Web-Grundlagen Teil 4:
Einführung in HTML und CSS

Weitere Selektoren

Selektoren dienen dazu, Elemente des HTML-Dokuments auszuwählen, für die bestimmte CSS-Eigenschaften gesetzt werden sollen. Zusätzlich dienen Selektoren in JavaScript dazu, Elemente auszuwählen, etwa um diese Elemente dynamisch anzupassen oder Event-Handler für an diesen Elementen auftretende Ereignisse zu registrieren. Es ist also ratsam, sich etwas genauer mit den Selektoren zu beschäftigen.

Im Teil 3 haben wir bereits ein paar Selektoren gesehen:

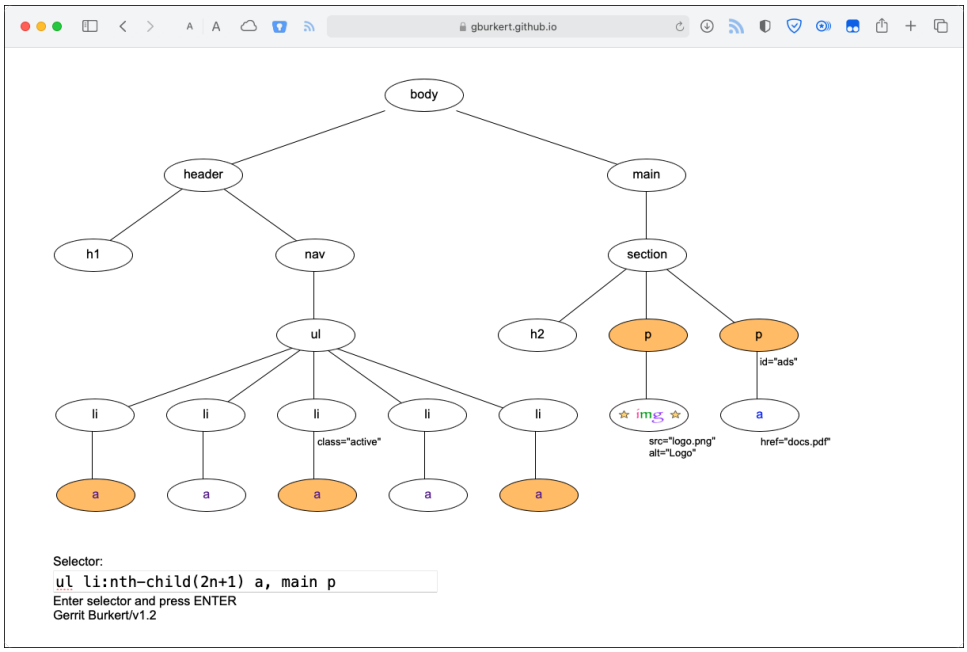
Selektor	**Beispiel**	**Bedeutung**
Typselektor	`h1`	alle h1-Elemente
Klassenselektor	`.neu`	alle Elemente mit der Klasse "neu"
ID-Selektor	`#menu`	das Element mit der id="menu"

Sobald nicht alle Elemente eines bestimmten Typs (zum Beispiel alle h1) ausgewählt werden sollen, ist man mit diesen Selektoren darauf angewiesen, den HTML-Code um Klassen oder IDs zu ergänzen. Das ist nicht immer erwünscht und deshalb gibt es zahlreiche weitere Selektoren, welche es erlauben, bestimmte Elemente oder Gruppen von Elementen aus einem HTML-Dokument auszuwählen. Die wichtigsten werden wir uns nun ansehen.

Selektor	**Beispiel**	**Bedeutung**
Kontext	`p strong`	alle strong-Elemente unter p
Kind	`.intro > h2`	alle h2 unmittelbar unter einem Element mit der Klasse intro
nächstes Geschwister	`h2 + p`	alle p, die auf gleicher Ebene unmittelbar auf ein h2 folgen
Geschwister	`h2 ~ p`	alle p auf gleicher Ebene nach einem h2
Alternativen	`h1, h2`	alle h1 oder h2
Pseudoklassen	`:first-child`	alle Elemente, die erstes Kindelement eines Elements sind
Pseudoelemente	`::first-line`	die erste Textzeile
Attribut	`[src]`	Elemente mit einem src-Attribut

Das ist nur ein erster grober Überblick. Tatsächlich gibt es zahlreiche Variationen und Kombinationsmöglichkeiten dieser Selektoren. Wir beschränken uns auf eine Auswahl davon.

Mit einer kleinen Demo-Anwendung können Sie die Selektoren ausprobieren und üben. Sie finden diese unter: https://gburkert.github.io/selectors/.



Aufgaben:

- Öffnen Sie die Selektor-Demo und probieren Sie, anhand der obigen Angaben die folgenden Elemente der HTML-Struktur auszuwählen:
  - Alle a-Elemente
  - Das Element mit der Klasse "active"
  - Das Element mit der ID "ads"
  - Alle a-Elemente unterhalb von main
  - Alle a-Elemente unmittelbar unterhalb einem Element mit der Klasse "active"
  - Das p-Element unmittelbar nach dem h2
  - Alle p-Elemente unterhalb von main, die nach einem h2 vorkommen
  - Alle li- und alle p-Elemente
  - Das erste li-Element in der Navigation
  - Das zweite li-Element in der Navigation
  - Alle li-Elemente ausser dem ersten
  - Alle Elemente mit einem alt-Attribut
- Überlegen Sie, welche Elemente die folgenden Selektoren auswählen. Probieren Sie es aus.
  - `:first-child`
  - `a:first-child`
  - `main a, li:first-child + li a`

- Der Universalselektor * wählt alle Elemente aus. Probieren Sie die folgenden Kombinationen:
  - `main *`
  - `body * header`
  - `body * ul`
  - `nav > * > a, section > * > a`
- Pseudoklassen beziehen sich auf den Zustand oder die Position eines Elements. Probieren Sie den folgenden Selektor und fahren Sie mit dem Mauszeiger über verschiedene Elemente:
  - `nav a:hover`
- In neueren CSS-Versionen wurden verschiedene Pseudoklassen hinzugefügt. Eine Auswahl davon zum Testen in der Selektor-Demo:
  - `li:nth-child(4)`
  - `li:nth-child(2n+1)`
  - `li:nth-child(even)`
  - `p:nth-child(2)`
  - `p:nth-of-type(2)`
- Und noch ein paar Möglichkeiten des Attributselektors:
  - `[href="docs.pdf"]`
  - `img[src$=".png"]`
  - `p a[href^="doc"]`

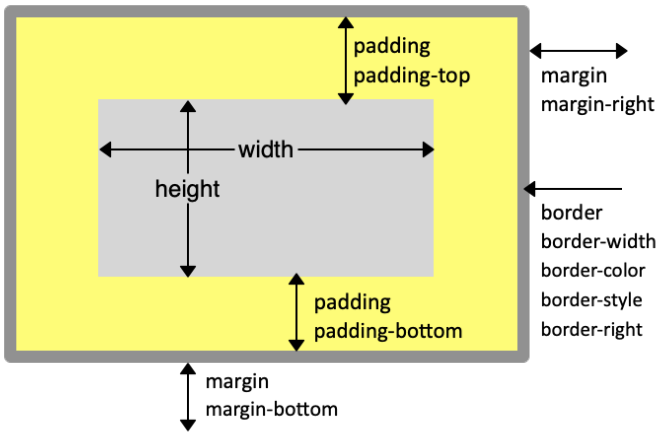
Auch wenn es noch viele weitere Pseudoklassen und Pseudoelemente gibt, sollen die erwähnten Selektoren erst einmal genügen. Bei Bedarf kann man ja jeweils eine der Referenzen konsultieren:

- https://www.w3schools.com/cssref/css_selectors.asp
  Überblick über die Selektoren
- https://www.w3.org/TR/selectors-3/
  CSS-Standard «Selectors Level 3» (Level 4 hat erst den Status eines «Working Draft»)

Box-Modell

Sobald man etwas mehr machen möchte als Text schön zu gestalten, muss man sich mit der Darstellung und Positionierung von Seitenbereichen beschäftigen. Ein erster Schritt ist das Verstehen des CSS-Box-Modells und der zugehörigen Eigenschaften. Die folgenden Ausführungen beziehen sich auch Block-Elemente wie *div*, *p* oder *h1*.

Zunächst hat eine Box eine bestimmte Breite und eine bestimmte Höhe (Eigenschaften *width* und *height*). Weiter kann eine Box von einem Rahmen umgeben sein (*border*). Schliesslich gibt es einen Abstand zwischen Rahmen und Inhalt (Innenabstand, *padding*), sowie ausserhalb des Rahmens zu anderen Elementen (Aussenabstand, *margin*). Das Bild zeigt diese Eigenschaften im Überblick:

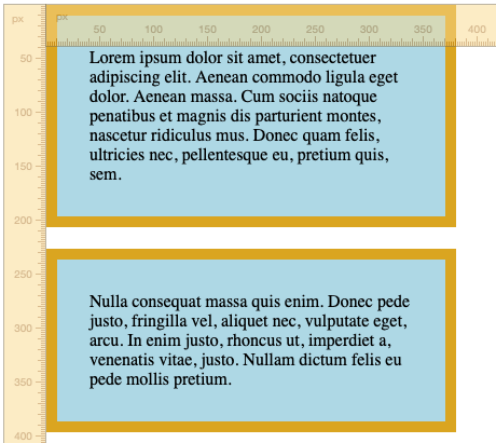


Ein paar Versuche sollen die Zusammenhänge verdeutlichen.

Aufgaben:

- Zum Start verwenden wir wieder das Beispiel mit den drei Absätzen aus Teil 3 dieser Einführung und starten mit leerem CSS. Öffnen Sie die Seite im Code-Editor und am besten auch gleich eine Vorschau im Browser.
- Ergänzen Sie eine CSS-Regel, welche die den benötigten Platz visualisiert:
  `p { background-color: lightblue; }`
- Ergänzen Sie die Regel um eine Breitenangabe:
  `width: 25vw;`
  Es zeigt sich, dass die Absätze höher werden, wenn die Breite reduziert ist.
- Setzen Sie zusätzlich die Höhe:
  `height: 12rem;`
- Überschreiben Sie die Breite des letzten Absatzes mit einem zu kleinen Wert:
  `p:nth-of-type(3) { width: 7em; }`
  Der Text ragt nun unten aus der Box heraus. Das Verhalten in diesem Fall kann mit der overflow-Eigenschaft gesteuert werden. Probieren Sie:
  `overflow: hidden;`
  `overflow: scroll;`
- Wenn möglich lassen wir die Boxen ihre Höhe anhand des Inhalts bestimmen. Löschen Sie also alle Höhen- und Overflow-Angabe wieder.
- Um die Grössen besser vergleichen zu können, sollen nun alle Angaben in *px* erfolgen. Setzen Sie ausserdem für alle Absätze einen Innen- und Aussenabstand sowie einen Rahmen:
  ```
  width: 300px;
  padding: 30px;
  margin: 20px;
  border: 10px solid goldenrod;
  ```

Das Bild zeigt den aktuellen Stand, ergänzt um ein Lineal zum Messen der Grössen:



Einige Hinweise zum Zwischenergebnis:

- Die Breite der Box inklusive Innenabstand und Rahmen ist nun 380px[1], da sich *width* (300px) nur auf die Breite des Inhalts bezieht. Häufig möchte man mit *width* aber die gesamte Breite der Box angeben, inklusive Innenabstand und Rahmen. Das ist durch diese Angabe möglich: `box-sizing: border-box;`
- Der vertikale Abstand zwischen den beiden Boxen ist 20px, obwohl jede einen Aussenabstand von 20px hat. Die Abstände addieren sich in diesem Fall nicht, es wird der grössere der beiden genommen (collapsing margins). Da beide gleich gross sind also 20px.
- Die Abstände und Rahmenangaben können für alle Richtungen einzeln angegeben werden, zum Beispiel: *margin-top, padding-left, border-right, border-bottom-style, border-right-color, …*
- Nach *margin, padding, border-width* können eine bis vier Zahlen stehen. Bei vier Zahlen bezieht sich die erste auf *top*, die anderen auf die weiteren Seiten im Uhrzeigersinn.
- Es gibt auch *max-width* und *min-width* für maximale und minimale Breite. Entsprechende Angaben sind für die Höhe von Boxen möglich.

Aufgaben:

- Ersetzen Sie die Breitenangaben für alle Boxen durch maximale und minimale Grössen:
  ```
  max-width: 50rem;
  min-width: 25rem;
  ```
- Testen Sie, wie sich die Grösse mit verschiedenen Grössen des Browserfensters verändert. So kann man erreichen, dass Texte weder zu breit noch zu schmal für angenehmes Lesen werden.
- Ecken können auch abgerundet werden. Probieren Sie diese Varianten:
  ```
  border-radius: 2rem;
  border-radius: 50%;
  ```

[1] width + 2 * padding + 2 * border-width

Wir haben nun einige Möglichkeiten zur Verfügung, Boxen zu gestalten. Es gibt natürlich noch zahlreiche weitere, etwa Hintergrundfarben, Farbverläufe, oder Hintergrundbilder. Als nächstes konzentrieren wir uns aber darauf, wie die Boxen auf der Seite platziert werden.

Positionierung

Was noch offen ist: wie können die Boxen auf einer Seite positioniert werden? Wenn wir nichts weiter unternehmen, werden die Block-Elemente einfach untereinander auf der Seite platziert. Jede Box nimmt die verfügbare Breite in Anspruch, ausser wir geben eine andere Breite an. Diese Positionierungsart nennt man *statische Positionierung*. Das ist die Default-Einstellung. In den folgenden Aufgaben werden die verschiedenen Alternativen ausprobiert.

Aufgaben:

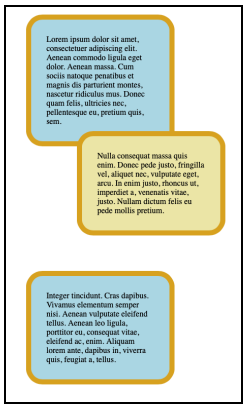
- Damit wir eine einheitliche Ausgangsposition haben, setzen Sie das Beispieldokument auf nur eine CSS-Regel zurück:

```css
p {
  background-color: lightblue;
  box-sizing: border-box;
  width: 25vw;
  padding: 30px;
  margin: 20px;
  border: 10px solid goldenrod;
  border-radius: 2rem;
}
```

- Nun probieren wir anhand der mittleren Box verschiedene Positionierungsvarianten aus:

```css
h1:first-child + p + p {
  position: static;
  background-color: palegoldenrod;
}
```

- Da die *statische Positionierung* die Voreinstellung ist, ändert sich an der Position der Box erst einmal nichts. Dies ist auch der Fall, wenn wir die Box relativ positionieren:
  `position: relative;`
- Nun haben wir aber die Möglichkeit, die Position zu verschieben:
  `top: -50px;`
  `left: 100px;`
- Beachten Sie, dass die dritte Box ihre Position nicht verändert (Bild), die mittlere Box reserviert immer noch den ursprünglichen Platz, ist aber horizontal und vertikal versetzt. Neben *top* und *left* können auch *bottom* und *right* verwendet werden.



- Ändern Sie die Position nun auf absolut:
  `position: absolute;`
  `top: 0; left: 0;`
- Nun verschwindet der Zwischenraum zwischen der ersten und dritten Box. Die mittlere Box wird wie aus dem Fluss der Elemente herausgehoben. *Ihre Position richtet sich nun nach dem nächst höheren Element in der HTML-Hierarchie, welche nicht statisch positioniert ist.* Alles klar? Eventuell müssen Sie den Satz nochmal lesen. Wir haben oberhalb des zweiten *p* aber kein nichtstatisches Element im Dokument. Daher richtet sich die Position nach den Grenzen des Dokuments. Es ist also in der linken oberen Ecke. Fast. Denn die Box hat noch einen Aussenabstand.
- Probieren Sie verschiedene Positionen der Box mit absoluter Positionierung. Verwenden Sie die Eigenschaften *top, bottom, left, right*.
- Ändern Sie die Positionierung wieder auf *static*. Die Eigenschaften *top, bottom, left, right* haben dann keine Wirkung. Setzen Sie den Aussenabstand nun folgendermassen:
  `margin: 20px auto;`
  Das bedeutet: oben und unten 20px, links und rechts automatisch.[2] So wird die Box horizontal zentriert. Testen Sie es, indem Sie die Grösse des Browserfensters variieren.
- Bringen Sie die mittlere Box mit dieser Ergänzung ganz zum Verschwinden:
  `display: none;`

Mit der *display*-Eigenschaft kann man ein Element also komplett aus der Anzeige entfernen. Es nimmt dann auch keinen Platz mehr ein. Im Unterschied dazu macht *visibility:hidden* ein Element unsichtbar, sein Platz bleibt aber reserviert. Mit Hilde der *display*-Eigenschaft kann man übrigens ein Block-Element zu einem Inline-Element machen (*display:inline*) oder umgekehrt (*display:block*).

Ein weiterer *position*-Wert ist *fixed*. Die Position richtet sich in diesem Fall an den Grenzen des Viewports aus, das Element bleibt beim Scrollen stehen. Damit kann man also zum Beispiel eine Fusszeile unten im Browser-Fenster fixieren.

Noch ein Hinweis für Interessierte: eine gute Erklärung zur *position*-Eigenschaft mit Beispielen liefert der schon etwas ältere Artikel «CSS Positioning 101» von Noah Stokes (A List Apart):
https://alistapart.com/article/css-positioning-101/

Fliessende Boxen

Eine Sache fehlt noch, um einfache Layouts erstellen zu können: mit *float* und *clear* ist es möglich, Boxen durch andere Inhalte umfliessen zu lassen. Damit lassen sich einfache mehrspaltige Layouts umsetzen, wie sie für unsere Zwecke vorerst genügen sollen.

Mittlerweile gibt es mächtigere CSS-Ansätze für Layouts, zum Beispiel CSS Flexbox, CSS Grid, oder CSS Multicolumn Layout. Alle drei sind in eigenen CSS-Standards beschrieben und können im

[2] Entspricht der ausführlichen Schreibweise `margin: 20px auto 20px auto;`

Rahmen dieser Einführung nicht behandelt werden. Auf Flexbox und Grid wird aber im Wahlfach Mobile Applications (MOBA) eingegangen.

Die *float*-Eigenschaft ist eigentlich dazu gedacht, Bilder von Text umfliessen zu lassen. Mit der clear-Eigenschaft kann das Umfliessen beendet werden.

Aufgaben:

- Öffnen Sie das folgende Beispiel auf der Website w3schools.com: https://www.w3schools.com/css/tryit.asp?filename=trycss_layout_float
- Ersetzen Sie das *float:right* aus der CSS-Regel durch *float:none* (das ist die Standardeinstellung) und klicken Sie auf «Run». Das Bild befindet sich im HTML-Code am Anfang des zweiten Absatzes. Da *img* ein Inline-Element ist wird der Text rechts vom Bild fortgesetzt.
- Teilen Sie den zweiten Absatz in der Mitte durch Einfügen von `</p><p>` und setzen Sie in der CSS-Regel wieder *float:right*. Sie sehen, dass beide Absätze um das Bild fliessen.
- Ersetzen Sie *float:right* durch *float:left*.
- Setzen Sie den Stil des letzten Absatzes auf *clear:both*. Statt *both* wäre hier auch *right* möglich. Der Einfachheit halber können Sie den Stil direkt im Element angeben:
  `<p style="clear:both">`
- Mit *clear* kann das Umfliessen also beendet werden.

Damit haben wir die Möglichkeiten von *float* und *clear* im Wesentlichen ausprobiert. Mit diesen Eigenschaften sowie den Möglichkeiten des Box-Modells und der Positionierung lassen sich nun einfache Layouts aufbauen. Ein weiterer Versuch soll die Einführung von *float* und *clear* abschliessen.

Aufgaben:

- Nehmen Sie sich noch einmal das Beispieldokument aus den letzten Übungen vor mit der bereits verwendeten CSS-Regel zum Start:

```css
p {
  background-color: lightblue;
  box-sizing: border-box;
  width: 25vw;
  padding: 30px;
  margin: 20px;
  border: 10px solid goldenrod;
  border-radius: 2rem;
}
```

- Ergänzen Sie die Regel um *float:left*. Die drei Boxen werden nun nebeneinander dargestellt. Da die Breite der Boxen abhängig von der Viewport-Breite angegeben wurde, passen die drei Boxen immer nebeneinander. Ändern Sie die Breite des Browserfensters, um das zu testen.
- Ändern Sie die Breite der Boxen auf einen festen Wert, z.B. 350px. Ändern Sie wieder die Breite des Browserfensters. Jetzt rutschen die Boxen nach unten, wenn nicht genug Platz ist.

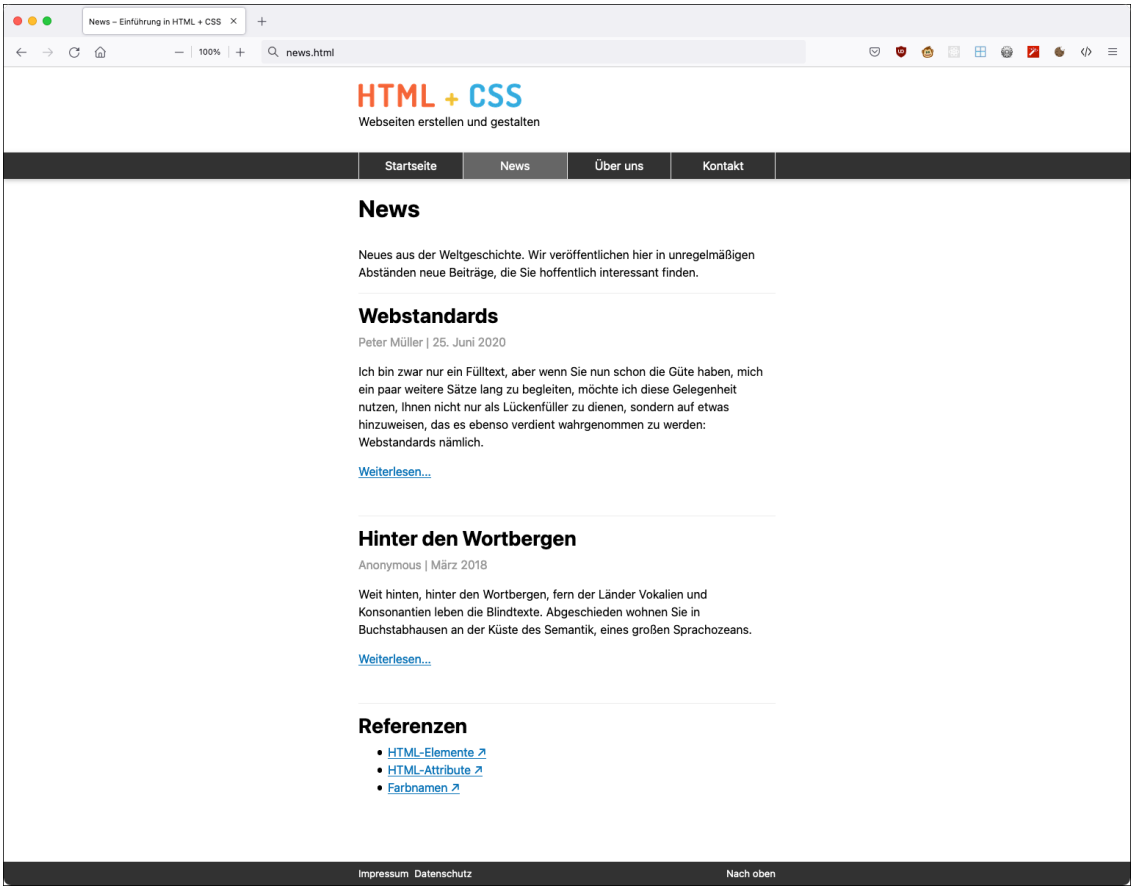
- Die Technik, Elemente mit *float:left* nebeneinander zu stellen, wird häufig eingesetzt um zum Beispiel horizontale Navigationen anzulegen. Da *li*-Elemente Block-Elemente sind, werden Sie normalerweise untereinander dargestellt. Die Bullet-Points bekommt man weg, indem man am *ul*-Element *list-style-type:none* setzt.

Seite mit HTML und CSS umsetzen

Die wichtigsten Möglichkeiten zum Bau von Webseiten mit HTML und CSS stehen Ihnen nun zur Verfügung. Zeit, diese für eine komplette Seite einzusetzen.

Aufgabe (empfohlen):

- Am Ende von Teil 2 dieser Grundlagen-Serie gab es eine Aufgabe, den HTML-Code für eine vorgegebene Seite zu erstellen. Nun ist es an der Zeit, zu versuchen, der Seite mit Hilfe von CSS das gewünschte Aussehen zu verpassen. Als Ausgangspunkt können Sie Ihr Ergebnis aus Teil 2 oder den HTML-Code aus der Musterlösung verwenden. Hier noch einmal die Seite:[3]



[3] Das Beispiel stammt aus dem Buch «HTML + CSS – Webseiten erstellen und gestalten» von Peter Müller (Rheinwerk-Verlag, 2020): https://html-und-css.de/buch/