

Die Sprache CSS3 im Detail

3IB

Matthes Würbs

1822873

Inhaltsverzeichnis

1 Grundlagen.....	2
1.1 Was ist CSS?.....	2
1.2 Responsives Webdesign.....	2
1.3 Kaskadierung.....	2
1.4 Vorbereitung im HTML-Code.....	3
1.5 Einbinden von CSS.....	3
1.6 Farben.....	4
1.7 Maßangaben.....	4
2 Aufbau von CSS-Dokumenten.....	6
2.1 Das Grundgerüst.....	6
2.2 Selektoren.....	6
2.2.1 Einfache Selektoren.....	6
2.2.2 Kombinierte Selektoren.....	7
2.2.3 Pseudoklassen Selektoren.....	7
2.2.4 Pseudoelement Selektoren.....	8
2.2.5 Attributs Selektoren.....	8
2.3 CSS-Shorthand.....	8
3 Das Box-Modell.....	10
3.1 Allgemeines.....	10
3.2 Margin, Border und Padding.....	10
3.3 Overflow.....	11
4 Text und Textformatierung.....	12
4.1 Schriftarten.....	12
4.2 Textdekoration.....	13
4.3 Textlayout.....	14
5 Layout.....	16
5.1 Position.....	16
5.1.1 Der z-Index.....	17
5.2 Float und Clear.....	17
5.2.1 Der Clearfix-Hack.....	18
A Lösungen der Aufgaben zur Selbstprüfung.....	20
B Literaturverzeichnis.....	22
C Abbildungsverzeichnis.....	23
D Tabellenverzeichnis.....	23

1 Grundlagen

Ziel dieses Kapitels ist es, Sie mit den Grundlagen von CSS vertraut zu machen. Nach diesem Kapitel sollten Sie in der Lage sein,

- den Nutzen von CSS zu sehen
- Responsives Webdesign zu verstehen
- CSS in ein HTML-Dokument einzubinden
- Farben und Maßangaben zu benutzen
- Kaskadierung zu verstehen

1.1 Was ist CSS?

CSS, kurz für Cascading Style Sheets, ist eine Sprache, um HTML-Seiten gestalterisch zu bearbeiten. Die Hauptidee hinter CSS war, Inhalte von Gestaltung zu trennen. Der Vorteil hiervon liegt darin, das Stylesheet nur einmal schreiben zu müssen und es trotzdem auf verschiedene HTML-Dokumente anwenden zu können. Es erlaubt die Anpassung von allen möglichen inhaltlichen Elementen wie Farben, Layout oder Text. Die Dateierweiterung von CSS-Dateien lautet .css.

1.2 Responsives Webdesign

Das Responsive Webdesign ist eine Vorgehensweise beim Designen von Websites. Nach dieser Vorgehensweise ist es das Ziel, „Websites flexibel und reaktionsfähig zu gestalten.“ [Zil12, S.8] Dabei wird nicht nur die Ausrichtung der verschiedenen Endgeräte beachtet, die in Abb. 1.1 exemplarisch dargestellt werden, sondern auch die Auflösung und Eingabemöglichkeiten dieser. Um dies zu vereinfachen, bietet CSS verschiedene Möglichkeiten. Beispielsweise kann man für verschiedene Endgeräte verschiedene Stylesheets erstellen, welche dasselbe HTML-Dokument stylen. Mehr dazu im Punkt „Einbinden von CSS“. Außerdem gibt es in CSS verschiedene skalierbare Maßeinheiten, welche das Anpassen an verschiedenen Bildschirmgrößen vereinfachen.



Abb. 1.1: Responsives Webdesign

https://upload.wikimedia.org/wikipedia/commons/e/e2/Responsive_Web_Design.png [2019-11-08]

1.3 Kaskadierung

Grundsätzlich bezeichnet Kaskadierung das Aneinanderreihen verschiedener Elemente. Im Fall von CSS funktioniert Kaskadierung so, dass jeder Regel eine bestimmte Wichtigkeit zugeordnet wird. Die Regel, welche für ein Element die Höchste Wichtigkeit hat, bestimmt den Stil für dieses Element, vgl. [BB16, S29]. Sind zwei Regeln gleich wichtig, wird der Stil der Regel angewendet, die später festgelegt wird, wie in Abb. 1.2 dargelegt. Auch der Browser des Nutzers hat (sehr grundlegende) Regeln, diese sind jedoch am wenigstens wichtig, vgl. [HB11, S.81]

```
1  .div1 {  
2    height: 50px;  
3    width: 50px;  
4    background: red;  
5  }  
6  
7  .div1 {  
8    height: 50px;  
9    width: 50px;  
10   background: green;  
11 }
```



Abb. 1.2: Der weiter unten definierte Wert wird angenommen

1.4 Vorbereitung im HTML-Code

Grundsätzlich kann man für jedes HTML-Element einen Style angeben. Da dies jedoch nur sehr grobes Styling ermöglicht, sollte man seinen HTML-Code mit verschiedenen HTML-Tags strukturieren, um diesen genauer stylen zu können. Will man einen kompletten Block stylen, bietet sich beispielsweise der `<div>...</div>` Tag an, falls es sich nur um einen Abschnitt in einem Paragraph (`<p>...</p>`) handelt, kann man diesen mit `...` umschließen. Um verschiedene Tags der gleichen Art unterscheiden zu können, kann man ihnen mit dem `class=""`-Attribut eine Klasse zuteilen, oder man teilt ihnen mit dem `id=""`-Attribut einen eindeutigen Identifikator zu. Letzteres sollte man nur machen, wenn man nur genau ein Element stylen will. Hat man mehrere Elemente, die man gleich stylen will, sollte man ihnen dieselbe Klasse zuweisen.

IDs und Klassennamen können nicht mit Zahlen beginnen, diese aber enthalten. Mehrere Wörter sollten durch Bindestriche getrennt werden, camelCase ist nicht angebracht. Will man eine Variation eines Elements oder einer Klasse benennen, bieten sich zwei Bindestriche an (zB `btn-warning`), für Kinder eines Elements zwei Unterstriche (zB `btn__text`)

1.5 Einbinden von CSS

Um CSS in ein HTML-Dokument einzubinden, gibt es mehrere Möglichkeiten. Am direktesten ist es, das `style=""`-Attribut zu benutzen, welches man auf die meisten HTML-Tags anwenden kann. Der so genannte „**Inline-Style**“ hat die größte Gewichtung, überschreibt also alle anderen Stylevorgaben, jedoch ist diese Art des Stylings zu vermeiden da man nur ein einziges Element (und eventuelle Kinder) per `style`-Attribut stylen kann. Somit müsste man das für jedes andere Element in jedem anderem Dokument wiederholen. Außerdem erschwert es die Lesbarkeit deutlich, da man die Stylevorgaben nicht zentral an einem Platz hat.

Die nächste Möglichkeit ist, den `<style>...</style>` Tag im Header der HTML-Datei zu benutzen dies vereinfacht die Lesbarkeit zwar extrem, jedoch muss man den Tag trotzdem für jede Datei, welche auf diese Art gestyled werden soll, anlegen. Somit eignet sich diese Methode für Websites mit mehreren HTML-Dokumenten ebenfalls nicht.

Vorzugsweise nutzt man also diese Methode: Das Einbinden eines Stylesheets im HTML-Header. Dies erlaubt es nicht nur, dasselbe Sheet für verschiedene Dokumente zu nutzen, man kann sogar verschiedene Stylesheets für verschiedene Endgeräte nutzen. Hierfür nutzt man das Standalone Tag `<link/>`. In diesem Tag setzt man mit dem Attribut `rel=""` die Beziehung, man schreibt also `rel="stylesheet"`. Zusätzlich benötigt wird auch der `href=""`-Tag, dem man den Link des gewünschten Stylesheets übergibt, beispielsweise `href="/stylesheet.css"`. Neben diesen beiden Pflicht-Attributen gibt es noch das `media=""` und das `title=""`-Attribut. Mit dem `media=""`-Attribut lassen sich verschieden Styles für verschiedene Endgeräte festlegen:

```
<link rel="stylesheet" href="smartphone.css" media="handheld"/>
```

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






Verschiedene `media=""`-Werte sind zum Beispiel **screen** (Standartwert, für normale Computerbildschirme), **handheld** (für PDAs und Smartphones) oder **print** (für die Druckversion der Seite). Das `title=""`-Attribut erlaubt es, verschiedene Styles für die gleiche Seite auf dem gleichen Medium anzulegen. Die so definierten Styles lassen sich im Browser wechseln, auf Firefox beispielsweise mit einem Klick auf Ansicht > Webseiten-Stil.

1.6 Farben

Mittels CSS lassen sich HTML-Elemente einfärben. Die gewünschte Farbe kann man auf verschiedenen Wegen definieren. Der einfachste Weg sind Namen. Für viele vordefinierte Farben wurden Namen vergeben, die man einfach und ohne Anführungszeichen angeben kann. Beispiele hierfür sind „simple“ Farben wie „Red“, „Blue“ oder „Green“, es gibt aber auch „verstecktere“ Farbnamen wie „GoldenRod“, oder „DarkSeaGreen“, vgl [BB16, S.102].

Zusätzlich zu den Namen kann man unter anderem auch noch Hex-Farben, RGB-Farben und HSL-Farben angeben. Anschließend folgt eine Tabelle, welche oben benannte Werte in den verschiedenen Schreibweisen vergleicht.

Tab. 1.1: Vergleich verschiedener Farben und ihrer Schreibweisen

Name	Hex-Wert	RGB-Wert	HSL-Wert	Ergebnis
Red	#ff0000	rgb(255, 0, 0)	hsl(0, 100%, 50%)	
Blue	#0000ff	rgb(0, 0, 255)	hsl(240, 100%, 50%)	
Green	#00ff00	rgb(0, 255, 0)	hsl(120, 100%, 50%)	
GoldenRod	#DAA520	rgb(218, 165, 32)	hsl(43, 74%, 49%)	
DarkSeaGreen	#8fbc8f	rgb(143, 188, 143)	hsl(120, 25%, 65%)	

Grundsätzlich kann man die Farbnamen auch alle klein schreiben.

1.7 Maßangaben

Wie oben schon erwähnt, gibt es in CSS Maßangaben, um Abstände, Größen und Ähnliches festzulegen. In CSS wird unterschieden zwischen **relativen** Längen und **absoluten** Längen.

Tab. 1.2: Relative Längeneinheiten

Einheit	Beschreibung
em	Abhängig von der Höhe der gewählten Schriftart. Orientiert sich am großen „M“
ex	Abhängig von der Höhe der gewählten Schriftart. Orientiert sich am kleinen „x“
ch	Abhängig von der Breite der Null „0“
vw	1% von der Breite des Browser Fensters
vh	1% von der Höhe des Browser Fensters
vmin	1% von der kleineren Größe des Fensters
vmax	1% von der größeren Größe des Fensters
%	Angabe in Prozent des Übergeordneten Elements

„Relative Maßeinheiten eignen sich für die Bildschirmdarstellung besser als absolute Einheiten, da sie skalierbar sind“ [HB11, S86]. Eine Ausnahme bildet die px-Einheit, diese bietet sich gut an, um dünne Grenzlinien zu definieren. Wenn die Größe des Ausgabemediums bekannt ist, bieten sich absolute Längeneinheiten an, da sich die Abstände so genauer festlegen lassen. Wenn man beispielsweise ein Stylesheet für eine Druckversion einer Seite anlegt, weiß man, wie groß das Papier ist und kann hervorragend absolute Einheiten nutzen.

Tab. 1.3: Absolute Längeneinheiten

Einheit	Länge	Beschreibung
cm	1cm	
mm	1mm	
in	2.54cm	Inch
px*	0,26mm	Pixel, abhängig vom Gerät. Low-DPI: 1px ist ein Pixel des Displays. Drucker und hochauflösende Bildschirme: 1px sind mehrere Display Pixel
pt	0,35mm	Point
pc	0,4cm	Pica

Zusammenfassung

In diesem Kapitel wurde erklärt, was CSS ist und warum man es benutzen sollte. Außerdem wurde beschrieben, wie man sein HTML-Dokument auf Styling mittels CSS vorbereiten kann und wie man CSS in sein Dokument einbindet.

Abschließend wurden noch grundlegende Einheiten von CSS definiert und erklärt, wann und wie sie zu benutzen sind.

Aufgaben zur Selbstprüfung

Aufgabe 1:
Was ist CSS?

Aufgabe 2:
Was versteht man unter responsivem Webdesign?

Aufgabe 3:
Was macht die Kaskadierung in CSS so nützlich?

Aufgabe 4:
Weshalb sollte man auf inline-style verzichten?

Aufgabe 5:
Binden Sie das Stylesheet „/handheld.css“ ein, sodass es nur auf Smartphones/PDAs benutzt wird.

2 Aufbau von CSS-Dokumenten

In diesem Kapitel lernen Sie, wie Sie ein CSS-Stylesheet aufbauen und wie sie das gewünschte HTML-Element präzise ansprechen. Außerdem lernen Sie, was man unter CSS-Shorthand versteht und Ihnen werden ein paar grundlegende Eigenschaften von Elementen gezeigt.

2.1 Das Grundgerüst

Ein CSS Stylesheet besteht im Grunde genommen aus drei wiederkehrenden Elementen:

Selektoren, **Eigenschaften** und **Werte**. Der Selektor wählt aus, welches Element zu stylen ist, die Eigenschaft und der Wert, zusammen **Deklaration** genannt, legen fest, wie sich die entsprechende Eigenschaft äußert. Nach dem Selektor kommen, in geschweiften Klammern eingeschlossen die Regeln, jeweils durch Semikolons getrennt. Diese Kombination bezeichnet man als **Regel**, das Bild unter diesem Abschnitt zeigt eine solche. Von diesen Regeln gibt es mehrere in einem Stylesheet.

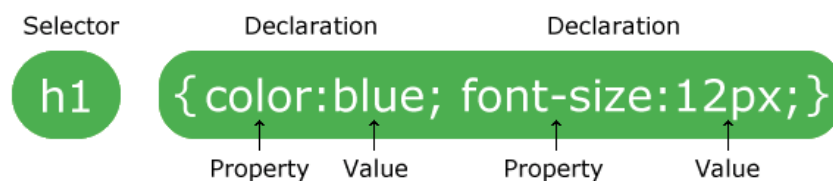


Abb. 2.1: Eine simple CSS Regel

<https://www.w3schools.com/css/selector.gif> [2019-11-08]

2.2 Selektoren

Mit Hilfe der Selektoren lässt sich ganz genau festlegen, auf welches Element man die Regel anwenden möchte. Man unterscheidet zwischen einfachen Selektoren, kombinierten Selektoren, Pseudoklassen und -element Selektoren sowie Attributsselektoren.

2.2.1 Einfache Selektoren

Mittels einfachen Selektoren lassen sich Elemente rein nach ihrem Namen, ID, Klasse oder Typ ansprechen. Dies ist die einfachste Möglichkeit, Elemente auszuwählen, jedoch braucht sie manchmal mehr Aufwand als beispielsweise kombinierte Selektoren. Hat man unterschiedliche Elemente, die man gleich stylen will, kann man sie mit Kommas aneinanderreihen. Eine Klasse spricht man mit einem Punkt vor dem Klassennamen an, zum Beispiel `.my-class`. Dadurch werden alle Elemente mit dem Attribut `class="my-class"` ausgewählt. Will man hingegen ein einzelnes Element auswählen, dem man mittels `id="my-id"` eine ID gegeben hat, nutzt man eine Raute vor der ID, also `#my-id`. HTML-Tags, wie `<p>` oder `<div>` spricht man an, indem man die Klammern weglässt: `p`, `div` `{...}` zum Beispiel würde die Deklarationen auf alle Paragraphen und Divisions anwenden. Außerdem gibt es noch den `*`-Selektor, welcher die Deklaration(en) auf alle Elemente anwendet.

2.2.2 Kombinierte Selektoren

Kombinierte Selektoren bestehen aus zwei oder mehr Selektoren. Durch sogenannte **Kombinatoren** werden die Selektoren verbunden. Der **Nachfahren-Kombinator** besteht schlichtweg aus einem Leerzeichen und selektiert, wie der Name sagt, jeden passenden Nachfahre. So färbt zum Beispiel `div p {color: red;}` jeden Paragraphen, welcher sich in einer Division befindet, gelb ein. Ob sich der Paragraph dabei in einem weiteren Block befindet oder nicht, spielt keine Rolle.

Der **Kind-Kombinator** „>“ wählt nur direkte Kinder aus. Nimmt man obiges Beispiel und ersetzt den Nachfahren-Kombinator durch den Kind-Kombinator (`div>p {color: red;}`), werden nur direkt unterhalb der Division liegende Paragraphen eingefärbt. Falls sich ein Paragraph in einem weiteren Block, bspw. einem Navigations-Block, befinden, werden diese nicht eingefärbt.

Mit dem **Nachbar-Kombinator** „+“ lässt sich der direkte (untere) Nachbar eines Elements wählen, und auch nur, wenn er dem Selektor entspricht. Zu beachten ist, dass die Elemente sich auf der gleichen Ebene befinden müssen. `div+p {color: red;}` wählt den ersten Paragraph nach jeder Division und färbt den Hintergrund gelb. Befindet sich ein Element zwischen einer Division und einem Paragraph, wird der Paragraph nicht eingefärbt!

Der **Geschwister-Kombinator** „~“ hingegen wählt alle folgenden, zum Selektor passenden, Elemente aus, die sich auf der gleichen Ebene befinden. Im Beispiel `div~p {color: red;}` würden also alle Paragraphen nach der ersten Division gefärbt werden, wenn sie sich auf derselben Ebene befinden.

Im folgenden werden alle Kombinatoren beispielhaft gezeigt, Abb. 2.2 enthält den zugrundeliegenden HTML-Code

```
HTML ▼
1 <div>
2   <p>Eins</p>
3   <nav><p>Zwei</p></nav>
4   <p>Drei</p>
5 </div>
6
7 <p>Vier</p>
8 <code>code</code>
9 <p>Fünf</p>
```

Abb. 2.2: HTML-Code für die folgenden Beispiele

```
CSS ▼
1 div p {
2   color: red;
3 }
```

Abb. 2.3: Der Nachfahren-Kombinator

```
CSS ▼
1 div + p {
2   color: red;
3 }
```

Abb. 2.5: Der Nachbar-Kombinator

```
CSS ▼
1 div > p {
2   color: red;
3 }
```

Abb. 2.4: Der Kind-Kombinator

```
CSS ▼
1 div ~ p {
2   color: red;
3 }
```

Abb. 2.6: Der Geschwister-Kombinator

2.2.3 Pseudoklassen Selektoren

Pseudoklassen definieren bestimmte Zustände eines Elements. Ein gutes Beispiel hierfür ist ein einfacher, mittels <a> eingebundener, Link. Dieser Link hat verschiedene Zustände: Unbesucht, besucht, hover (Die Maus befindet sich auf dem Link) sowie aktiv (Der Link wird mit gehaltener Maustaste geklickt). Für diese Zustände kann man dem Link unterschiedliche Eigenschaften zuweisen. Diese Zustände werden angesprochen, indem man `Element:Zustandsname` nutzt. Im Beispiel mit dem Link spricht man den Zustand `hover` folgendermaßen an: `a:hover{Deklarationen}`. Von diesen Pseudoklassen gibt es zu viele, um sie hier alle aufzuzählen, ein paar zusätzliche Beispiele aber sind `input:focus`, welches das aktuell im Fokus liegende <input> Element selektiert, `p:lang(de)`, welches jeden Paragraph auswählt, dessen `lang=""`-Attribut auf „de“ gesetzt ist, oder

p:only-child, das jeden Paragraphen auswählt, der das einzige Kind seines umschließenden Elements ist. Man muss keine HTML-Elemente vor den Doppelpunkt setzen, es können auch Klassen oder mit einer ID versehene Elemente angesteuert werden.

2.2.4 Pseudoelement Selektoren

Pseudoelemente werden benutzt, um verschiedene Teile eines Elements zu stylen. Ähnlich wie Pseudoklassen werden sie mit :: angesteuert. `.my-class::after{content: „Ende Klasse“;}` würde zum Beispiel nach jedes Element der Klasse `my-class` „Ende Klasse“ schreiben. Im Kontrast dazu würde `.my-class::after{content: „Anfang Klasse“;}` „Anfang Klasse“ vor jedes Element der Klasse setzen. Mit `::first-line` beziehungsweise `::first-letter` lassen sich die erste Zeile respektive der erste Buchstabe ansteuern. `::selection` bezieht sich auf den vom User ausgewählten Text/Inhalt. So könnte man mit `*::selection{color: red; background: black;}` festlegen, dass jeglicher auf der Seite ausgewählter Text rot auf schwarzem Hintergrund erscheint.

2.2.5 Attributs Selektoren

Attributs Selektoren selektieren Elemente, welches ein bestimmtes Attribut haben. Man kann auch auswählen, dass das Attribut einen bestimmten Wert haben soll. Den Selektor schreibt man in eine eckige Klammer hinter das gewünschte Element, `a[target]` beispielsweise selektiert jedes anchor-Element, welches extra ein Ziel spezifiziert hat. Attribute mit einem bestimmten Wert lassen sich per `[attribut="wert"]` anwählen. Mittels `[attribut*="wert"]` und `[attribut~="wert"]` lassen sich nach Substrings suchen. Die Unterscheidung der beiden sieht wie folgt aus: Nutzt man `~` muss der gesuchte Substring durch Leerzeichen vom Rest des Strings abgetrennt sein, nutzt man `*` reicht es, wenn der Wert in irgend einer Form enthalten ist. Eine ähnliche Verwandtschaft teilen `[attribut^="wert"]` und `[attribut|=“wert“]`. Beide werden genutzt, um zu prüfen, ob das Attribut mit dem Wert „wert“ beginnt. Benutzt man `|` muss das Präfix durch einen Bindestrich („-“) vom Rest des Attributs getrennt sein, nutzt man `^` spielt das keine Rolle. Die Suche eines Suffix erfolgt mittels `[attribut$="wert"]`. Hier gibt es keine Gegenstück.

Achtung:

Das Trennzeichen für einen Substring ist ein Leerzeichen, bei einem Präfix ist das Trennzeichen ein Bindestrich!

2.3 CSS-Shorthand

CSS erlaubt es, bei Eigenschaften, welche enge Geschwister haben, die Werte in eine Zeile zu schreiben. Beispielsweise kann man, anstatt `p{border-width: 2px; border-style: solid; border-color: black;}` einfach `p{border: 2px solid black;}` schreiben. In diesem Fall spielt die Reihenfolge keine Rolle, da die Angaben alle eindeutig sind. Will man hingegen die padding-Eigenschaft, welche den Innenabstand einer Box zu ihrem Inhalt festlegt, in Shorthand schreiben, muss man die Reihenfolge beachten: `p{padding: 1px 2px 3px 4px;}` bedeutet `p{padding-top: 1px; padding-left: 2px; padding-bottom: 3px; padding-right: 4px;}`. Hier kann es zu Verwechslungen kommen, falls man nicht aufpasst. Das Gleiche gilt für die margin-Eigenschaft. Weitere Eigenschaften, bei denen man diese Schreibweise einsetzen kann, sind `background`, `font` oder auch `animation`.

Zusammenfassung

In diesem Kapitel wurde erklärt, wie .css Dokumente aufgebaut sind, außerdem wurde erklärt, wie man Selektoren, Kombinatoren und Pseudoklassen sowie -elemente und Attributs Selektoren benutzt. Zu allerletzt wurde kurz dargelegt, was man unter CSS-Shorthand versteht.

Aufgaben zur Selbstprüfung

Aufgabe 1:

Benennen Sie alle fünf Elemente: `p { color: red; }`.

Aufgabe 2:

Entwerfen sie eine Regel, die den Hintergrund des Elements mit der `id="first"` gelb färbt (Tipp: Die Hintergrund-Eigenschaft heißt `background`).

Aufgabe 3:

Nennen sie den Selektor, der alle Links (`<a>`) auswählt, die direkte Kinder eines Navigations-Blocks (`<nav>`) sind.

Aufgabe 4:

Welche Elemente werden durch folgenden Selektor ausgewählt: `q:lang(de)`

Aufgabe 5:

Schreiben Sie einen Selektor, der alle Elemente auswählt, deren Klassenname mit „big-“ beginnt auswählt.

3 Das Box-Modell

In diesem Kapitel sollen sie mit dem Boxmodell sowie den dazugehörigen Eigenschaften vertraut gemacht werden. Außerdem lernen sie, wie man die Breite eines Elements korrekt bestimmt und Inhalt an Boxen anpasst.

3.1 Allgemeines

Eine HTML-Seite ist eine Ansammlung von Blöcken, von denen einige auch ineinander verschachtelt sind. Diese Blöcke kann man auch als Boxen betrachten, daher der Name. Das Wort Box-Modell fällt meistens im Zusammenhang mit Design und Layout einer Website. In CSS besteht eine Box aus vier Teilen: Inhalt (Content), Innenabstand (Padding), Rahmen (Border) und Außenabstand (Margin). Das Bild rechts veranschaulicht den Aufbau einer Box. Der Inhalt enthält das entsprechende HTML-Element, Innenabstand, Rahmen und Außenabstand werden durch CSS festgelegt. Die Größe des Inhalts kann ebenfalls per CSS festgelegt werden. Die Breite eines Elements ist folglich nicht nur die Breite des Inhalts, sondern berechnet sich wie folgt:

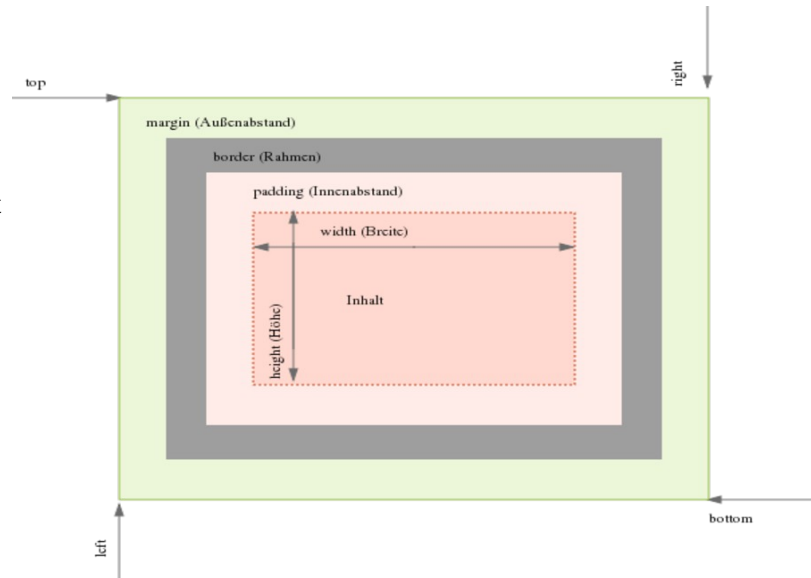


Abb. 3.1: Das Box-Modell

<https://wiki.selfhtml.org/images/thumb/f/f3/Box-Modell.svg/900px-Box-Modell.svg.png> [2019-11-11]

$\text{margin-left} + \text{border-left} + \text{padding-left} + \text{content-width} + \text{padding-right} + \text{border-right} + \text{margin-right}$

Die Berechnung der Höhe verhält sich analog, man muss margin, border und padding sowohl oben als auch mit unten miteinbeziehen.

3.2 Margin, Border und Padding

Der Außenabstand legt den Abstand zum nächsten Element auf der gleichen Ebene fest. Der Bereich bleibt transparent. Mit margin-top, margin-left, margin-bottom und margin-right lassen sich die Abstände für jede Seite einzeln festlegen. Um sich die Schreibarbeit zu sparen kann man hier CSS-Shorthand anwenden: Man schreibt nur noch margin: [] [] [] []; und setzt die gewünschten Werte in der Reihenfolge oben, links, unten, rechts ein. Falls die Werte alle gleich sind, reicht es, margin: []; zu schreiben.

Der Innenabstand legt den Abstand fest, den der Inhalt vom Rahmen hat. Dieser wird analog zum Außenabstand festgelegt, statt margin schreibt man jedoch padding. Auch hier gibt es die Möglichkeit, Shorthand zu nutzen.

Der Rahmen, oder border, eines Elements zieht eine sichtbare Linie um dieses. Man kann die Dicke, die Farbe und den Stil des Rahmens mit border-width, border-color und border-style festlegen. Auch hier gibt es eine Shorthand Funktion, man schreibt einfach border: [] [] []; und trägt die

gewünschten Werte ein. Die Reihenfolge spielt keine Rolle. Will man die verschiedenen Kanten des Rahmens unterschiedlich gestalten, so ist auch dies möglich. Mit `border-[seite]: [] [] [];` lassen sich die Kanten einzeln definieren. Hier wird ebenfalls die Shorthand Schreibweise genutzt.

Die geläufigsten `border-styles` sind `solid`, `dotted`, oder `dashed`. Außerdem lassen sich die Ecken mit `border-radius` abrunden. Weitere `border-styles` mit verschiedenen Radien sind in Abb. 3.2 zu sehen.

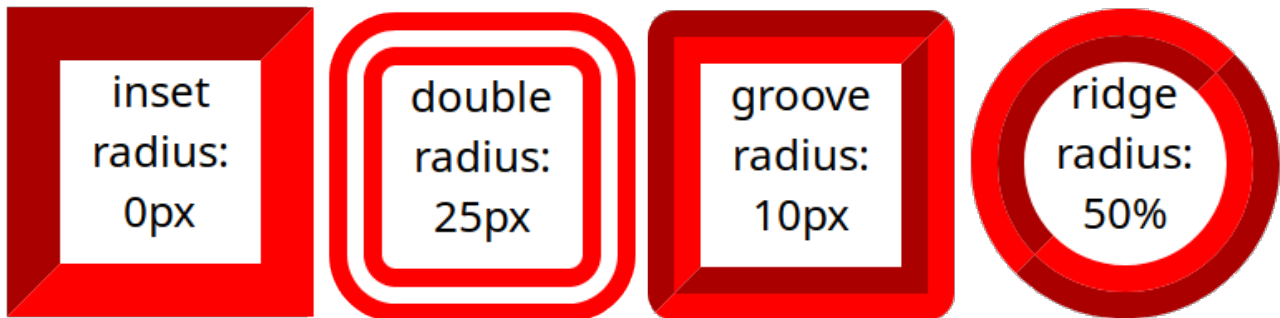


Abb. 3.2: Weitere Rahmenstilen

3.3 Overflow

Mit der `Overflow`-Eigenschaft (dt.: Überfluss) lässt sich festlegen, was mit Inhalt passiert, der über die vorgegebene Breite und Höhe hinaus ragt. Der default-Wert ist `overflow:visible`, dies bedeutet, dass der Text auch über die vorgegebene Größe hinaus sichtbar ist und möglicherweise in andere Elemente hineinragt. Will man dies nicht, so stehen einem `overflow:hidden` und `overflow:scroll` zur Verfügung, welche den Inhalt, welcher zu viel ist, auf verschiedene Arten verstecken.

`overflow:hidden` schneidet das, was an Inhalt zu viel ist, komplett ab. `overflow:scroll` schneidet den Inhalt ebenfalls ab, fügt der Box aber Scrollbars hinzu, mit denen man durch den Inhalt in der Box scrollen kann. Es gibt auch noch einen vierten Wert, `overflow:auto`. Wird dieser Wert genutzt, legt der Browser automatisch fest, welcher der anderen drei Werte angebracht ist. Praktischer ist es aber, den `Overflow` manuell festzulegen um ein konsistentes Aussehen zu garantieren.

Zusammenfassung

Das Ziel dieses kurzen Kapitels war es, Sie mit dem Box-Modell vertraut zu machen und Ihnen darzulegen, wie sie einzelne Elemente ihm Box-Modell richtig anordnen und mit Rahmen versehen. Außerdem wurde beschrieben, wie man mit dem Fall umgeht, wenn der Inhalt einer Box über die vorgesehenen Maße herausragt.

Aufgaben zu Selbstprüfung

Aufgabe 1:

Aus was besteht eine Box im CSS Box-Modell?

Aufgabe 2:

Berechnen sie die Breite des folgenden Elements:
`p {margin: 10px; border: 2px solid black; padding: 5px; width: 50px;}`

Aufgabe 3:

Wie verhindern Sie, dass der Inhalt einer Box über die festgelegte Größe hinausragt?

4 Text und Textformatierung

Ziel dieses Kapitels ist es, Ihnen Schriftarten und deren Familien näher zu bringen und Ihnen verschiedene Möglichkeiten zur Textdekoration zu zeigen. Außerdem wird Ihnen gezeigt, wie Sie das Layout eines Textes bestimmen können.

4.1 Schriftarten

Text ist der zentrale Bestandteil der meisten Websites. Man unterscheidet zwischen vielen unterschiedlichen Elementen wie Überschriften, Navigationselementen oder normalen Textkörpern. Folglich gibt es viele verschiedene Möglichkeiten, ihn zu dekorieren. Das wahrscheinlich wichtigste Werkzeug zur Beeinflussung des Aussehens von Texten sind wohl die Schriftarten (engl. „fonts“).



Abb. 4.1: Veranschaulichung von Serifen

<https://www.w3schools.com/css/serif.gif>
[2019-11-13]

Zunächst unterscheidet man zwischen Schriftarten mit **Serifen** und Schriftarten ohne **Serifen** („**Sans-serif**“). Serifen sind kleine Linien und Haken am Ende mancher Zeichen, in Abb. 4.1 sind diese mit rot markiert. Zusätzlich gibt es noch **Monospace** Schriftarten (Jeder Buchstabe hat genau die gleiche Breite), **kursive** Schriftarten (imitieren Handschriften) und **fantasy** Schriftarten, welche eher als Dekoration genutzt werden sollten. Für die Wiedergabe auf Bildschirmen sind Serifenlose Schriftarten meist besser geeignet, da sie meist leichter zu lesen sind.

Weiterhin unterscheidet man zwischen verschiedenen Familien von Schriftarten. Die bekanntesten sind Arial (sans-serif), Helvetica (sans-serif), Times New Roman (serif) oder auch Georgia (serif). Diese und einige weitere sind nativ in CSS eingebunden und können problemlos genutzt werden. Dies geschieht mittels der **font-family** Eigenschaft, der als Wert der Name der Schriftart übergeben wird (Falls der Name aus mehreren Worten besteht, muss man ihn in Anführungszeichen setzen. Mit `p {font-family: „Times New Roman“;}` wird die Schriftart jedes Paragraphen auf „Times New Roman“ gesetzt (in dieser Schriftart sind übrigens auch die Texte dieses Skripts verfasst). Man kann auch mehrere Schriftarten, durch Kommata getrennt, angeben. Diese weiteren Schriftarten werden genutzt, wenn der Browser die erste Schriftart nicht unterstützt. So macht es Sinn, die weiteren Schriftarten immer allgemeiner zu nennen, zum Beispiel:

```
p {font-family: „Times New Roman“, Times, serif;}
```

Erst versucht der Browser die Schriftart „Times New Roman“ anzuwenden. Falls das nicht klappt, versucht er eine Schriftart aus der Familie „Times“ zu nutzen. Klappt dies auch nicht, wendet er eine Schriftart an, die Serifen enthält.

Natürlich kann man auch die Schriftgröße ändern, hierfür sollte man in den meisten Fällen relative Maße nutzen. Die Standardgröße sind 16px, bzw 1em. Will man eine andere Größe für den Text

nutzt man **font-size** und eine beliebige Größenangabe. Nutzen sollte man em, da dies den Nutzern erlaubt, die Schriftgröße zu vergrößern/verkleinern. Man kann die Textgröße auch an die Größe des Browserfensters anpassen, indem man die Einheit vw nimmt.

Neben der Größe kann man die Schriftarten auch weiter bestimmen. Nutzt man **font-style: italic;** wird der Text kursiv gezeigt. Weitere Werte für diese Eigenschaft sind **normal** und **oblique**, welches im Prinzip sehr ähnlich zu italic ist, aber weniger unterstützt wird.

Zudem kann man mit **font-weight: bold;** festlegen, dass der Text fett gedruckt wird. Alternativ kann man statt bold auch **lighter/bolder** nehmen, dies bezieht sich auf das Elternelement. Es besteht auch die Möglichkeit, ein vielfaches von 100 zu nehmen, der Maximalwert ist 900. Durch die **font-variant** Eigenschaft kann man den Text in „kleinen Großbuchstaben“ darstellen, der dazugehörige Wert ist **small-caps**. Der Standardwert ist **normal**.

4.2 Textdekoration

Neben den verschiedenen Schriftarten gibt es natürlich noch weitere Methoden, um Text anders aussehen zu lassen.

Die Farbe eines Textes lässt sich mit der **color**-Eigenschaft verändern. Als Wert lässt sich hier jeder beliebige Farbwert eintragen, der wie im Kapitel 1.6 – Farben beschrieben formatiert ist. Man muss beachten, dass man, um dem World Wide Web Consortium konform zu sein, eine Hintergrundfarbe angeben muss, falls man die Textfarbe ändert. Dies schafft man mit der **background**-Eigenschaft, der man als Wert ebenfalls jeden Farbe übergeben kann, solange diese den Vorgaben entspricht.

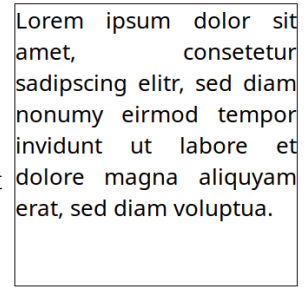
Mit der Eigenschaft **text-decoration-line** lässt sich der Text unterstreichen, überstreichen oder auch durchstreichen. Die Werte hierfür sind **underline**, **overline** und **line-through**. Man kann auch beliebige Kombinationen der drei Werte nutzen, dafür schreibt man sie einfach ohne Komma(!) hintereinander. Will man diese Linien farbig gestalten, so nutzt man die Eigenschaft **text-decoration-color** mit einem Farbwert nach Kapitel 1.6. Mittels **text-decoration-style** lässt sich der Stil der Linien festlegen, als Werte kann man beispielsweise **dotted**, **double** oder **dashed** nehmen. Der Standardwert ist **solid**. Für diese Eigenschaften gibt es eine Shorthand Schreibweise, nämlich **text-decoration**. Dieser kann man alle vorherig genannten Eigenschaften übergeben, außerdem kann man die dicke der Dekoration festlegen, beispielsweise 5px. Tipp: Will man die automatischen Unterstreichungen von Links entfernen, kann man text-decoration: none nutzen.

Mit **text-transform** lässt sich die Groß-/Kleinschreibung eines Textes beeinflussen. Der Wert **uppercase** transformiert alle Buchstaben in Großbuchstaben, **lowercase** bewirkt das Gegenteil und transformiert alle Buchstaben in Kleinbuchstaben. Den ersten Buchstaben eines jeden Wortes in einen Großbuchstaben transformieren kann man mittels **capitalize**.

Man kann seinem Text auch einen Schatten hinzufügen. Dies funktioniert durch die **text-shadow** Eigenschaft, diese benötigt drei Werte: Horizontalen Versatz, Vertikalen Versatz und die Farbe des Schattens. Ein Beispiel hierfür wäre `p {text-shadow: 2px 3px lightgrey;}`.

4.3 Textlayout

Auch für das Textlayout gibt es einige Eigenschaften. Diese erlauben es unter anderem, den Text auszurichten oder den Abstand zwischen Zeichen zu verändern (dies nennt man Kerning). Die Eigenschaft **text-align** erlaubt das Ausrichten von Texten. Mittels den Werten **center**, **left** oder **right** lässt sich der Text an der gewünschten Position platzieren. Ein interessanter Wert hierfür ist auch **justify**. Nutzt man diesen Wert, wird der Abstand zwischen den Worten so eingestellt, dass der Text immer auf beiden Seiten perfekt bündig ist, siehe Abb. 4.2. Will man die erste Zeile eines Textes einrücken, hilft einem **text-indent**. Diese Eigenschaft rückt die erste Zeile soweit ein, wie man in ihrem Wert festgelegt hat. Den Buchstabenabstand ändert man, indem man die Eigenschaft **letter-spacing** benutzt. Als Wert können hier sowohl positive als auch negative Werte übergeben werden. Positive Werte bewirken ein Strecken der Wörter, durch negative Werte hingegen werden sie gestaucht. Analog dazu verhält sich **word-spacing**, welches die Abstände zwischen Wörtern ändert. Auch hier lassen sich positive und negative Werte angeben.



Lorem ipsum dolor sit
amet, consetetur
sadipscing elitr, sed diam
conipsum elitr, sed diam
conipsum elitr, sed diam
conipsum elitr, sed diam
conipsum elitr, sed diam
conipsum elitr, sed diam
conipsum elitr, sed diam
conipsum elitr, sed diam

Abb. 4.2: text-align:
justify;

Die Schreibrichtung eines Textes lässt sich durch **direction** beeinflussen. Die möglichen Werte sind **rtl**, **ltr** und **inherit**. ltr bedeutet left-to-right und ist unsere normale Schreibrichtung und damit der Standardwert. Nutzt man rtl, wird der Text von rechts nach links geschrieben. Inherit heißt, dass die Schreibrichtung vom Elternelement geerbt wird.

Zusammenfassung

In diesem Kapitel wurde zunächst erklärt was es für Schriftarten gibt und worin diese sich hauptsächlich unterscheiden. Zusätzlich wurde dargelegt, wie man die Schriftgröße ändert und was für Eigenschaften die Schriftarten haben. Weiterhin wurden einige Methoden zur Textdekoration erörtert und schlussendlich wurde dargestellt, wie man Text grundlegend formatiert.

Aufgaben zur Selbstprüfung

Aufgabe 1:

Worin besteht der Unterschied der zwei großen Hauptfamilien von Schriftarten?

Aufgabe 2:

Was für eine Schriftart sollten Sie für Inhalte nutzen, die auf einem Computerbildschirm dargestellt werden?

Aufgabe 3:

Erstellen Sie eine CSS-Regel, sodass jeder nicht anders formatierte Paragraph in der Schriftart Arial und kursiv dargestellt wird.

Aufgabe 4:

Sie ändern die Farbe eines Textes. Was sollten Sie zusätzlich tun?

Aufgabe 5:

Nennen Sie eine Regel, die eine rote Linie unter einen Text mit der ID „unterstrichen“ zieht.

Aufgabe 6:

Schreiben Sie eine Regel, die dafür sorgt, dass der Text mit der ID „angepasst“ sowohl links als auch rechts bündig mit der Box ist.

5 Layout

Im Verlauf dieses Kapitels werden Ihnen die Grundlagen des Layout-Managments vorgestellt. Nach diesem Kapitel sollten Sie einen Überblick haben, wie man das Layout seiner Seite nach seinen Vorstellungen gestaltet.

5.1 Position

Mit Hilfe der **position**-Eigenschaft lassen sich Elemente beliebig auf dem Bildschirm anordnen. Dabei gibt es auch die Möglichkeit, die Elemente aus dem normalen Fluss der Seite herauszunehmen und beispielsweise am Browserfenster auszurichten. Nachdem man der Eigenschaft einen anderen Wert als den Standardwert zugeordnet hat, lassen sich die Elemente mit den Eigenschaften **left**, **right**, **top** und **bottom** ausrichten.

Achtung:

Alle durch position angeordneten Elemente laufen Gefahr, andere Elemente zu überdecken!

Der Standardwert für position ist **static**, nutzt man diesen Wert, wird das Element normal Im Fluss der Seite positioniert und die left, right, top und bottom Eigenschaften haben keine Auswirkung.

Anders sieht es schon aus, wenn man den Wert **relative** nutzt. Dieser Wert erlaubt es einem, das Element relativ zu der normalen Position zu platzieren. Mit left, right, top und bottom lässt sich nun der Abstand zu den entsprechenden Rändern/Elementen festlegen. Freigewordener Raum wird nicht von anderen Elementen genutzt! Im Beispiel rechts wurde der rechte grüne Block mit position:relative platziert.

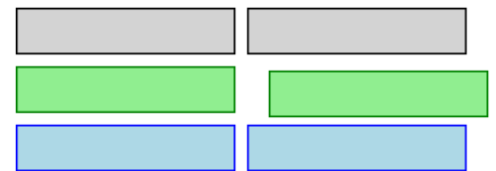


Abb. 5.1: position: relative; top: 2px; left: 10px;

Der Wert **fixed** erlaubt es, ein Element beliebig im Browserfenster zu platzieren. Die Positionsattribute beziehen sich ebenfalls auf die Position im Browserfenster. An der Stelle, an der es normalerweise stünde, entsteht keine Lücke, der Platz wird mit dem nächsten Element aufgefüllt. Auch beim Scrollen bleibt ein fixiertes Element an der gleichen Stelle stehen, dies erlaubt es beispielsweise Hinweise auf Cookies zu erstellen, die erst verschwinden, wenn sie weggeklickt werden.

Mit **absolute** lässt sich ein Element, ähnlich wie mit fixed, komplett aus dem Fluss herausnehmen, das heißt, die hinterlassene Lücke wird mit dem nächsten Element aufgefüllt. Die Positionsangaben beziehen sich hierbei jedoch auf das umschließende Element, das heißt, beim Scrollen kann das so positionierte Element verschwinden. Es ist jedoch Vorsicht geboten: Hat das zu positionierende Element kein umschließendes positioniertes Element, orientiert es sich am Browserfenster. Beim Scrollen wird es, im Gegensatz zu einem fixierten Element, jedoch mit.

Ein mit dem Wert **sticky** versehenes Element verhält sich wie folgt: Es bleibt an der Stelle, an die es gesetzt wurde, so lange, bis der Nutzer an dem Element „vorbei“ scrollt. Wenn dies geschieht, bleibt das Element an der festgelegten Position stehen. Die Positionsangaben beziehen sich hierbei

auf das Browserfenster. Zu beachten ist, dass das Element nicht stehen bleibt, wenn man keine Positionsangabe macht. Will man, dass das Element beim horizontalen Scrollen bestehen bleibt, muss man den Abstand zum linken Rand angeben. Will man, dass es beim vertikalen Scrollen bestehen bleibt, muss man den Abstand nach oben angeben.

5.1.1 Der z-Index

Wenn man Elemente so positioniert, dass sie andere Elemente überlappen, kann man mittels der Eigenschaft **z-index** festlegen, welches Element „näher am Betrachter“, also oben liegt. Dies ist in Abb. 5.2. zu sehen. Akzeptierte Werte sind alle ganzen Zahlen. Ein Element mit einem negativen z-index liegt stets hinter Elementen ohne z-index, ein Element mit positivem z-index steht davor. Der Standardwert ist also 0. Hinweis: Damit der z-index funktioniert, müssen Elemente, die diese Eigenschaft haben, einen anderen Wert als `static` bei `position` haben. Notfalls kann man `position: relative` setzen, dies an sich verändert den Fluss nicht aber erlaubt die Positionierung auf der z-Achse.

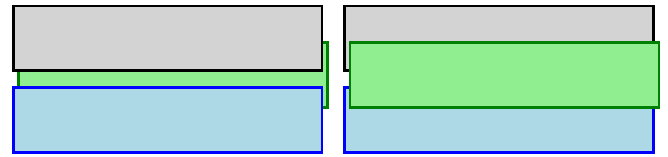


Abb. 5.2: links: grün hat z-index: -1, rechts: grün hat z-index: 1

5.2 Float und Clear

Mit dem **float** Attribut lassen sich Elemente in ihrem umgebenden Element platzieren. Dies erlaubt es beispielsweise, ein Bild ordentlich neben Text zu platzieren. Gültige Werte hierfür sind **left**, **right**, **none** und **inherit**. `left` bzw. `right` positionieren die Elemente auf der entsprechenden Seite. `none` ist der Standardwert, `inherit` bedeutet, dass das Element den Wert von seinem umgebenden Element erbt. Abb. 5.3 und 5.4 zeigen den Unterschied zwischen einem Block, der mit `float` platziert wurde, und einem, der nicht extra platziert wurde.

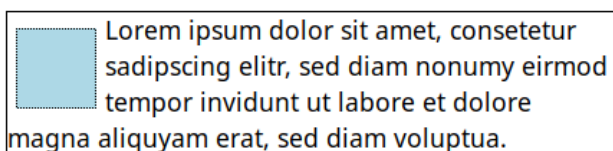


Abb. 5.3: Der blaue Block wurde mittels `float:left` platziert

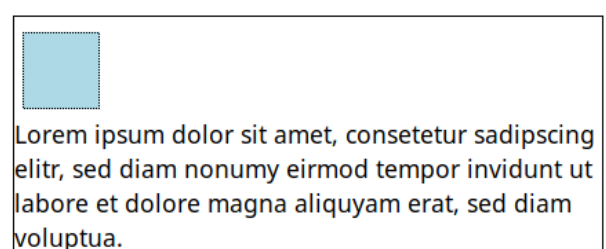


Abb. 5.4: Das gleiche Beispiel ohne `float`

Will man verhindern, dass Elemente neben einem Element schweben können, nutzt man das Attribut **clear**. `clear` akzeptiert die Werte **none**, **left**, **right**, **both** und **inherit**. `none` ist der Standardwert, `inherit` bedeutet, dass der Wert vom umschließenden Element übernommen wird. Setzt man `clear`:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor.

Abb. 5.5: Der blaue Block wurde mittels `float:right` platziert

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor.

Abb. 5.6: Der Textabschnitt wird mittels `clear:right` von schwebenden Elementen freigehalten

left; kann links neben dem Element kein Element schweben, das gilt in umgekehrter Version auch für clear: right;. clear:both verbietet das Schweben von Elementen auf beiden Seiten. In den Abbildungen 5.5 und 5.6 wurde der blaue Block mit float:right platziert, und in Abb. 5.6 wurde der Text durch clear:right von dem Block freigehalten.

5.2.1 Der Clearfix-Hack

Enthält ein Container ein schwebendes Element, das größer ist, als der Container selbst, kommt es vor, dass das Element aus dem Container herausragt. Dies passiert, da schwebende Elemente nicht zum Inhalt eines Containers zählen. Um dies zu verhindern, wird auf den Container ein sogenannter Clearfix angewendet. Dieser fügt hinter dem Container einen leeren Inhalt hinzu und nutzt clear:both, um zu verhindern, dass er „unter“ das bereits schwebende Element fällt.

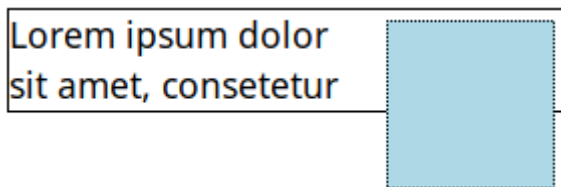


Abb. 5.7: Beispiel ohne Clearfix

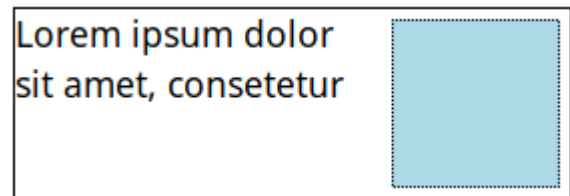


Abb. 5.8: Beispiel mit Clearfix

Für den Clearfix kann man entweder eine eigene Klasse erstellen und diese auf viele Elemente anwenden, oder man erweitert eine existierende Klasse/ID um die entsprechenden Deklarationen. Zunächst benötigt man **.containername::after**, um die Änderungen nach dem Container zu machen. In diese Regel benötigt man die Deklaration **content: „“**; um einen Block hinzuzufügen bei dem man mit **clear: both** verhindern kann, dass er unter schwebende Elemente enthält. Zusätzlich nutzt man **display: table**; um den Block so breit zu machen, wie der Container ist. Abb. 5.9 veranschaulicht die Selektoren, die man für einen Clearfix braucht, nochmal.

```
1  .fixable::after {  
2      content: "";  
3      clear: both;  
4      display: table;  
5  }
```

Abb. 5.9: Die Clearfix-Regel

Zusammenfassung

In diesem Kapitel wurde beschrieben, wie man Elemente auf seiner Website korrekt positioniert. Es wurde die position-Eigenschaft und ihre Werte erklärt, es wurde erklärt, was es mit dem z-Index auf sich hat und es wurde dargelegt, wie man mit float und clear umgeht. Außerdem wurde erklärt, wo Fehler auftreten können und wie man diese vermeiden kann.

Aufgaben zur Selbstprüfung

Aufgabe 1:

Wie erstellen Sie ein Element, welches immer die gleiche Position im Fenster hat?

Aufgabe 2:

Was kann passieren, wenn man beim positionieren von Elementen nicht aufpasst?

Aufgabe 3:

Wie können Sie ein Bild im Hintergrund eines Textes platzieren?

Aufgabe 4:

Wie kann man Bilder (oder andere Blocks) ordentlich neben einen Text (oder andere Elemente) platzieren?

Aufgabe 5:

Sie haben ein Bild in einen Rahmen mit Text platziert, aber das Bild ist zu groß für den Rahmen und schaut unten heraus. Wie beseitigen Sie diesen Fehler?

A Lösungen der Aufgaben zur Selbstprüfung

1.1

CSS, also Cascading Style Sheets, ist eine Designsprache, die es ermöglicht, Webseiten bis ins kleinste Detail abzustimmen.

1.2

Unter responsivem Webdesign versteht man die Erstellung von Websites, die auf allen Endgeräten, unabhängig von Größe oder Auflösung des Bildschirms und Eingabemöglichkeiten, gut zu lesen und zu nutzen sind.

1.3

Die Kaskadierung in CSS erlaubt es, ein generelles Design für Gruppen von Elementen zu entwerfen und trotzdem noch in der Lage zu sein, Untergruppen oder einzelne Elemente anders zu gestalten. Dies spart viel Arbeit.

1.4

Nutzt man den inline-style direkt im HTML-Dokument, geht die Übersichtlichkeit des Codes verloren. Außerdem verliert man die Vorteile der Kaskadierung und muss, wenn man den Style von verschiedenen Elementen ändern will, bei jedem Element den inline-style umschreiben anstatt es einmal zentral machen zu können.

1.5

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

2.1

Regel (Selektor { Deklaration(Eigenschaft: Wert;) })

2.2

```
#first { background: yellow; }
```

2.3

```
nav > a
```

2.4

Alle Zitate, deren Sprache mittels lang="de" auf deutsch gesetzt wurde.

2.5

```
*[class|="big"]
```

3.1

Eine Box besteht aus dem Außenabstand (margin), dem Rahmen (border), dem Innenabstand (padding) und dem Inhalt(content).

3.2

$10\text{px} + 2\text{px} + 5\text{px} + 50\text{px} + 5\text{px} + 2\text{px} + 10\text{px} = 84\text{px}$

3.3

Man nutzt entweder `overflow: hidden;` oder `overflow: scroll;`.

4.1

Es gibt Schriftarten mit Serifen und es gibt Schriftarten ohne Serifen.

4.2

Eine serifenlose Schriftart.

4.3

```
p { font-family: Arial; font-weight: bold; }
```

4.4

Man sollte zusätzlich noch die Hintergrundfarbe ändern.

4.5

```
#unterstrichen { text-decoration: underline red; }
```

4.6

```
#angepasst { text-align: justify; }
```

5.1

Man nutzt die Deklaration `position: fixed;` und legt mit `left`, `right`, `top` oder `bottom` die Position fest.

5.2

Es könnte passieren, dass ein Element ein anderes verdeckt.

5.3

Man nutzt `position: absolute;` und platziert das Bild mit `left`, `right`, `top`, oder `bottom` an die richtige Position. Zusätzlich nutzt man `z-index: -1;` um dafür zu sorgen, dass das Bild im Hintergrund ist.

5.4

Man nutzt `float: left` oder `float: right`, je nachdem, auf welcher Seite sich das Bild befinden soll.

5.5

Man wendet einen Clearfix an.

B Literaturverzeichnis

Andy Budd, Emil Björklund: CSS Mastery; Apress Verlag; 2016; DOI <https://doi.org/10.1007/978-1-4302-5864-3>

Norbert Hammer, Karen Bensmann: Webdesign für Studium und Beruf; Springer Verlag; 2011; ISBN 978-3-642-17069-0

Christoph Zillgens: Responsive Webdesign; Carl Hanser Verlag; 2012; ISBN 978-3-446-43015-0

C Abbildungsverzeichnis

Abb. 1.1: Responsives Webdesign.....	2
Abb. 1.2: Der weiter unten definierte Wert wird angenommen.....	2
Abb. 2.1: Eine simple CSS Regel.....	6
Abb. 2.2: HTML-Code für die folgenden Beispiele.....	7
Abb. 2.3: Der Nachfahren-Kombinator.....	7
Abb. 2.4: Der Kind-Kombinator.....	7
Abb. 2.5: Der Nachbar-Kombinator.....	7
Abb. 2.6: Der Geschwister-Kombinator.....	7
Abb. 3.1: Das Box-Modell.....	10
Abb. 3.2: Weitere Rahmenstile.....	11
Abb. 4.1: Veranschaulichung von Serifen.....	12
Abb. 4.2: text-align: justify;.....	14
Abb. 5.1: position: relative; top: 2px; left: 10px;.....	16
Abb. 5.2: links: grün hat z-index: -1, rechts: grün hat z-index: 1.....	17
Abb. 5.3: Der blaue Block wurde mittels float: left platziert.....	17
Abb. 5.4: Das gleiche Beispiel ohne float.....	17
Abb. 5.5: Der blaue Block wurde mittels float: right platziert.....	17
Abb. 5.6: Der Textabschnitt wird mittels clear: right von schwebenden Elementen freigehalten.....	17
Abb. 5.7: Beispiel ohne Clearfix.....	18
Abb. 5.8: Beispiel mit Clearfix.....	18
Abb. 5.9: Die Clearfix Regel.....	18

D Tabellenverzeichnis

Tab. 1.1: Vergleich verschiedener Farben und ihrer Schreibweisen.....	4
Tab. 1.2: Relative Längeneinheiten.....	4
Tab. 1.3: Absolute Längeneinheiten.....	5