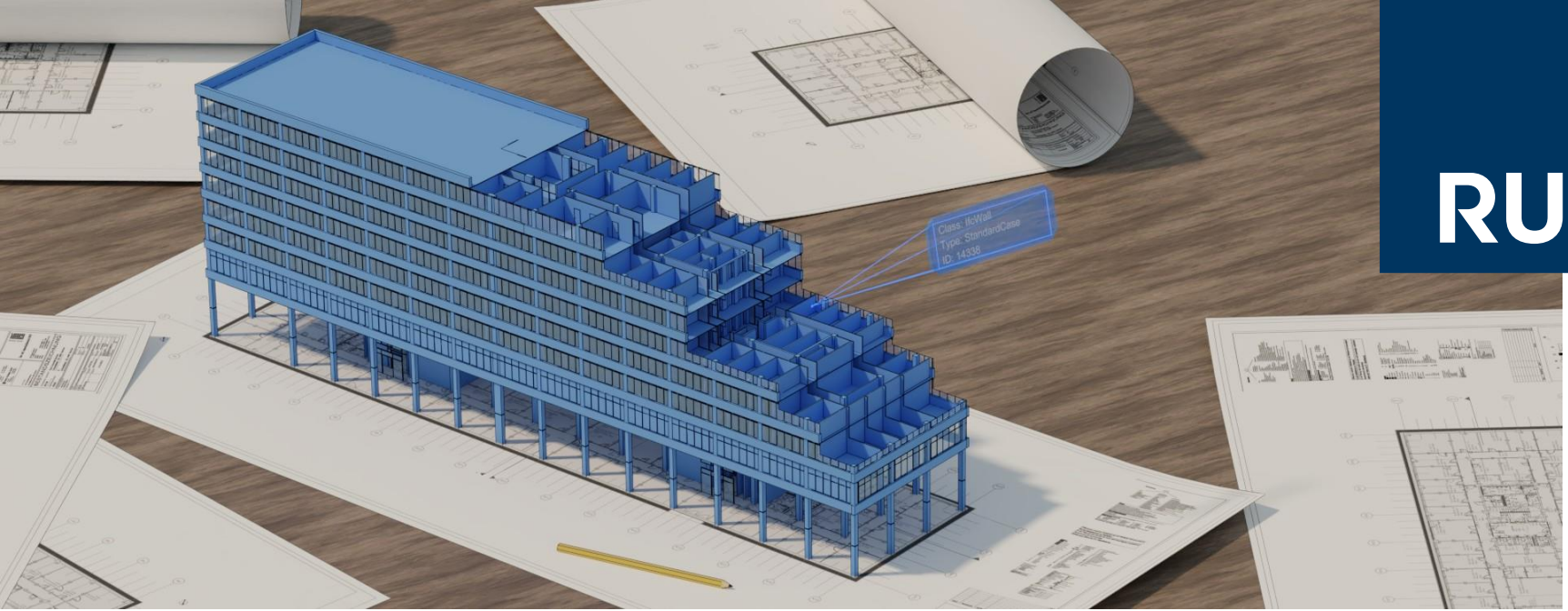


RUB

RUHR-UNIVERSITÄT BOCHUM

WEB-ENGINEERING

Informatik
im Bauwesen

Sommersemester 2023

Web-Engineering
CSS Einführung

Einführung

Was ist CSS

- CSS steht für Cascading Style Sheets
- CSS beschreibt wie ein Webseiten-Element auf dem Bildschirm, auf gedrucktem Papier oder in einem anderen Medium dargestellt wird
- Die Beschreibung von Struktur und Erscheinungsbild sind bei der Gestaltung von Webseiten getrennt zu betrachten

CSS Einbinden

CSS Code kann auf verschiedene Arten verwendet werden

- Inline CSS:
  - Kann benutzt werden, um ein einzelnes HTML-Element zu gestalten

```
<h1 style="color:blue;"> Das ist eine blaue Überschrift </h1>
```

- Internal CSS:
  - Kann benutzt werden, um eine einzelne HTML-Seite zu gestalten

```
<head>
    <style>
        body {background-color: blue;}
    </style>
</head>
```

CSS Einbinden

CSS Code kann auf verschiedene Arten verwendet werden

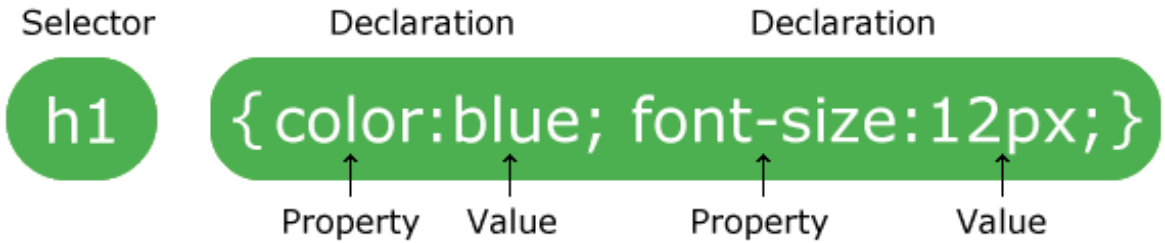
- External CSS:
  - Wird üblicherweise für die Gestaltung von Webseiten verwendet

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

Web-Engineering
CSS Syntax

Syntax

Ein CSS rule-set besteht aus mindestens einem Selektor und mindestens einer Deklaration.



Quelle : https://www.w3schools.com/css/css_syntax.asp

```
p {
    color: red;
    text-align: center;
}
```

Das Beispiel-Element `<p>` bekommt durch dieses Styling eine rote Textfarbe und wird zentriert dargestellt.

Selektoren

Einfache Selektoren

- Der `element` -Selektor selektiert alle Elemente mit diesem Namen
- Der `class` -Selektor selektiert alle Elemente mit dieser Klasse
- Der `id` -Selektor selektiert ein spezifisches Element mit dieser ID
- Der `*` -Selektor ist ein Universalselektor
- Mit `,` können Selektoren gruppiert werden

```
<p class="text1" id="para1">
```

Element-Selektor :

```
p {
  color: red;
}
```

Klassen-Selektor :

```
.text1 {
  color: red;
}
```

Id-Selektor :

```
#para1  {
  color: red;
}
```

Selektoren

Nachfolger- und Kindselektoren

- Selektoren können miteinander kombiniert werden
- Der Nachfolgerselektor (`Leerzeichen`) bezieht sich auf alle entsprechenden Elemente, die dem ersten Element untergestellt sind
- Der Kindselektor (>) bezieht sich auf alle entsprechenden Elemente, die dem ersten Element direkt unterstellt sind

Geschwisterselektor

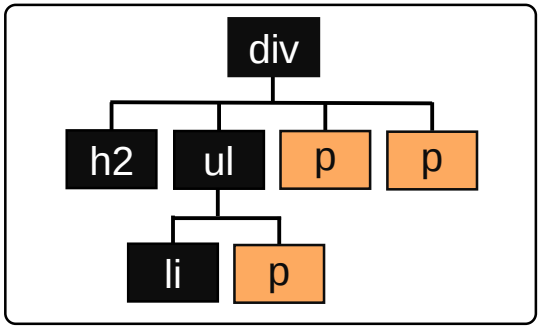
- Der Geschwisterselektor (~) bezieht sich auf das Element, welches dem ersten Folgt und die gleichen Eltern besitzt
- Der angrenzende (+) Geschwisterselektor bezieht sich auf das Element, welches dem ersten direkt Folgt und die gleichen Eltern besitzt

Selektoren

Nachfolger- und Kindselektoren

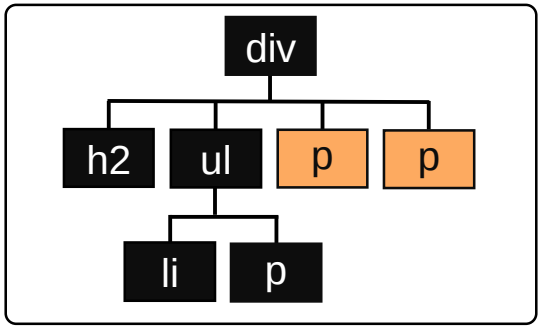
Der Nachfolgerselektor

```
div p
{
  background-color: orange;
}
```



Der Kind-Selektor

```
div > p
{
  background-color: orange;
}
```

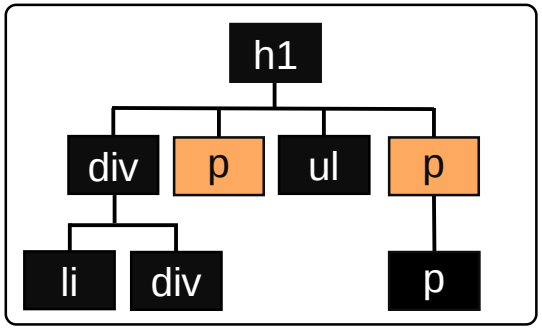


Selektoren

Geschwisterselektor

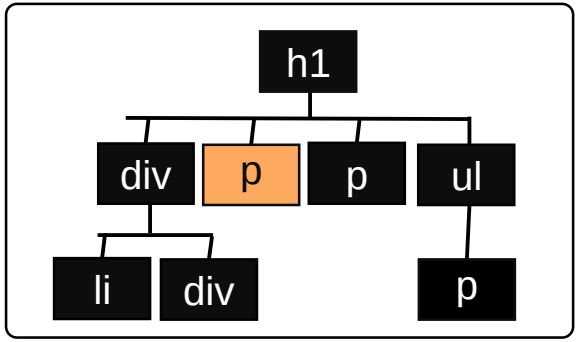
Der Geschwisterselektor

```
div ~ p
{
  background-color: orange;
}
```



Der angrenzende Geschwisterselektor

```
div + p
{
  background-color: orange;
}
```



Selektoren

Pseudoselektoren

- Der Pseudo-Element Selektor (`:`) kann Elemente basierend auf ihrem Status auswählen. Beispielsweise das fünfte Child-Element
  - Ein besuchter Link kann mit `a:visited` angewählt werden
  - Ein Link über dem die Maus schwebt mit `a:hover`
- Der Pseudo-Klassen Selektor (`::`) kann spezielle Entitäten auswählen
  - Die erste Zeile eines Absatzes kann mit `p::first-line` angewählt werden

Der Pseudo-Element Selektor

```
a:hover {
  color: yellow;
}
```

Der Pseudo-Klassen Selektor

```
p::first-line {
  color: yellow;
}
```

Listen für mögliche Selektoren finden sich online z.B. unter *https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes*

Selektoren

spezifische Selektion eines Kindes mittels `nth-child`-Anweisung

- In diesem Beispiel wird die Klasse `elem` referenziert
- Die `nth-child`-Anweisung wird auf die `paragraph`-Elemente im `div`-Container angewendet und bekommt ein Index als Argument übergeben (im Beispiel die `4`)

```
<body>
    <div class="elem">
      <p>Element_1</p>
      <p>Element_2</p>
      <p>Element_3</p>
      <p>Element_4</p>
      <p>Element_5</p>
    </div>
</body>

.elem p:nth-child(4) {
    color: red;
}
```

Element_1

Element_2

Element_3

Element_4

Element_5

Selektoren

Spezifische Selektion eines Kindes mittels `nth-child`-Anweisung

- Es gibt die Möglichkeit der `nth-child`-Anweisung eine Formel als Argument zu übergeben
- So können über große Strukturen hinweg Selektionen getroffen warden
- Formel: `nth-child(An+B)`
  A = integer Schrittweite,
  n = alle positive integer beginnend bei 0,
  B = Startwert

Jedes Element mit ungeraden Index wird ausgewählt:

```
.elem p:nth-child(2n + 1) {
    color: red;}
```

Element_1

Element_2

Element_3

Element_4

Element_5

Selektoren

Spezifische Selektion eines Kindes mittels `nth-child`-Anweisung

- Es können für bestimme Fälle auch vordefinierte Schlüsselwörter genutzt werden
  - odd → wählt alle ungerade Elemente aus 1, 3, 5, 7…
  - even → wählt alle geraden Elemente aus 2, 4, 6, 8 …

Jedes Element mit ungeraden Index wird ausgewählt:

```
.elem p:nth-child(2n + 1) {
    color: red;}
```

Element_1

Element_2

Element_3

Element_4

Element_5

Web-Engineering
Webseitengestaltung

Colors

In CSS können Farben über die HEX/RGB/HSL-Werte definiert werden.

- Farben können für alle visuell dargestellten HTML-Elemente definiert werden.
- Optional können auch vordefinierte Farben verwendet werden. *https://www.w3schools.com/cssref/css_colors.asp*

Beispiele:

- RGB
  - `<h1 style="background-color:rgb(255, 99, 71);">...</h1>`
- HEX
  - `<h1 style="background-color:#ff6347;">...</h1>`
- HSL
  - `<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>`
- VORDEFINIERT
  - `<h1 style="background-color:Crimson;">...</h1>`

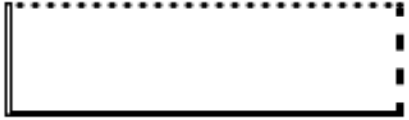
Colors

Mit den CSS `border` Eigenschaften können Rahmen gestaltet werden

- Die `border`-Eigenschaft ist eine Kurzschreibweise für die Definition aller Rahmeneigenschaften
- Rahmeneigenschaften können über die Eigenschaften `border-width`, `border-style` und `border-color` angepasst werden
  - Nur `border-style` MUSS angegeben werden

```
.borderedA {
    border: 4px solid green;
}
```

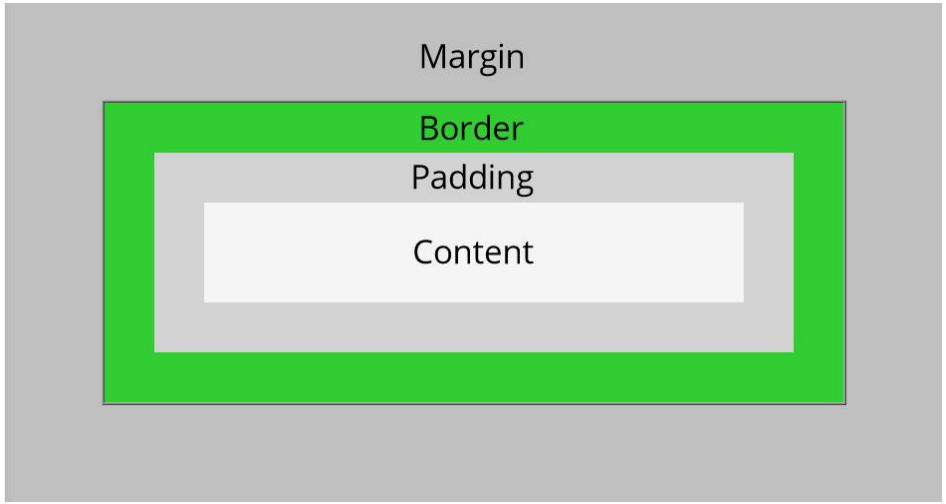
```
.borderedB {
    borderstyle:
      dotted dashed solid double;
}
```



Box Model

Das Box-Modell ist im wesentlichen eine Box, die jedes HTML-Element umschließt

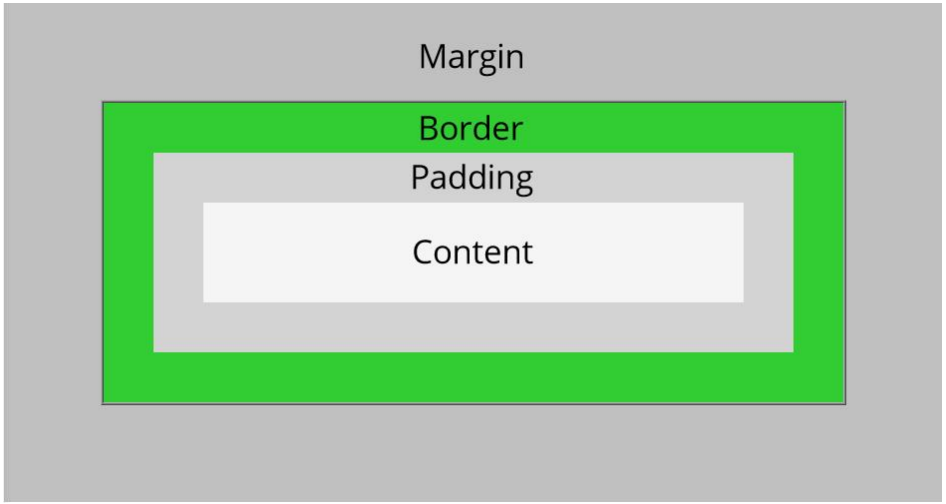
- Jede Box enthält `margins`, `borders`, `paddings` und den aktuellen `Inhalt`



Box Model

Erläuterung der Komponenten

- `Content` → ist eine Box, in welcher der Inhalt steht (z. B. Text und Bilder)
- `Padding` → ist ein Innenabstand des Inhalts zum Rahmen
- `Border` → ist ein Rahmen um den Inhalt inkl. padding
- `Margin` → ist ein Außenabstand vom Rahmen zu den umliegenden Elementen



Innen- & Außenabstand

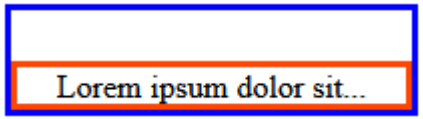
margin → Außenabstand

- Die `margin`-Anweisung setzt einen Außenabstand um den Rahmen eines Elements
- Der Außenabstand des HTML-Elements wird hier mit der Klasse `applyMargin` angepasst

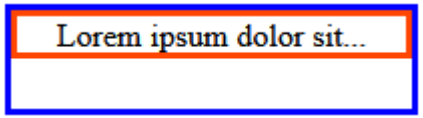
```
<div id="outborder" style="border:solid blue; width: 200px;">
  <div class="applyMargin" style="border: solid orangered;">
    <p>Lorem ipsum dolor sit...</p>
  </div>
</div>
```

Innen- & Außenabstand

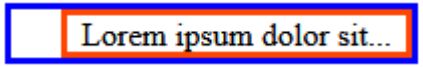
```
.applyMargin{
        margin-top: 25px;
}
```



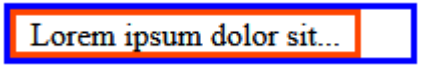

```
.applyMargin{
        margin-bottom: 25px;
}
```




```
.applyMargin{
        margin-left: 25px;
}
```




```
.applyMargin{
        margin-right: 25px;
}
```



Innen- & Außenabstand

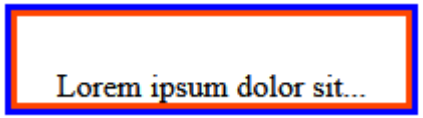
padding → Innenabstand

- Die `padding`-Anweisung setzt einen Innenabstand innerhalb des Rahmens eines Elements
- Der Außenabstand des HTML-Elements wird hier mit der Klasse `applyPadding` angepasst

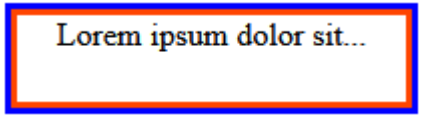
```
<div id="outborder" style="border:solid blue; width: 200px;">
  <div class="applyPadding" style="border: solid orangered;">
    <p>Lorem ipsum dolor sit...</p>
  </div>
</div>
```

Innen- & Außenabstand

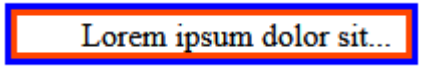
```
.applyPadding{
      padding-top: 25px;
}
```



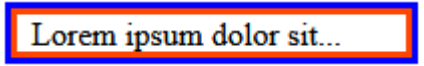

```
.applyPadding{
      padding-bottom: 25px;
}
```




```
.applyPadding{
      padding-left: 25px;
}
```




```
.applyPadding{
      padding-right: 25px;
}
```



Höhe und Breite

Die Eigenschaften `width` und `height` werden zur Definition der Größe einzelner Elemente verwendet.

- Angaben zur Größendefinition eines Elements können in CSS absolut mit beispielsweise `cm` (Centimeter), `mm` (Milimeter) oder `px` (Pixel) angegeben werden

```
#ueberschrift {
    height: 35px;
}
```

Höhe und Breite

Die Eigenschaften `width` und `height` werden zur Definition der Größe einzelner Elemente verwendet.

- Die Definition der Größe kann auch Relative angegeben werden, mit `vw` (relative zur Breite des Bildes), `vh` (relative zur Höhe des Bildes), % (relative zum Eltern-Element)
  - `vw` und `vh` passen auch Schriftgrößen an den Viewport des Geräts oder die Größe des Browserfensters an
  - 1vw = 1% der Breite des Viewports, 1vh = 1% der Höhe des Viewports

```
.maincontent {
    width: 25%;
}
```

```
.maincontent {
    width: 40vw;
    height: 25vh;
}
```

Höhe und Breite

Die Tatsächliche Höhe und Breite eines Elements berechnet sich wie folgt:

- Real Height = Height + Padding + Border
- Real Width = Width + Padding + Border

Jedes HTML-Element kann separat in seiner Größe angepasst werden.

Diese Art der Größenberechnung eines Elements kann mit folgender Anweisung angepasst werden:

```
{box-sizing: border-box;}
```

- Diese Eigenschaft sorgt dafür, das padding und border in die Berechnung von Height und Width einbezogen werden

Fonts

Font Benutzung

- Die Font eines Textes wird mit dem `Font-Family`-Attribut festgelegt
- Über den `family-name` werden verschiedene Fonts der Priorität nach angegeben, wenn eine Font nicht verfügbar ist, wird die nächste in der Liste gewählt
- Als letztes sollte die `generic-family` angegeben werden, diese kommt zum Einsatz, wenn keine der Fonts verfügbar ist – zum Beispiel `serif`, `cursive`, `monospace` oder weitere, wobei dann eine Font dieser Familie gewählt wird
- Fonts lassen sich lokal oder aus dem Web einbinden, dazu wird `@font-face` verwendet
- Die Textgröße wird über die `font-size` Eigenschaft gesetzt
- Es gibt verschiedene Font Formate mit unterschiedlicher Kompatibilität

Fonts

Font Beispiele

```
@font-face {
  font-family: 'Roboto';
  src: url(https://fonts.gstatic.com/s/roboto/v20/
  KFOlCnqEu92Fr1MmSU5fChc4EsA.woff2) format('woff2');
}
.p1 {
  font-family: Times New Roman, Lucida Console, Times, serif;
}
.p2 {
  font-family: Gnurps, Lucida Console, Times, serif;
  font-size: 20px;
}
.p3 {
  font-family: "Roboto", "Courier New", monospace;
  font-size: 25px;
}
```

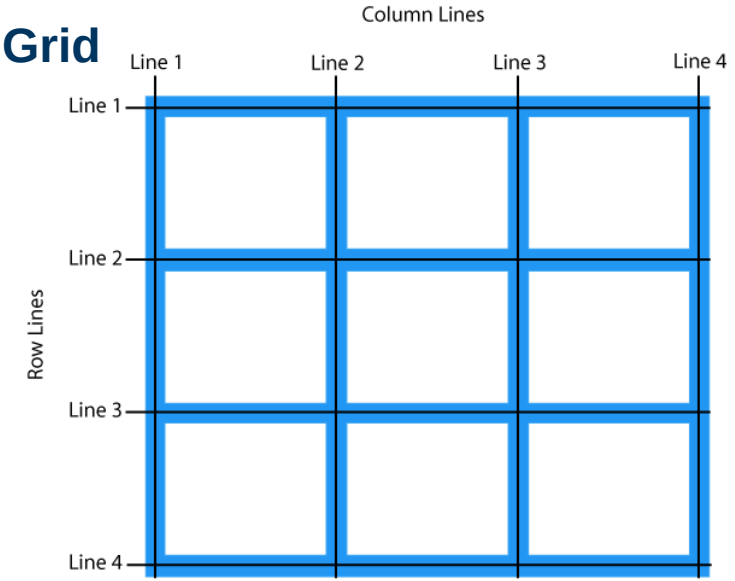
Web-Engineering
CSS Layouts

Flexbox und Grid

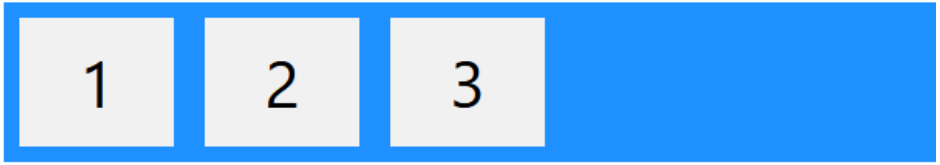
Häufig werden Flexbox oder Grid für die Anordnung der Elemente verwendet

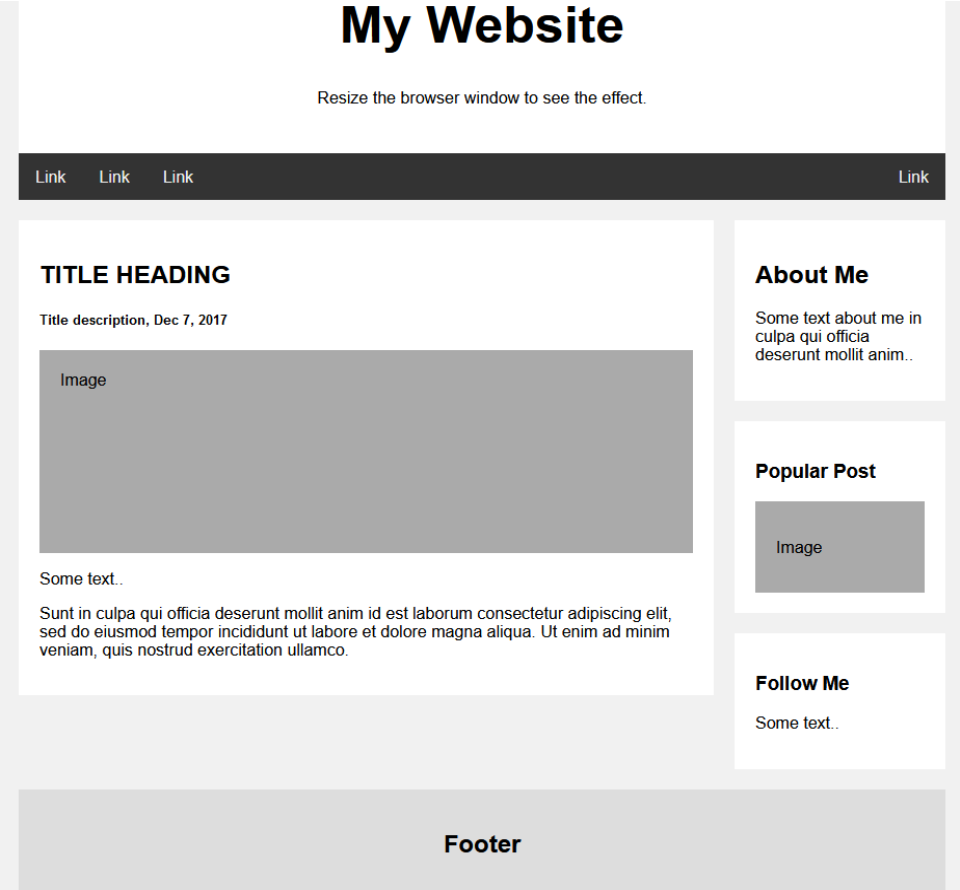
- Ist das Layout für eine Reihe oder Spalte vorgegeben, bietet sich `Flexbox` an
- Wenn das Layout für Reihe und Spalte vorgegeben ist, bietet sich `Grid` an
- Beide Methoden sind nützlich um Elemente schnell und einfacher als beispielsweise mit einer Tabelle anzuordnen

Grid



Flexbox





Quelle https://www.w3schools.com/css/css_website_layout.asp

Flexbox-Layout

Das Flexbox-Layout positioniert Elemente linear in Zeilen oder in Spalten.

`flex` kann als `display`-Eigenschaft angegeben werden, um die `flex`-Funktionalitäten zu nutzen: `display: flex;`

Nun können die Flexbox-Eigenschaften genutzt werden, einige davon sind:

- `justify-content`
  - Legt den Abstand der HTML-Elemente innerhalb eines Flexbox-Containers fest
- `flex-wrap`
  - legt fest ob der Flexbox-Container Inhalt auf mehrere Zeilen verteilt

Flexbox-Layout

Flexbox-Eigenschaften (Fortsetzung)

- `align-items`
  - Gibt an wie die Elemente den vorhandenen Platz im Flexbox-Container nutzen
- `flex:`
  - Hat 3 Parameter `flex-grow`, `flex-shrink` und `flex-basis`

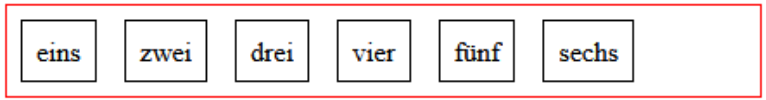
    Über diese Parameter können Angaben zum Verhalten der Elemente bei einer sich ändernden Flexbox-Container-Größe gemacht werden

Flexbox-Layout

CSS Basis für alle Flexbox-Layout Beispiele:

```css
ul {
  list-style-type: none;
  margin: 0px;
  padding: 0px;
  border: 1px solid red;
  display: flex;
}
li {
  border: 1px solid black;
  margin: 5px;
  padding: 5px;
}
```

HTML-Elemente mit Styling:



HTML Basis für alle Flexbox-Layout Beispiele:

```html
<ul>
  <li>eins</li>
  <li>zwei</li>
  <li>drei</li>
  <li>vier</li>
  <li>fünf</li>
  <li>sechs</li>
</ul>
```

HTML-Elemente ohne Styling:

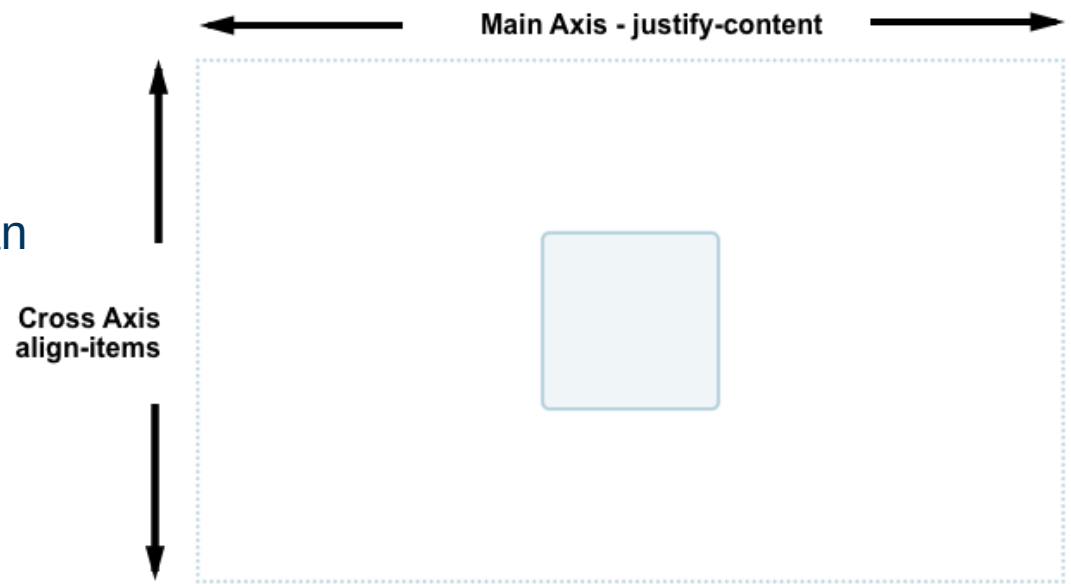
- Ungeordnete Liste `<ul>`
- 6 x Listenitems `<li>`

- eins
- zwei
- drei
- vier
- fünf
- sechs

Flexbox-Layout

`align-items` und `justify-content` werden genutzt um Elemente in der Flexbox zu positionieren.

- `align-items` orientiert sich an der `Cross Axis`
- `justify-content` orientiert sich an der `Main Axis`



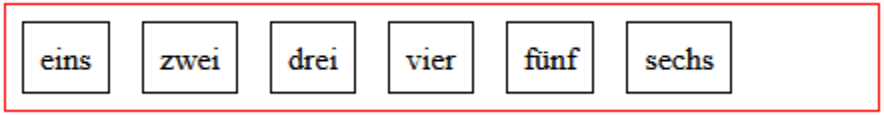
Quelle: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Aligning_Items_in_a_Flex_Container

Flexbox-Layout

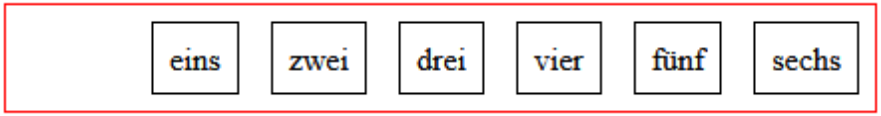
`justify-content` kann verschiedene Werte zum positionieren an der `Main Axis` annehmen. Dabei ist `flex-start` als Standard definiert, weitere sind:

```
ul {... justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | ... ;}
```

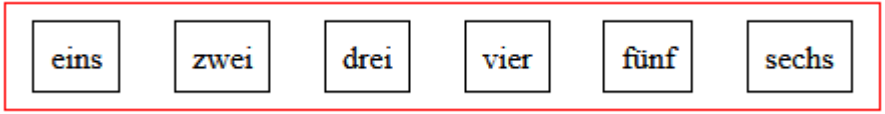
`justify-content: flex-start`



`justify-content: flex-end`



`justify-content: space-evenly`



Flexbox-Layout

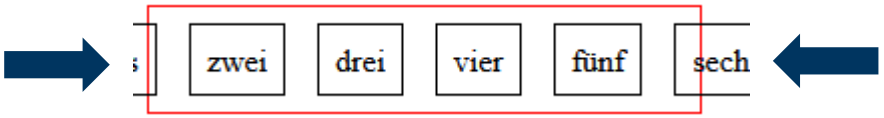
`flex-wrap` gibt die Zeilenaufteilung der Elemente an.

Elemente können bei Platzmangel...

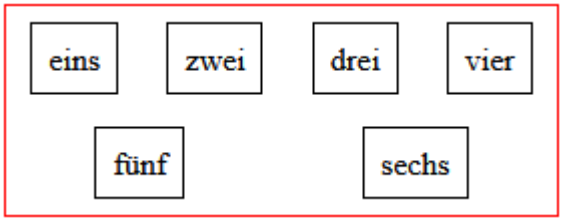
- ... in eine Zeile gelistet werden.
- ... auf mehreren Zeilen aufgeteilt werden.

```
ul {
  ...
  flex-wrap: wrap;
}
```

Problem: Zu viele Elemente in einer Zeile



Lösung: `flex-wrap: wrap`



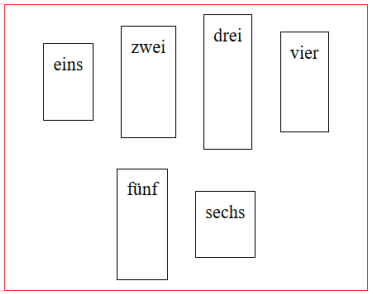
Flexbox-Layout

`Align-items` **kann verschiedene Werte zum positionieren an der** `Cross Axis` **annehmen.**

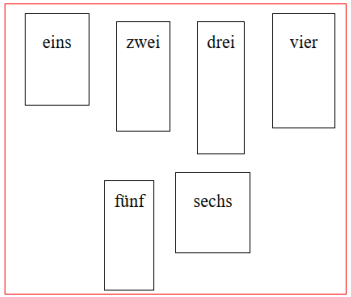
- Für unterschiedliche Höhen der Elemente geeignet
- Dabei ist `stretch` als Standard angegeben

```
ul { ... align-items: stretch | flex-start | flex-end | center |
baseline | first baseline | last baseline | ... ;}
```

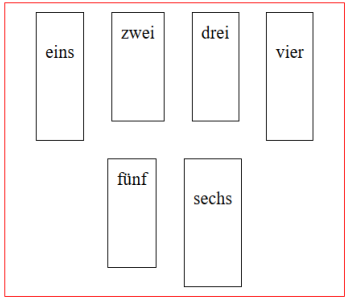
`align-items: center`



`align-items: baseline`



`align-items: stretch`

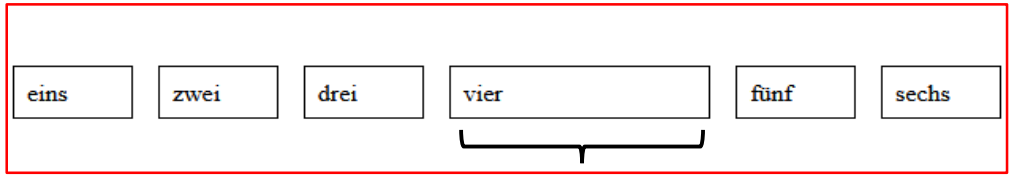


Flexbox-Layout

Die 3 Parameter der `flex`-Eigenschaft

- `flex-grow`
  - Wachstum des Elementes im Verhältnis zu den anderen flexiblen Elementen
- `flex-shrink`
  - Gegensatz zu `flex-grow`
- `flex-basis`
  - Die Länge des Elements

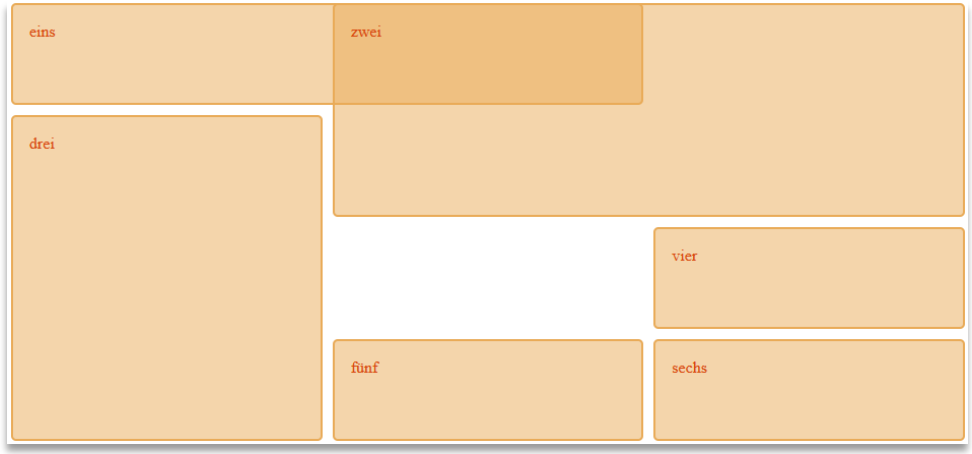
```
/*Alle Elemente außer 4*/
.g1 {
    flex: 0.1 0.1 3em;
}

/*Nur Element 4 */
.g2 {
    flex: 1 1 3em;
}
```



`vier` wächst und schrumpft 10x schneller als die anderen Elemente

Grid-Layout

- Ein Grid-Layout ist ein durch horizontale und vertikale Linien erzeugtes Raster
- Es wird zum positionieren von HTML-Elementen genutzt
- Es besteht aus einem Eltern-Element als Container, mit einem oder mehreren Kind-Elementen als Items



Grid-Layout

Feste und flexible Breite

- Absolute Bemessung
  - `px` → Pixel
  - `em` → Abhängig von Schriftgröße
- Relative Bemessung
  - % → Prozent
  - `fr` → Flexible Länge, nutzt den Rest der Freiräume im Grid-Layout
- Es empfiehlt sich `fr` anstatt `%` innerhalb des Grid-Layouts zu nutzen
  - `fr` bietet in Anbetracht von margin und padding einen erleichterten Umgang

Grid-Layout

Ein zweidimensionales Layout, das Elemente in Zeilen oder in Spalten positioniert.

`grid` kann als `display`-Eigenschaft angegeben werden, um die `grid`-Funktionlitäten nutzen zu können -> `display: grid;`

Nun können die CSS Grid-Eigenschaften genutzt werden, einige davon sind:

- `grid-template-columns`
  - Teilt den verfügbaren Platz der Webseiten Oberfläche in Spalten ein
- `gap, column-gap, row-gap`
  - Legt einen Abstand zwischen den Grid-Elementen fest

Grid-Layout

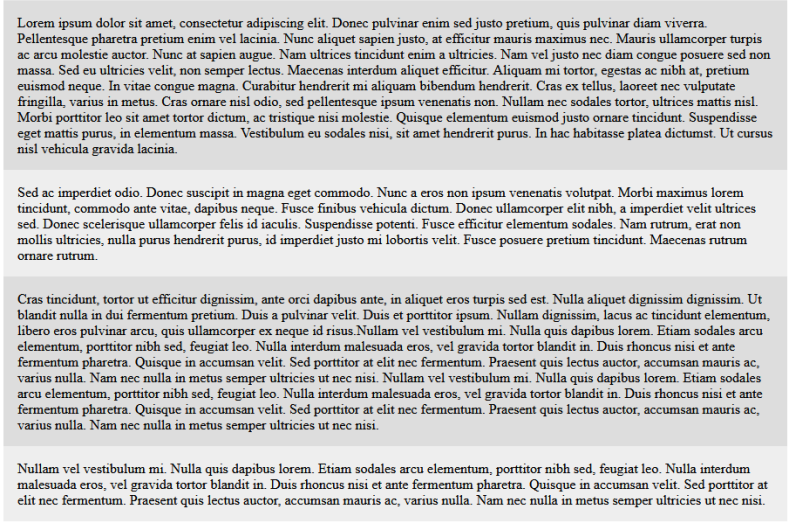
CSS Grid-Eigenschaften (Fortsetzung)

- `grid-auto-rows: minmax(px, auto);`
  - Größe für den Rahmen aller Elemente, bei Bedarf für einzelne mehr Platz
- `grid-column, grid-row`
  - Geben die Spannweite einzelner Elemente auf dem CSS zwölfer-Grid an
- `justify-content` & `align-items`
  - Haben die selbe Funktion wie im Flexbox Layout

Grid-Layout

- Die HTML-Basis bildet in diesem Beispiel ein div-Container (parent) `wrapper` mit inneren `div`-Containern (children)
- Die inneren div-Container enthalten `Lorem Ipsum` Fülltext

```
.wrapper>div {
  background-color: #eee;
  padding: 1em;
}

.wrapper>div:nth-
child(odd) {
  background-color: #ddd;
}
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec pulvinar enim sed justo pretium, quis pulvinar diam viverra. Pellentesque pharetra pretium enim vel lacinia. Nunc aliquet sapien justo, at efficitur mauris maximus nec. Mauris ullamcorper turpis ac arcu molestie auctor. Nunc at sapien augue. Nam ultrices tincidunt enim a ultricies. Nam vel justo nec diam congue posuere sed non massa. Sed eu ultricies velit, non semper lectus. Maecenas interdum aliquet efficitur. Aliquam mi tortor, egestas ac nibh at, pretium euismod neque. In vitae congue magna. Curabitur hendrerit mi aliquam bibendum hendrerit. Cras ex tellus, laoreet nec vulputate fringilla, varius in metus. Cras ornare nisl odio, sed pellentesque ipsum venenatis non. Nullam nec sodales tortor, ultrices mattis nisl. Morbi porttitor leo sit amet tortor dictum, ac tristique nisi molestie. Quisque elementum euismod justo ornare tincidunt. Suspendisse eget mattis purus, in elementum massa. Vestibulum eu sodales nisi, sit amet hendrerit purus. In hac habitasse platea dictumst. Ut cursus nisl vehicula gravida lacinia.

Sed ac imperdiet odio. Donec suscipit in magna eget commodo. Nunc a eros non ipsum venenatis volutpat. Morbi maximus lorem tincidunt, commodo ante vitae, dapibus neque. Fusce finibus vehicula dictum. Donec ullamcorper elit nibh, a imperdiet velit ultrices sed. Donec scelerisque ullamcorper felis id iaculis. Suspendisse potenti. Fusce efficitur elementum sodales. Nam rutrum, erat non mollis ultricies, nulla purus hendrerit purus, id imperdiet justo mi lobortis velit. Fusce posuere pretium tincidunt. Maecenas rutrum ornare rutrum.

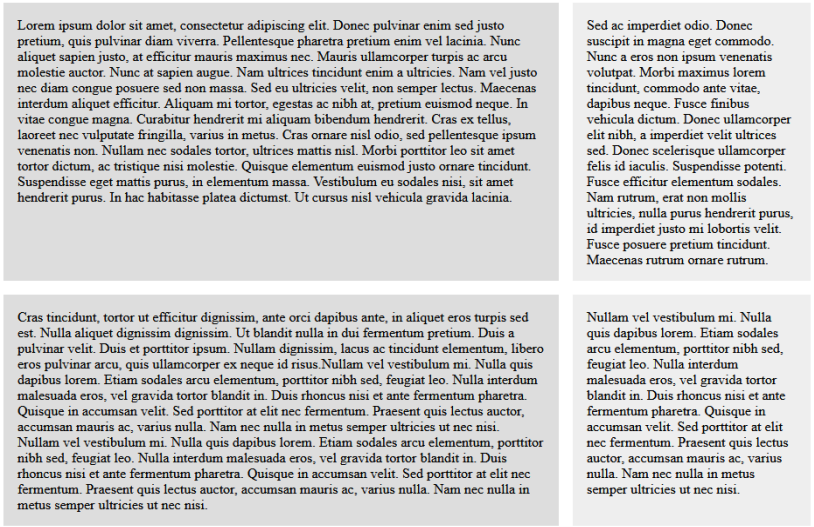
Cras tincidunt, tortor ut efficitur dignissim, ante orci dapibus ante, in aliquet eros turpis sed est. Nulla aliquet dignissim dignissim. Ut blandit nulla in dui fermentum pretium. Duis a pulvinar velit. Duis et porttitor ipsum. Nullam dignissim, lacus ac tincidunt elementum, libero eros pulvinar arcu, quis ullamcorper ex neque id risus.Nullam vel vestibulum mi. Nulla quis dapibus lorem. Etiam sodales arcu elementum, porttitor nibh sed, feugiat leo. Nulla interdum malesuada eros, vel gravida tortor blandit in. Duis rhoncus nisi et ante fermentum pharetra. Quisque in accumsan velit. Sed porttitor at elit nec fermentum. Praesent quis lectus auctor, accumsan mauris ac, varius nulla. Nam nec nulla in metus semper ultricies ut nec nisi. Nullam vel vestibulum mi. Nulla quis dapibus lorem. Etiam sodales arcu elementum, porttitor nibh sed, feugiat leo. Nulla interdum malesuada eros, vel gravida tortor blandit in. Duis rhoncus nisi et ante fermentum pharetra. Quisque in accumsan velit. Sed porttitor at elit nec fermentum. Praesent quis lectus auctor, accumsan mauris ac, varius nulla. Nam nec nulla in metus semper ultricies ut nec nisi.

Nullam vel vestibulum mi. Nulla quis dapibus lorem. Etiam sodales arcu elementum, porttitor nibh sed, feugiat leo. Nulla interdum malesuada eros, vel gravida tortor blandit in. Duis rhoncus nisi et ante fermentum pharetra. Quisque in accumsan velit. Sed porttitor at elit nec fermentum. Praesent quis lectus auctor, accumsan mauris ac, varius nulla. Nam nec nulla in metus semper ultricies ut nec nisi.

Grid-Layout

- Durch die `grid-template-columns`-Anweisung wird die Webseite in zwei Spalten im Verhältnis 7 zu 3 aufgeteilt
- Die `gap`-Anweisung legt einen freien Raum zwischen den einzelnen Elementen fest

```
.wrapper {
  display: grid;
  grid-template-
columns: 7fr 3fr;
  gap: 1em;
}
```



Grid-Layout

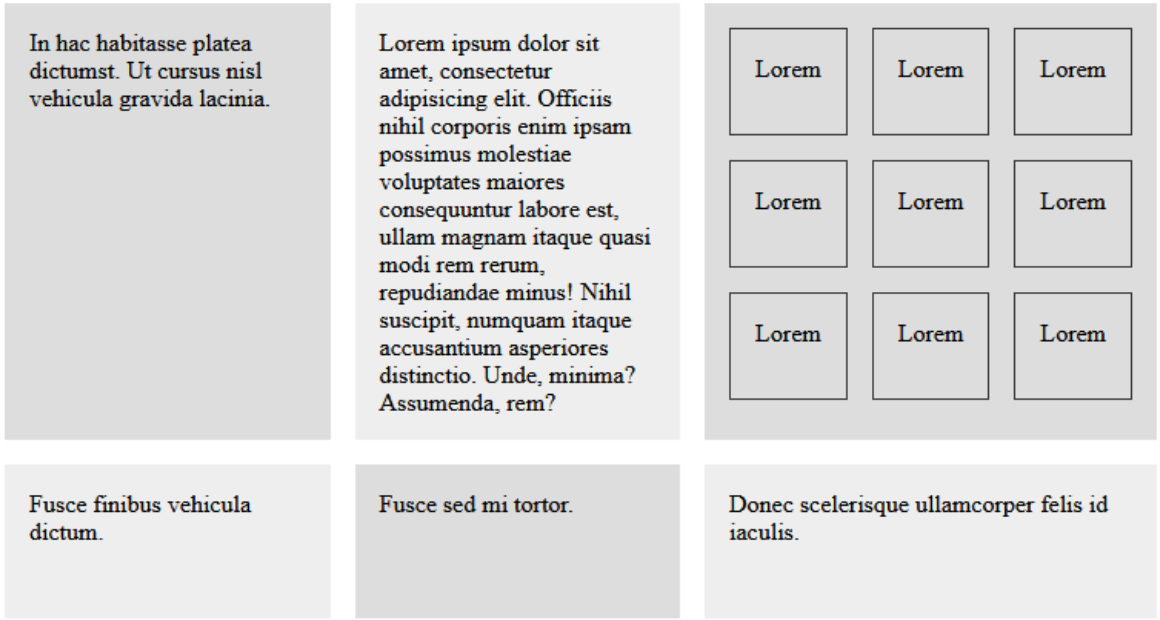
Die Verschachtelung von inneren div-Containern ist ebenfalls möglich.

- Innerhalb des Containers `wrapper` wird neben den bereits erstellten ein weiterer div-Container `verschachtelt`
- Die `repeat`-Anweisung führt eine Zuweisung wiederholt durch – dadurch können mit einem Befehl mehrere Zeilen oder Spalten gestaltet werden
- Die `grid-auto-rows`-Anweisung setzt innerhalb der `minmax`-Anweisung die Größe aller Elemente auf 100px
  - Der Parameter `auto` sorgt bei Bedarf für mehr Platz für die ganze Zeile

```
.wrapper {
  display: grid;
  /*grid-template-
columns:1fr 1fr 1fr;*/
  grid-template-
columns: repeat(3, 1fr);
  grid-gap: 1em;
  grid-auto-
rows: minmax(100px, auto);
}

.verschachtelt {
  display: grid;
  grid-template-
columns: repeat(3, 1fr);
  grid-auto-rows: 70px;
  grid-gap: 1em;
}
```

Grid-Layout

Die Verschachtelung von inneren div-Containern ist ebenfalls möglich.



Grid-Layout HTML Beispiel

Gekürzter HTML-Code zum letzten Beispiel:

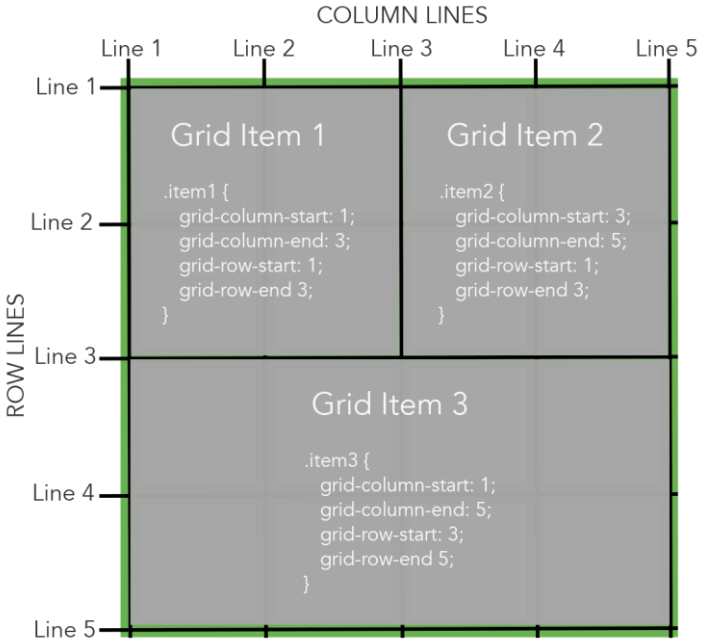
```
<body>
  <main>
    <div class="wrapper">
      <div>In hac habitasse platea dictumst.</div>
      <div>Lorem ipsum itaque accusantium asperiores distinctio.</div>
      <div class="verschachtelt">
        <div>Lorem</div>
        <div>Lorem</div>
        <div>Lorem</div>
      </div>
      <div>Fusce finibus vehicula dictum.</div>
      <div>Fusce sed mi tortor.</div>
      <div>Donec scelerisque ullamcorper felis id iaculis.</div>
    </div>
  </main>
</body>
```

Insgesammt 9x

Grid-Layout

Über `grid-column` & `grid-row` können Angaben gemacht werden, wie weit sich ein Element erstrecken soll.

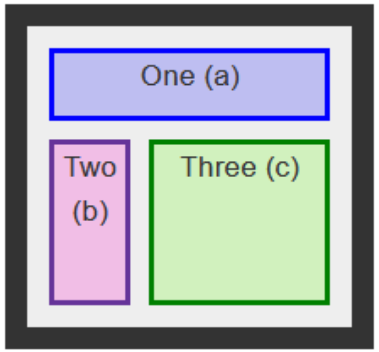
```
.item1 {
    grid-column: 1/3;
    grid-row: 1/3;
}
.item2 {
    grid-column: 3/5;
    grid-row: 1/3;
}
.item3 {
    grid-column: 1/5;
    grid-row: 3/5;
}
```



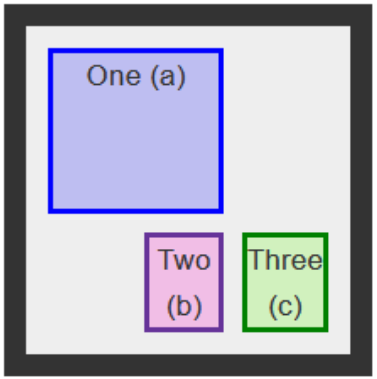
Grid-Layout

Bereiche im Grid-Layout können alternativ mit der Grid-Template-Area benannt und verwaltet werden.

```
grid-template-areas:
                "a a a"
                "b c c"
                "b c c";
```




```
grid-template-areas:
                "a a ."
                "a a ."
                ". b c";
```



Quelle: https://developer.mozilla.org/en-US/docs/Web/CSS/grid-template-areas

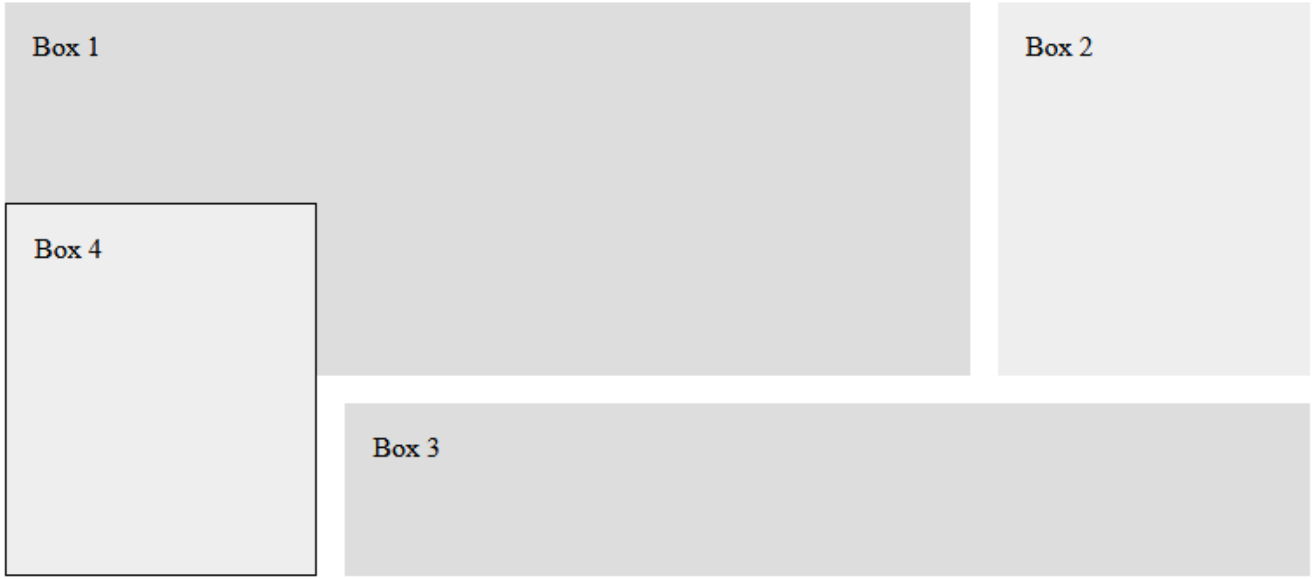
Grid-Layout

Überlappen von Elementen mit `grid-column` & `grid-row`:

```
<body>
  <main>
      <div class="wrapper">
        <div class="box1">Box 1</div>
        <div class="box2">Box 2</div>
        <div class="box3">Box 3</div>
        <div class="box4">Box 4</div>
      </div>
  </main>
</body>
.wrapper {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    grid-auto-rows: minmax(100px, auto);
    gap: 1em;
    justify-items: stretch;
    align-items: stretch;}
```

```
.box1 {
    grid-column: 1/2;
    grid-row: 1/2;
}
.box2 {
    grid-column: 3;
    grid-row: 1/2;
}
.box3 {
    grid-column: 2/3;
    grid-row: 3;
}
.box4 {
    grid-column: 1;
    grid-row: 2/3;
    border: 1px solid;
}
```

Grid-Layout




```
.box1 {
    grid-column: 1/2;
    grid-row: 1/2;
}
.box2 {
    grid-column: 3;
    grid-row: 1/2;
}
.box3 {
    grid-column: 2/3;
    grid-row: 3;
}
.box4 {
    grid-column: 1;
    grid-row: 2/3;
    border: 1px solid;
}
```

Dynamisches Design

Mit dem Aufkommen der verschiedenen mobilen Endgeräte, wie Smartphones oder Tablets, ist für die breite Masse der Benutzer die Wichtigkeit der Computer immer weiter ins Hintertreffen gerückt.

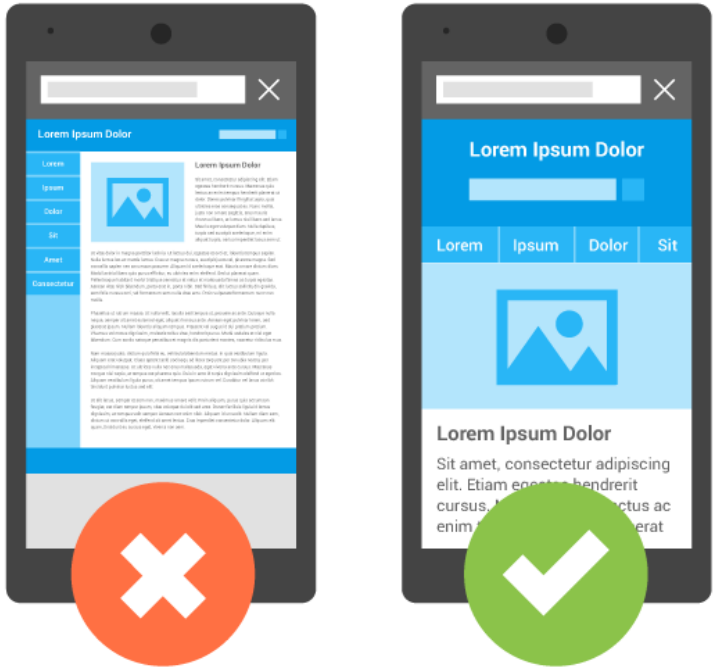
Webseiten sollten auf jedem Gerät akkurat dargestellt werden, ...

- ... jedoch gilt mittlerweile der Ansatz „Mobile First!“.
- Der größte Teil der heutigen Besuche auf Webseiten erfolgen über mobile Endgeräte
- Webseiten können auf vielen verschiedenen Geräten dargestellt werden (Smartphones, Tablets, Monitoren, …) und sollten immer richtig formatiert sein

Dynamisches Design

Responsive Design

- Das Design ändert sich abhängig vom Anzeigegerät
- Sollte im Idealfall, unabhängig vom Design, den selben Informationsgehalt bieten
- Responsive Webdesign (RWD) ist das Konzept zur optimalen Darstellung auf verschiedenen Endgeräten
  - Elemente verkleinern, verschwinden lassen oder neu anordnen



Quelle: https://developers.google.com/search/mobile-sites

Dynamisches Design

Der Viewport ist der sichtbare Teil der Webseite.

- Dieser Teil variiert von Gerät zu Gerät (groß am Monitor, klein am Handy)
- Es ist daher sinnvoll absolute Positionierung von Elementen zu vermeiden!

Wie bereits in der ersten HTML-Vorlesung gezeigt, ist es seit HTML 5 möglich mittels des <meta> tags die Kontrolle über den Viewport zu erlangen.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

"width=device-width" → legt die Breite der Seite fest, die der Bildschirmbreite des Geräts folgt

Dynamisches Design

Breit

Schmal