereits erwähnte CodeKit mit einer leicht bedienbaren grafischen Oberfläche.

Eine weitere Aufgabe für Task-Runner ist es, Ihnen wiederkehrende, gleichartige (und dadurch automatisierbare) Arbeitsschritte abzunehmen, zum Beispiel CSS-, HTML- und JavaScript-Dateien zu minifizieren und zu zippen.

Minifizierung nennt man das Entfernen von nicht benötigten Zeichen in Dateien. Browser sind weder auf Einrückungen noch auf Zeilenumbrüche oder Leerzeichen angewiesen (abgesehen von wenigen Ausnahmen). Es spricht also nichts dagegen, auf dem Webserver eine Version Ihres CSS bereitzustellen, die nur aus einer einzigen Zeile besteht und aus der sämtliche Kommentare, Tabulatoren und unnötigen Leerzeichen entfernt wurden. Durch das Einstauen der unnötigen Daten wird Ihre Website schneller geladen.

14.2 CSS ist universell

Natürlich kann man mit CSS die Gestaltung von Webseiten technisch umsetzen. Doch was bedeutet das in der Praxis? Hier im Buch haben Sie die wichtigsten und meistbenötigten Eigenschaften von CSS kennengelernt. In Beispielen haben Sie gesehen, wie man HTML-Schnipsel optisch beeinflussen kann.

Allen Beispielen gemeinsam ist die Annahme, dass Sie eine ungestaltete Webseite vor sich haben, beinahe ein Stück weißes Papier, auf das Sie mit CSS bunte Formen zeichnen können.

Es ist aber gar nicht nötig, immer mit einer leeren Seite anzufangen. Oft ist es interessant, in Bestehendes einzugreifen.

Nutzerbedürfnisse umsetzen

Das Aussehen von Webseiten legen die Betreiber fest. Allerdings lässt sich die Kaskade für eigene Formatierungen nutzen.

Egal, ob Sie Webseiten besser lesbar darstellen wollen oder ihnen ein Aussehen nach Ihrem Geschmack oder einem Corporate Design mitgeben wollen: Mit CSS können Sie das Aussehen jeder beliebigen Webseite im Web anpassen.

Blendempfindliche Menschen können beispielsweise einen grauen Ton als Hintergrund festlegen, um nicht ständig mit dem allgegenwärtigen, großflächigen Weiß konfrontiert zu werden.

Das funktioniert, weil die CSS-Kaskade vorsieht, dass Nutzer-Stylesheets zuletzt ausgewertet werden. Alle vorherigen Angaben lassen sich so überschreiben.

Die Reihenfolge der verwendeten Stylesheets sieht wie folgt aus:

- ✓ Browser-Stylesheet (vom Hersteller mitgeliefert)
- ✓ Autoren-Stylesheet (vom Webseitenbetreiber erstellt)
- ✓ Nutzer-Stylesheet (vom Benutzer angelegt)

Nutzer-Stylesheets können einfach auf der Festplatte abgelegt werden. Abhängig vom verwendeten Browser gibt es verschiedene Möglichkeiten, sie einzubinden. Am komfortabelsten lösen das in der Regel diverse hierfür verfügbare Plug-ins, die auch die seitenspezifische Verwendung von Formatvorlagen erlauben.

Es führt an dieser Stelle zu weit, auf die unterschiedlichen Plug-ins einzugehen, zudem sich deren Verfügbarkeit und Funktionsumfang häufig ändern. Für Firefox gibt es beispielsweise Custom Style Script, mit dem sich eigenes CSS und JavaScript komfortabel auf unterschiedlichen Webseiten nutzen lassen. Wie lange und wie gut dieses derzeit in Version 0.1.4 erhältliche Tool weiter gepflegt wird, lässt sich nicht vorhersagen.

Dennoch kann die Hervorhebung von Elementen in häufig benutzten Web-Apps die eigene Produktivität erhöhen.

Für große, viel besuchte Webseiten wie Google, Facebook, Wikipedia und andere gibt es auch bereits eine große Anzahl an alternativen Layouts.

14.3 CSS ist individuell

Mit CSS haben Sie ein ideales Werkzeug, um das Web und diverse Anwendungen an Ihre persönlichen Bedürfnisse anzupassen.

Immer mehr Programme stehen im Web als Software als Service zur Verfügung. Web-Apps und sogar Apps für Smartphones lassen sich auf Basis von HTML, CSS und JavaScript umsetzen. Meta Platforms, Inc. bietet mit **React Native** ein komfortables Framework hierfür an. Selbst Desktop-Apps wie der bereits genannte Editor **Brackets** sind reine HTML/CSS/JavaScript-Anwendungen.

Remark

Remark.js ist eine Open-Source- Präsentations-App. Mit Markdown oder HTML lassen sich schnell Inhalte erstellen, die Sie mittels CSS an Ihre Bedürfnisse anpassen können.

Die Software besteht komplett aus HTML, CSS und JavaScript und läuft in jedem aktuellen Webbrowser. Ein Präsentationsmodus (wo der Vorführende Informationen sieht, die den Zuschauern verborgen bleiben), Überblendeffekte für Folien und andere wichtige, grundlegende Elemente sind vorhanden.

Mit eigenem CSS lässt sich das Aussehen vollständig an eigene Bedürfnisse anpassen.

Mehr dazu unter <https://remarkjs.com/>

Fertige Layouts individualisieren

Egal, ob Sie **WordPress**, **Drupal**, **TYPO3** oder ein anderes Content-Management-System für eine Webseite einsetzen: Am Ende wird immer HTML generiert, dessen Aussehen Sie mit CSS beeinflussen können.

Mit einem fertigen Layout erstellen Sie in kürzester Zeit eine ansehnliche Website, die Sie an die eigene Corporate Identity oder Layout-Vorstellungen anpassen.

Besonders viele Layouts stehen für WordPress zur Verfügung. Es gibt aber auch Vorlagen für viele andere Content-Management-Systeme oder statische HTML-Seiten.

Eine Websuche nach *HTML-Templates* oder *Website-Layouts*, *WordPress-Themes* oder Ähnliches wird Ihnen zahlreiche Ergebnisse liefern.

Bei der Auswahl eines fertigen Layouts sollten Sie darauf achten, eine möglichst aktuelle und gut gepflegte Vorlage zu wählen.

Weil es so viele frei verfügbare Layouts gibt, ist es unwahrscheinlich, dass Besucher Ihrer Webseite das Layout kürzlich woanders gesehen haben. Daher wird Ihre Webseite nach eigenen Anpassungen am CSS trotz der Verwendung einer vorgefertigten Basis individuell und einzigartig wirken.

Kostenpflichtige Layouts für Preise im mittleren zweistelligen Bereich werden weitaus weniger verwendet als Gratis-Templates. Für 30 bis 60 EUR können Sie also Layouts erwerben, die vergleichsweise selten anzutreffen sind. Diese bieten auch von sich aus oft zahlreiche Individualisierungsoptionen und außerdem in aller Regel Hilfe bei der Ersteinrichtung, Problemen oder Bugs.

14.4 Wo Sie sich weiterbilden können

Social Media und Videoportale

YouTube enthält zahlreiche Lehrvideos. Google selbst stellt für Webmaster umfangreiche Informationen zur Suchmaschinenoptimierung bereit. Von den nachfolgend genannten Quellen (Entwickler, Konferenzen, große Websites ...) werden ebenfalls zahlreiche Videos zu Themen rund um das Web bereitgestellt. Ähnliches gilt für **Vimeo**.

Weitere Anlaufstellen sind diverse **Technik-Podcasts** aus dem Webumfeld.

In sozialen Netzwerken wie **XING**, **Facebook** oder **LinkedIn** finden sich viele Gruppen, die sich mit dem Thema Webdesign oder -entwicklung beschäftigen.

Dokumentations-Webseiten

Im Web gibt es zahlreiche Webseiten, auf denen man gute Dokumentationen findet. Auf einige von ihnen wird auch in diesem Buch gelegentlich verwiesen. Zu den größten und bekanntesten im deutschsprachigen Raum gehört **selfHTML** (<https://selfhtml.org>).

In englischer Sprache gibt es weitere Angebote wie <https://www.w3schools.com> oder **MDN**, das **Mozilla Developer Network** (<https://developer.mozilla.org/>).

Offizieller Standard

Die Primärquellen sollten Sie sich natürlich unbedingt einmal anschauen (CSS <https://www.w3.org/Style/CSS/specs> und HTML5 <https://www.w3.org/TR/html5/>).

Jeder technische Begriff hat eine eindeutige Bedeutung, die bei der Lektüre klar wird. Da es sehr viele solcher Begriffe gibt, dauert die Eingewöhnung eine ganze Weile. Es liegt also nicht an Ihnen, wenn es Ihnen zu Beginn schwerfällt, die Texte zu verstehen. Die Erläuterungen unter den anderen hier genannten Quellen können dabei als „Übersetzungshilfe“ dienen.

Entwickler und Dozenten

Eine ganze Reihe bekannter und hervorragender Entwickler geben zahllose Anregungen und Beispiele für herausragende technische Lösungen. Diese in einem konkreten Problemfall zu finden, kann mühsam sein, da es sich häufig um (unsortierte) Blogeinträge oder Gastbeiträge auf anderen Webseiten handelt. Wenn Sie Lösungen finden, die Ihnen gefallen, sollten Sie diese in einer kommentierten Linkssammlung (z. B. in einer Notizen-App) sammeln und sortieren.

Dort können Sie dann leicht veraltete Informationen auch wieder entfernen.

- ✓ Rachel Andrew (CSS-Grid by example)
- ✓ Lea Verou
- ✓ Sven Wohlfahrt
- ✓ Heydon Pickering
- ✓ Paul Irish
- ✓ Brad Frost
- ✓ Chris Coyier
- ✓ ...

Die Liste lässt sich noch lange fortsetzen. Wenn Sie nach Lösungen für Frontend-Probleme suchen, werden Sie immer wieder auf Namen stoßen, die Ihnen bereits begegnet sind.

Auch diese können Sie sich in der Notizen-App sichern und mit Anmerkungen zu den Personen und Ihren Fachgebieten versehen.

Konferenzen und persönliche Kontakte

Konferenzen sind eine ausgezeichnete Möglichkeit, sich nicht nur weiterzubilden, sondern auch die oben genannten Personen und viele andere Entwickler – zum Beispiel aus Ihrer Umgebung – kennenzulernen.

Austausch ist wichtig! Nichts ist schädlicher, als einen schlechten Ansatz stetig weiterzuentwickeln, bis er irgendwann einigermaßen funktioniert. Solche Lösungen sind meist umständlich, fehleranfällig und nicht zukunftssicher.

Der Dialog mit anderen Entwicklern hilft, die eigene Arbeitsweise und die eigenen Ansichten mit denen von anderen abzugleichen und Ideen für elegantere Lösungen gemeinsam zu entwickeln, auf die keiner der Dialogteilnehmer alleine gekommen wäre.

Lernen Sie, mit Kritik umzugehen und diese anzunehmen! Das wird Ihnen Ihr Leben erheblich erleichtern.

Nebenher können Sie auf Konferenzen natürlich auch Vorträge besuchen und etwas lernen. Die Vorträge selbst finden Sie aber in aller Regel auch auf den einschlägigen Videoportalen.

Außer auf Konferenzen können Sie Entwickler auf Web-Mondays, Treffen von Social-Media-Gruppen oder Bootcamps persönlich kennenlernen.

Websites

Natürlich finden Sie neben den oben genannten Seiten und Personen zahlreiche Seiten im Netz, die sich mit Themen wie Design, Suchmaschinenoptimierung, HTML, CSS und anderen verwandten Themen beschäftigen.

- ✓ CSS-Tricks (<https://css-tricks.com>)
- ✓ Smashing Magazine (<https://www.smashingmagazine.com>)
- ✓ t3n (Cheatsheet-Sammlung unter <http://t3n.de/news/10-cheat-sheets-tipps-tricks-designer-743029/>)
- ✓ a list apart (<https://alistapart.com>)

Zwar lässt sich die Liste lange fortsetzen, doch mit diesen Seiten haben Sie gute Einstiegspunkte mit vielen weiterführenden Links. Diese wenigen Namen sollen Ihnen nur den Einstieg in die Online-Recherche erleichtern. Sie werden sicher bald Ihre eigenen Favoriten haben, auf denen Sie sich regelmäßig über Ihre Lieblingsthemen informieren.

14.5 Sinnvoll weiterbilden

HTML

Die zahlreichen HTML-Elemente haben einen Sinn. Wussten Sie, dass ein `footer` keine Fußzeile ist und daher auch nicht am Seitenende stehen muss – und dass eine Seite mehrere `footer` haben kann?

Wie binden Sie ein Bild in unterschiedlichen Auflösungen ein, sodass Browser automatisch ein kleines Bild für Smartphones und ein großes Bild für die Desktop-Ansicht lädt?

Wenn Sie sicher den Unterschied zwischen Elementen, Tags, Attributen und Befehlen kennen (HTML und CSS haben keine Befehle!), dann ist das eine gute Ausgangsbasis. Wenn Sie bei einer oder mehreren der oberen Fragen nicht ganz sicher sind, sollten Sie sich mit HTML beschäftigen. Denn für eine browserübergreifend korrekte Formatierung mittels CSS ist valides HTML die Grundvoraussetzung. Auch Barrierefreiheit und Benutzerfreundlichkeit hängen stark von der bestimmungsgemäßen Verwendung von HTML ab.

Barrierefreiheit und Usability

Benutzerfreundlichkeit (engl.: usability) bedeutet, dass beim technischen und grafischen Design einer Webseite die Bedürfnisse der Nutzer im Vordergrund stehen. Eng damit verwandt ist die **Barrierefreiheit**¹, weil Webseiten, die auch für Menschen mit Behinderungen zugänglich sind, für alle Menschen besser lesbar und bedienbar sind.

¹ Umfangreiche Informationen zur Barrierefreiheit finden Sie im HERDT-Buch *Das Web barrierefrei gestalten – Leitfaden für Entwickler*.

Dahinter steht auch die Erkenntnis, dass zufriedene Besucher wiederkehren. Wenn Nutzer schnell relevante Informationen finden, sagen sie das weiter.

Suchmaschinenoptimierung (SEO)

Bevor Besucher wiederkehren können, müssen sie Ihre Webseite erst einmal gefunden haben. Damit Ihre Webseite bei den Suchmaschinen auf den vorderen Plätzen gelistet wird, gibt es einiges zu beachten.

Am Anfang aller Überlegungen steht dabei, was die Webseite für Sie erreichen soll. Verfolgen Sie die Weiterentwicklung Ihrer Firma oder möchten Sie möglichst vielen Menschen Informationen anbieten? Sind diese privat oder geschäftlich? Erst, wenn solche grundlegenden Fragen geklärt sind, können Sie sich Gedanken machen, wie Sie Ihre Webseite aufbauen, mit welchen Inhalten Sie diese befüllen. Die Inhalte sind das Wichtigste bei der Suchmaschinenoptimierung, denn Google versucht die „guten“ Webseiten finden.

Dabei kommt es nicht nur auf die Menge der Besucher an, sondern auf die „richtigen“ Besucher. Sie haben sich zunächst ein Ziel überlegt, bei deren Erreichung die Website Sie unterstützen soll. Es gilt also nicht, beliebige Menschen auf Ihre Webseite zu locken, sondern solche, die an dem interessiert sind, was Sie anbieten – ganz gleich, ob das Informationen (z. B. Newsletter), Waren oder Dienstleistungen sind.

JavaScript

Machen Sie Ihre Webseiten interaktiv. Mit JavaScript erstellen Sie HTML-Dokumente, die sich vom Besucher anpassen lassen, Eingaben entgegennehmen, Plausibilitätsprüfungen vornehmen und Daten auf dem Rechner des Nutzers speichern, sodass diese nicht über das Internet übertragen werden müssen. Zahlreiche HTML-Features sind allein für die Programmierung ausgelegt.

Mittels HTML, CSS und JavaScript erstellen Sie sogenannte Rich Internet Applications oder Web-Apps, die wie Programme funktionieren – nur, dass diese im Browser laufen und daher nicht installiert werden müssen.

Berühmte Beispiele sind Google Docs, die sozialen Netze oder Webmail-Dienste. Auch Browser-Spiele gehören zu dieser Art Anwendungen.

Geschickt programmiert laufen solche Anwendungen auch bei fehlender Internetverbindung. Was der Nutzer erstellt, wird auf seinem Rechner gespeichert und geht nicht verloren. Wenn wieder eine Verbindung zum Internet besteht, können die Daten wieder ausgetauscht werden (z. B. eine Mail wird versendet), ohne dass der Nutzer nochmals eingreifen muss.

Symbole		E	
!important	15	Blockssatz	68
%	60	border	126
@media	17	border-bottom	127
<fieldset>	208	border-collapse	148
<label>	208	border-color	123
<legend>	208	border-image	131
		border-left	127
		border-radius	128
		border-right	127
		border-spacing	148
		border-style	122
		border-top	127
		border-width	122
		bottom	154
		box-sizing	137
		Breite, Element	157, 160
		Buchstabenabstand	79
		Buchstabenweite	79
		Buttons	214
		Ebenen anlegen	169
		Ebenen, Reihenfolge	170
		Ecken, abgerundete	128
		Editor	225
		Einbinden, Dokumentenkopf	10
		Einbinden, Import	12
		Einbindung, externe	11
		Eingabefeld, Hintergrund	208
		Einheit	59
		Element, gefloatetes	165
		Elementbreite	157, 160
		Elementhöhe	158
		Element-Selektor	27
		Elternelement	24, 63
		em	59
		E-Mail-Feld	218
		empty-cells	149
		ex	59
A		C	
Abstand, Rahmen	134, 138	calc	181
active	42	caption-side	147
after	45	ch	60
all	16	charset	7
alternate stylesheet	19	Checkboxen	222
Alternative	19	clear	165
Anführungszeichen	173	cm	59
aspect-ratio	18	color	89, 91
Attribut-Selektor	37	content	176, 178
Aufzählung, Kurzform	119	counter	178
Aufzählungszeichen	115	counter-increment	179
Ausgabemedium	15	counter-reset	179
Ausrichten, Hintergrundgrafik	100	CSS importieren	12
Ausrichten, horizontal	68	CSS, Vorteile	7
Ausrichten, Überschrift	147	CSS-Grid	200, 205
Ausrichten, vertikal	70	CSS-Variablen	182
Ausschluss	78	cursor	183
Aussehen, Rahmen	122	Custom Properties	182
Auswahlboxen	212		
Außenabstand	138	D	
		Datumsfeld	218
		Deklaration	23
		display	143
		doctype	7
		Druckausgabe	15
		Durchschuss	76
		E	
		Gefloatetes Element	165
		Gemeine	57
		Geschwister-Selektor	30
		GitHub	226
B		F	
background-attachment	103	Farben, websichere	90
background-clip	109	Farbiges Kombinationsfeld	212
background-color	93	Farbnamen	90
background-image	97	Farbverlauf	207, 211
background-origin	107	Farbwähler	218
background-position	100	fieldset	210
background-repeat	98	filter	110
background-size	105, 106, 107	first-letter	44
Backup	226	first-line	44
before	45	first-of-type	50
Bild, Hintergrund	97	Flexbox	192, 204
Bild, Schaltfläche	214	float	165, 205
Bildhelligkeit	110	focus	42
Bildkontrast	110	font-face	54
Bildsättigung	110	font-family	52
Bildschirmausgabe	15	font-size	60
		font-style	56
		font-variant	57
		font-weight	56
		Formatvorlagen	6, 8, 9
		Formulare, Eingabefelder	207
D		G	
		Gefloatetes Element	165
		Gemeine	57
		Geschwister-Selektor	30
		GitHub	226

Grafik für Listen	118	Link unterstreichen	74	P		
Grafik, Hintergrund	97	Link-Pseudoklasse	41			
Graustufen	110	Linksbündig	68			
Grundfarben	88	Listentyp	115			
Grundgerüst	10	list-style	119			
		list-style-image	118			
		list-style-position	118			
		list-style-type	115			
H						
height	18, 158	margin	138	M		
Hexadezimalsystem	89	margin-bottom	139			
Hintergrund positionieren	100	margin-left	139			
Hintergrund wiederholen	98	margin-right	139			
Hintergrundfarben	93	margin-top	139			
Hintergrundgrafik	97	Mauszeiger	183			
Höhe, Element	158	max-height	159			
hover	42	Maximalbreite, Element	159			
		max-width	159			
I		media	17			
ID-Selektor	34	Media Features	17			
in	59	Media-Queries	17			
Inhalt	176	Medienmerkmale	17			
inherit	63	Menü	196			
inline-block	205	Mindestbreite, Element	159			
Innenabstand	134	Mindesthöhe, Element	159			
		min-height	159			
K		Minifizierung	228			
Kapitälchen	57	min-width	159			
Kaskadierung	13	Mischfarbe	88			
Kindelement	24, 191	mm	59			
Kind-Selektor	29					
Klasse, Schreibweise	32	N				
Klassenname	32	Nachfahren-Selektor	27			
Klassen-Selektor	31	Namenskonventionen	32			
Klassifizierung	143	not	48			
Kombinator	26	nth-child	48			
Kommentar	51	nth-last-child	49			
Konturen	127	nth-of-type	50			
L						
lang	47	O				
Langform	130	only-child	50			
last-child	50	orientation	18			
Laufweite	79	outline	128			
left	154	outline-color	128			
legend	210	outline-style	128			
linear-gradient	211	outline-width	128			
line-height	76	overflow	162			
link	41					
P						
padding	134	Schaltflächen verändern	214			
padding-bottom	134	Schatteneffekt	76			
padding-left	134	Schrift, Kurzform	65, 109			
padding-right	134	Schriftart	52			
padding-top	134	Schriften, Liste	54			
pc	59	Schriftgröße	60			
position	153, 205	Schriftschnitt	56			
Positionieren, Hintergrund	100	Schriftstärke	56			
Positionierung	153	Schriftstil	56			
Postprozessoren	227					
Präprozessoren	227					
print	16					
Pseudelement	39, 44					
Pseudoklasse	39, 40					
Pseudoklasse, dynamische	42					
pt	59					
px	59					
Q						
quotes	173					
R						
Radiobuttons	222					
Radius	128					
Rahmen	127					
Rahmen mit Bildern	131					
Rahmen, Farbe	123					
Rahmen, Tabelle	148					
Rahmenabstand	138					
Rahmengrafik	131					
Rangfolge, Stylesheets	13					
Rechtsbündig	68					
rem	59					
Responsive Website Design	18					
right	154					
S						
Schaltflächen verändern	214					
Schatteneffekt	76					
Schrift, Kurzform	65, 109					
Schriftart	52					
Schriften, Liste	54					
Schriftgröße	60					
Schriftschnitt	56					
Schriftstärke	56					
Schriftstil	56					

screen	16	text-overflow	163	white-space	83
Seitenverhältnis	160	text-shadow	76	width	18, 137, 157, 160
Selektor	23	text-transform	72	Wiederholen,	
Selektor, Attribut	37	Textumbruch	83	Hintergrundgrafik	98
Selektor, benachbarter	30	top	154	Wortabstand	78
Selektor, Element	27	Transparenz	110	Wortzwischenraum	78
Selektor, Geschwister	30	Transparent, Element	167		
Selektor, ID	34				
Selektor, Kind	29				
Selektor, Klasse	31				
Selektor, Nachfahre	27				
Sepia-Ton	110	Überlauf	162	Zahlenfeld	218
Serifen	53	Universal-Selektor	26	Zähler	178
Sichtbarkeit	167	Unsichtbar, Element	167	Zeichenabstand	79
Stapelreihenfolge	170	Unterbrechen, Textfluss	165	Zeichensatz	8
Stil, Rahmen	122	Unterstreichen, Link	74	Zeilenumbruch	83
style	10, 13			Zellen, Breite	149
Stylesheet einbinden	10			Zellen, leere	149
Stylesheet, Alternativen	19			Zentriert	68
Stylesheet, Aufbau	23			z-index	170
				Zweispalten-Layout	188
				Zwischenraum, Tabelle	148
				Zwischenraum, Wort	78
T					
table-layout	149	Variablen	182		
target	48	Vererbung	25		
Text, Überlauf	163	Versalie	57		
Textabstand	78, 79	Viewport	154		
text-align	68	visibility	167		
textarea	210	visited	41		
text-decoration	74				
Textfluss unterbrechen	165				
text-indent	80				
		W			
		Webseite, unterschiedliche			
		Ausgabegeräte	18		
		Weichzeichnung	110		
		Weißenraum	138		

Impressum

Matchcode: CSS3_2024

Autoren: Isolde Kommer, Marc Haunschild

1. Ausgabe, Januar 2024

HERDT-Verlag für Bildungsmedien GmbH
Uwe-Zeidler-Ring 12
55294 Bodenheim
Internet: www.herdt.com
E-Mail: info@herdt.com

© HERDT-Verlag für Bildungsmedien GmbH, Bodenheim

Druck und Bindearbeiten: Esser printSolutions GmbH, D-75015 Bretten
Edubook AG, CH-5634 Merenschwand

Alle Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren) ohne schriftliche Genehmigung des Verlags reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Dieses Buch wurde mit großer Sorgfalt erstellt und geprüft. Trotzdem können Fehler nicht vollkommen ausgeschlossen werden. Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Wenn nicht explizit an anderer Stelle des Werkes aufgeführt, liegen die Copyrights an allen Screenshots beim HERDT-Verlag. Sollte es trotz intensiver Recherche nicht gelungen sein, alle weiteren Rechteinhaber der verwendeten Quellen und Abbildungen zu finden, bitten wir um kurze Nachricht an die Redaktion.

Aus Gründen der besseren Lesbarkeit wird auf die gleichzeitige Verwendung der Sprachformen männlich, weiblich und divers (m/w/d) verzichtet. Sämtliche Personenbezeichnungen gelten gleichermaßen für alle Geschlechter.

Die in diesem Buch und in den abgebildeten bzw. zum Download angebotenen Dateien genannten Personen und Organisationen, Adress- und Telekommunikationsangaben, Bankverbindungen etc. sind frei erfunden. Eventuelle Übereinstimmungen oder Ähnlichkeiten sind unbeabsichtigt und rein zufällig.

Die Bildungsmedien des HERDT-Verlags enthalten Verweise auf Webseiten Dritter. Diese Webseiten unterliegen der Haftung der jeweiligen Betreiber, wir haben keinerlei Einfluss auf die Gestaltung und die Inhalte dieser Webseiten. Bei der Bucherstellung haben wir die fremden Inhalte daraufhin überprüft, ob etwaige Rechtsverstöße bestehen. Zu diesem Zeitpunkt waren keine Rechtsverstöße ersichtlich. Wir werden bei Kenntnis von Rechtsverstößen jedoch umgehend die entsprechenden Internetadressen aus dem Buch entfernen.

Die in den Bildungsmedien des HERDT-Verlags vorhandenen Internetadressen, Screenshots, Bezeichnungen bzw. Beschreibungen und Funktionen waren zum Zeitpunkt der Erstellung der jeweiligen Produkte aktuell und gültig. Sollten Sie die Webseiten nicht mehr unter den angegebenen Adressen finden, sind diese eventuell inzwischen komplett aus dem Internet genommen worden oder unter einer neuen Adresse zu finden. Sollten im vorliegenden Produkt vorhandene Screenshots, Bezeichnungen bzw. Beschreibungen und Funktionen nicht mehr der beschriebenen Software entsprechen, hat der Hersteller der jeweiligen Software nach Drucklegung Änderungen vorgenommen oder vorhandene Funktionen geändert oder entfernt.