

03 CSS (Cascading Style Sheets)

Einführung in die moderne (Full-Stack) Webentwicklung mit
JavaScript, Node.js und React.js

Nils Constantin Hellwig
Wissenschaftliche Hilfskraft
RECHENZENTRUM



Universität Regensburg

Überblick Themen

- Wozu benötigt man CSS?
- Wie definiert man CSS-Regeln?
- CSS Box Model
- CSS Styling
- CSS Layouting
- Flexbox

Was ist CSS?

- CSS steht für Cascading Style Sheets
- CSS beschreibt, wie HTML-Elemente dargestellt werden sollen
- Anhand von CSS können Formateigenschaften wie Rahmen, Farben, Hintergrund, Zeichengröße oder Schriftart definiert werden

Was ist CSS?

- Statt das Aussehen jedes HTML-Elements einzeln zu definieren, können mit CSS Regeln definiert werden, die auf verschiedene Elemente angewendet werden. Regeln müssen somit nicht für jedes Element einzeln definiert werden.
- Layout von Webseiten auf verschiedenen Geräten anpassen: Regeln können für verschiedene Bildschirmgrößen festgelegt werden, damit eine Webseite auf allen Geräten angenehm zu benutzen ist und gut aussieht

CSS Syntax

Eine CSS-Regel besteht immer aus einem **Selektor** und einem **Deklarationsblock**:

Selektor
↓
p {

Eigenschaft *Wert*
↓ ↓
font-size: 12px;
color: blue;

Deklarationsblock

}

CSS Selektoren: Element Selektor

Der Elementselektor wählt HTML-Elemente anhand des Elementnamens aus:

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

Hier werden alle `<p>` - Elemente auf der Seite mittig ausgerichtet und der Text wird blau eingefärbt

CSS Selektoren: id-Selektor

Der id-Selektor ("#") verwendet das id-Attribut eines HTML-Elements, um ein bestimmtes Element auszuwählen. Die id eines Elements ist innerhalb einer Seite eindeutig, daher wird der id-Selektor verwendet, um ein eindeutiges Element auszuwählen.

```
#search-bar {  
    text-align: center;  
    color: blue;  
}
```

Die CSS-Regel wird auf das HTML-Element mit `id="search-bar"` angewendet.

CSS Selektoren: class-Selektor

Der Klassenselektor (".") wählt HTML-Elemente mit einem bestimmten Klassenattribut aus.

```
.product-card {  
    text-align: center;  
    background-color: black;  
}
```

In diesem Beispiel haben alle HTML-Elemente mit `class="product-card"` einen schwarzen Hintergrund und sind mittig ausgerichtet.

CSS Selektoren: class-Selektor

Auch möglich: Klassenselektor (".") wählt div-Elemente mit einem bestimmten Klassenattribut aus.

```
div.product-card {  
    text-align: center;  
    background-color: black;  
}
```

In diesem Beispiel haben alle div-Elemente mit `class="product-card"` einen schwarzen Hintergrund und sind mittig ausgerichtet.

CSS Selektoren: Universal Selektor

Der Universalselektor ("*") wählt alle HTML-Elemente auf einer Seite aus.

```
* {  
    text-align: center;  
    background-color: black;  
}
```

CSS Selektoren: Gruppierungsselektor

Der Gruppierungsselektor wählt mehrere HTML-Elemente auf einmal aus.

```
h1, h2, p {  
    text-align: center;  
    background-color: black;  
}
```

Wie bindet man ein CSS Style Sheet ein?

Drei Möglichkeiten zum Einfügen von CSS:

- External CSS
- Internal CSS
- Inline CSS

External CSS

- Externe Stylesheets werden im `<link>`-Element innerhalb des `<head>`-Abschnitts einer HTML-Seite definiert.

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="style.css"/>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

Internal CSS

Wird im `<style>` - Element innerhalb des
`<head>` - Abschnitts einer HTML-Seite definiert.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {
        background-color: linen;
      }

      h1 {
        color: maroon;
        margin-left: 40px;
      }
    </style>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

Inline CSS

Inline Styles werden im `style` - Attribut des jeweiligen Elements definiert.

```
<!DOCTYPE html>
<html>
  <body>
    <h1 style="color: blue; text-align: center;">This is a heading</h1>
    <p style="color: red">This is a paragraph.</p>
  </body>
</html>
```

Mehrere Stylesheets

Wenn Eigenschaften für denselben Selektor (Element) in verschiedenen Stylesheets definiert wurden, wird der Wert aus der zuletzt gelesenen Stilvorlage verwendet.

```
<head>
  <link rel="stylesheet" type="text/css" href="style.css" /> ←
  <style>
    h1 {
      color: orange;
    }
  </style>
</head>
<head>
  <h1>Hallo Welt</h1> ← Textfarbe: Orange
</head>
```



CSS Comments

```
/* This is a single-line comment */
```

```
.p {  
    background-color: black;  
}
```

```
/* This is a  
multi-line comment */
```

```
.p {  
    font-size: 14px;  
}
```

```
.p {  
    color: red; /* Set text color to red */  
}
```

CSS Colors

Farben werden mit vordefinierten Farbnamen oder mit RGB-, HEX-, HSL-, RGBA- und HSLA-Werten angegeben.

- CSS Color Names (z.B. blue, red, white, black): https://www.w3schools.com/colors/colors_names.asp
- HEX Values: z.B. **#ff6347**
- RGB(A) (rot, grün, blau):

`rgb (73, 86, 40)`

`rgba (73, 86, 40, 0.5)`

CSS Colors

Es gibt viele Deklarationen, die eine Farbangabe erfordern:

- **background-color**: Farbe des Hintergrunds eines Elements
- **color**: Farbe des Textes
- **border**: Elementrahmen (einfärben)

```
p {  
    border: 2px solid red;  
}
```

CSS Variables

- CSS Variables (CSS Custom Properties) ermöglichen es, individuelle Variablen in CSS-Stylesheets zu verwenden. Diese Variablen können dann im gesamten Stylesheet oder sogar in anderen Stylesheets verwendet werden, um CSS-Eigenschaften dynamisch zu ändern.
- Eine CSS-Variable wird mit dem Doppel-Unterstrich-Symbol (--) definiert und kann überall im Stylesheet verwendet werden, indem sie mit **var()** aufgerufen wird.
- Beispiel:

```
:root {  
  --main-color: blue;  
}  
h1 {  
  color: var(--main-color);  
}
```

CSS Variables

- Auf **globale Variablen** kann im gesamten Dokument zugegriffen werden
- Auf **lokale Variablen** kann nur innerhalb des Selektors, in dem sie deklariert wurden, zugegriffen werden
- Um eine Variable mit globalem Geltungsbereich zu erstellen, muss man diese innerhalb des Selektors `:root` deklarieren. Der `:root`-Selektor entspricht dem Wurzelelement des Dokuments (In HTML ist das Wurzelelement immer das `<html>`-Element.).

CSS Units

- In CSS gibt es verschiedene Einheiten für die Angabe einer Größe
- Viele CSS-Eigenschaften nehmen "Größen"-Werte an, z.B. `font-size`
- Größen werden definiert mit einer Zahl, gefolgt von einer Einheit, z.B. `60px`
- Beispiel:

```
h1 {  
    font-size: 60px;  
}
```

CSS Units: Absolute Größen

- **px:** Pixel (px) sind relativ zum Gerät. Bei Geräten mit niedriger Pixeldichte entspricht 1px in der Regel einem Gerätepixel (Punkt) der Anzeige. Für hochauflösendere Bildschirme bedeutet 1px mehrere Gerätepixel (Mehr dazu: <https://elad.medium.com/understanding-the-difference-between-css-resolution-and-device-resolution-28acae23da0b> - „A pixel is not a pixel“).
- **cm, mm, in, pt** ($1\text{ pt} = 1/72\text{ in}$)

CSS Units: Relative Größen

- **em:** Relativ zur Schriftgröße des Elements (2em bedeutet die 2-fache Größe der Schrift des Elements)

Beispiel:

```
<p class="example">Dies ist ein Beispieltext.</p>
```

```
.example {  
  font-size: 16px; /* Schriftgröße */  
  margin-bottom: 1em; /* Abstand zum nächsten Element: 1-fache Größe der Schrift */  
}
```


CSS Units: Relative Größen

- **rem**: CSS-Einheit, die relativ zur Schriftgröße des Wurzelements (dem `<html>`-Element) ist.

Beispiel:

```
html {  
  font-size: 16px;  
}  
  
h1 {  
  font-size: 2rem;  
}
```

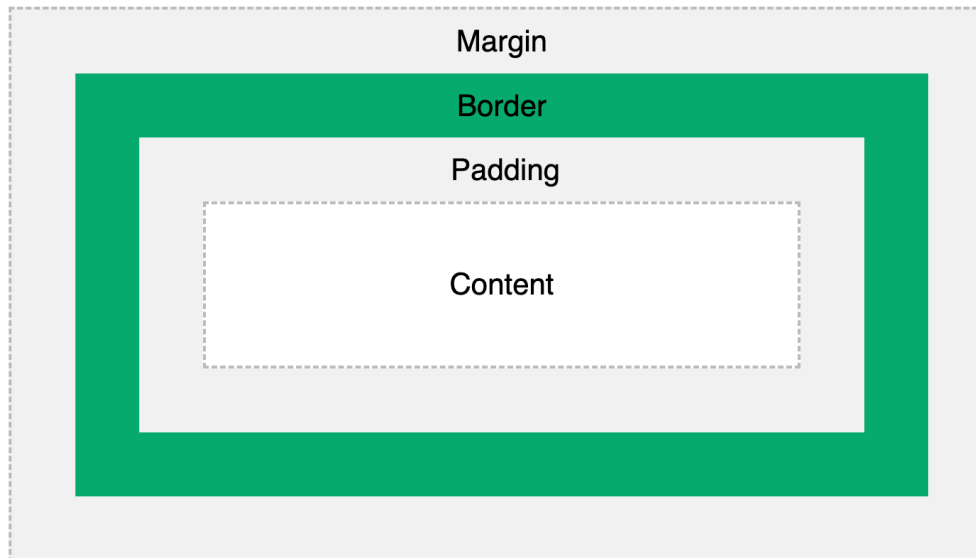
CSS Units: Relative Größen

- **vw**: Relativ zu 1% der Breite des Viewports (1vw = 1% der Breite des Viewports)
- **vh**: Relativ zu 1% der Höhe des Viewports (1vh = 1% der Höhe des Viewports)
- **vmin**: Relativ zu 1% der kleineren Dimension des Viewports (Beispielsweise bei einem iPhone wäre dies die Breite) (1 vmin = 1% der kleineren Dimension des Viewports)
- **vmax**: Relativ zu 1% der größeren Dimension des Viewports (Beispielsweise bei einem iPhone wäre dies die Höhe) (1 vmax = 1% der größeren Dimension des Viewports)
- **%**: Relativ zum übergeordneten (parent) Element

```
<section> ← font-size: 32px;
  <p>This is an example...</p> ← font-size: 50%; /* = 16px */
</section>
```

CSS Box Model

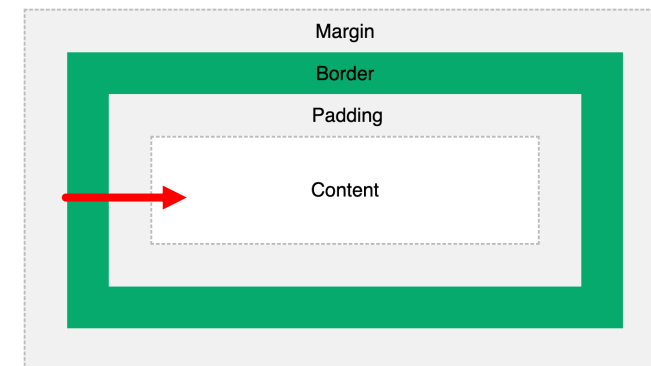
- Alle HTML-Elemente können als Boxen betrachtet werden
- Das CSS Box Model ist im Wesentlichen eine Box, die jedes HTML-Element umschließt. Es besteht aus: `margin`, `border`, `padding` und `content`



CSS Box Model: Content

- Der Inhalt des HTML-Elements (z.B. Text, Bilder oder verschachteltes Element)
- Breite und Höhe können mittels `height` und `width` festgelegt werden (`auto` ist der default-Wert, der Browser berechnet Höhe und Breite)
- Beispiel:

```
div {  
  height: 200px;  
  width: 50%;  
}
```



CSS Box Model: Padding

- Innenabstand eines HTML-Elementes zu seinem Rahmen.
- Die `padding`-Eigenschaft kann auf verschiedene Arten verwendet werden, um den Innenabstand zu definieren:

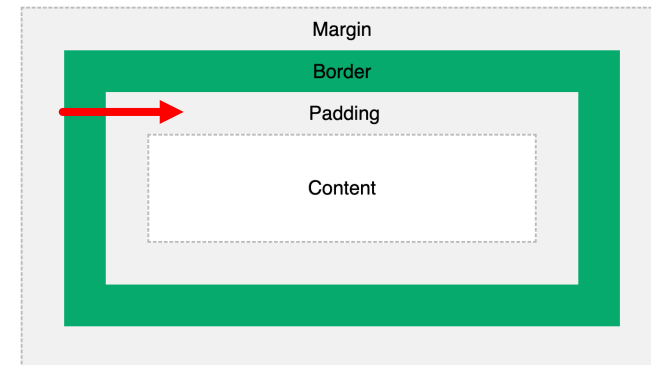
```
/* Innenabstand für alle Seiten */  
padding: 20px;
```

```
/* Innenabstand für [oben und unten], [links und rechts] */  
padding: 20px 40px;
```

```
/* Innenabstand für [oben], [rechts und links], [unten] */  
padding: 20px 40px 30px;
```

```
/* Innenabstand für [oben], [rechts], [unten] und [links] */  
padding: 20px 40px 30px 50px;
```

- Separate Eigenschaften für jede Seite: **`padding-top`**, **`padding-right`**, **`padding-bottom`** und **`padding-left`**.

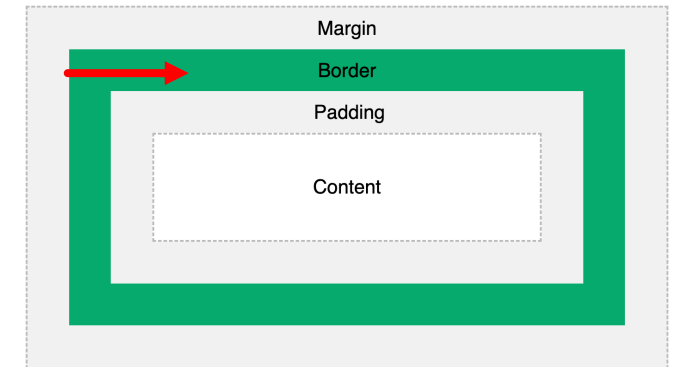


CSS Box Model: Border

- Rahmen um den Padding und Content
- Die **border**-Eigenschaft kann auf folgende Weise verwendet werden, um die Breite, Farbe und Stil der Linie zu definieren:

```
div {  
  border: 2px solid black;  
}
```

- Es gibt auch separate Eigenschaften für jede Seite des Randes:
border-top, **border-right**, **border-bottom** und **border-left**



CSS Box Model: Margin

- Margin beschreibt den Abstand außerhalb eines HTML-Elementes.
- Der Margin ist Transparent.
- Die `margin`-Eigenschaft kann auf verschiedene Arten verwendet werden, um den Außenabstand zu definieren.

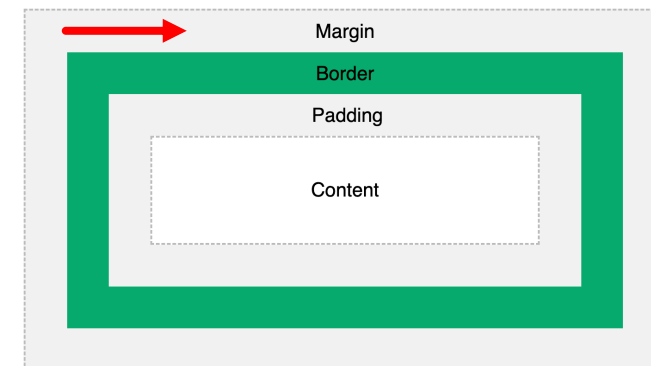
```
/* Außenabstand für alle Seiten */  
margin: 20px;
```

```
/* Außenabstand für [oben und unten], [links und rechts] */  
margin : 20px 40px;
```

```
/* Außenabstand für [oben], [rechts und links], [unten] */  
margin : 20px 40px 30px;
```

```
/* Außenabstand für [oben], [rechts], [unten], [links] */  
margin : 20px 40px 30px 50px;
```

- Separate Properties für jede Seite: **`margin-top`**, **`margin-right`**, **`margin-bottom`** und **`margin-left`**.



CSS Box Model

The screenshot displays the Chrome DevTools interface. The 'Elemente' (Elements) panel on the left shows the DOM tree with the following structure:

```
<!DOCTYPE html>
<html lang="de" data-lt-installed="true">
  <head>...</head>
  <body class="page" data-uid="1100186">
    <div class="skipnav">...</div>
    <div class="container area-da3c43">
      ::before
      <a href="https://www.uni-regensburg.de" class="logo">...</a>
      <div class="mobile-page-nav-bar">...</div>
      <div class="header">...</div>
      <div class="link-back">...</div>
      <div class="menu-left">...</div>
      <div class="left">
        <h1 style="width: 600px;">Willkommen auf der Webseite der Fakultät für Informatik und Data Science</h1>
      </div>
      <div class="article" id="c30656">...</div>
      <div class="article" id="c90789">...</div>
      <div class="footer">...</div>
    </div>
    <div class="right" data-uid="1547">...</div>
    <div class="mobile-menu">...</div>
    ::after
  </div>
  <div class="navigation">...</div>
  <script src="/typo3conf/ext/powermail/Resources/Public/JavaScript/Libraries/jquery.dataTables.min.js?1627416605"></script>
  <script src="/typo3conf/ext/powermail/Resources/Public/JavaScript/Libraries/jquery.dataTables.min.js?1627416605"></script>
</html>
```

The 'Stile' (Styles) panel on the right shows the 'Filtern' (Filter) dropdown set to ':hov .cls'. The list of styles includes various color and background properties, such as:

- lt color white: -- #fff !important;
- lt color black: -- #111 !important;
- lt color transparent: -- rgba(255, 255, 255, 0) !important;
- lt color background light: -- var(--lt color gray 100) !important;
- lt color background default: -- var(--lt color gray 200) !important;
- lt color background dark: -- var(--lt color gray 300) !important;
- lt color border light: -- var(--lt color gray 200) !important;
- lt color border default: -- var(--lt color gray 300) !important;
- lt color border dark: -- var(--lt color gray 400) !important;
- lt color text very light: -- var(--lt color gray 500) !important;
- lt color text light: -- var(--lt color gray 600) !important;
- lt color text default: -- var(--lt color gray 700) !important;
- lt color text dark: -- var(--lt color gray 800) !important;
- lt color overlay default: -- #fff !important;
- lt color overlay dark: -- #fff !important;
- lt color overlay transparent: -- rgba(0, 0, 0, 0.1) !important;
- lt shadow website overlay: -- 0 0 7px 0 rgba(0, 0, 0, 0.3) !important;

At the bottom right, a diagram illustrates the CSS Box Model for a selected element. The diagram shows a central blue box with a width of 600px and a height of 19px. This box is surrounded by a green padding area (10px on all sides), which is further enclosed by an orange border area (4px on all sides). The total width of the box model is 620px (600px + 2 * 10px padding + 2 * 4px border), and the total height is 23px (19px + 2 * 10px padding + 2 * 4px border).

CSS Backgrounds

- **background-color**: Hintergrundfarbe eines HTML-Elements

```
div {  
    background-color: rgba(0, 128, 0, 0.3); /* Green background with 30% opacity*/  
}
```

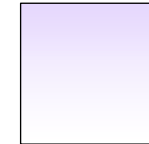
CSS Backgrounds

- **background-image**: Spezifiziert ein Bild, das als Hintergrund eines HTML-Elements verwendet werden soll

```
div {  
    background-image: url("background.jpg");  
}
```

CSS Backgrounds

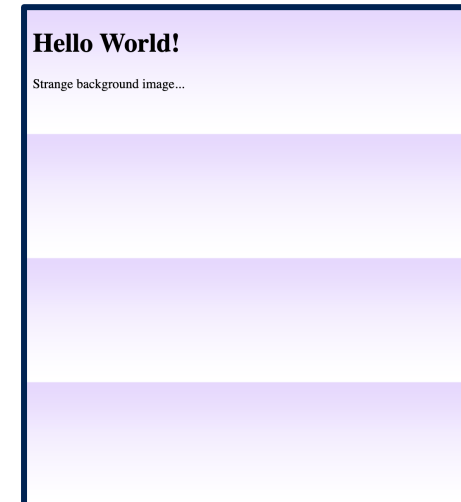
- Mit der Eigenschaft **background-image** wird ein Bild standardmäßig sowohl horizontal als auch vertikal wiederholt
- Es kann vorkommen, dass Bilder nur horizontal oder vertikal wiederholt werden sollen, da sie ansonsten seltsam aussehen
- **background-repeat**: Wiederholung eines Hintergrundbildes festlegen
- Werte von **background-repeat**: `repeat` (default), `repeat-x`, `repeat-y`, `no-repeat`



`background.jpg`

`repeat`

`repeat-x`



CSS Backgrounds

- **opacity**: Transparenz eines HTML-Elements
- Ein Wert von 1 bedeutet, dass das Element vollständig sichtbar ist und ein Wert von 0 bedeutet, dass das Element vollständig unsichtbar ist
- Beispiel: Transparenz des eines Elements auf 50% festlegen:

```
div {  
    opacity: 0.5;  
}
```

CSS Text Formatierung

- **color**: Farbe des Textes
- **text-align**: horizontale Ausrichtung eines Textes festlegen

```
h1 {  
  text-align: center;  
}  
h2 {  
  text-align: left;  
}  
h3 {  
  text-align: right;  
}  
p {  
  text-align: justify;  
}
```

```
<h1>Heading 1 (center)</h1>  
<h2>Heading 2 (left)</h2>  
<h3>Heading 3 (right)</h3>  
<p>Lorem ipsum dolor sit amet, ...</p>
```

Heading 1 (center)

Heading 2 (left)

Heading 3 (right)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus sagittis purus a lacus tincidunt, sed placerat nisi interdum. Maecenas dictum, ipsum ac sollicitudin facilisis, ligula enim consequat nulla, in mollis nunc ligula eget purus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. In dictum dictum tellus, eu ornare urna tincidunt quis. Integer sed magna porttitor, tincidunt neque a, feugiat quam. Aliquam a dictum eros, a luctus diam. Nullam rutrum turpis ut turpis pretium, id consectetur urna scelerisque. Proin sollicitudin lacus in tellus ultricies, eu suscipit quam ultricies.

CSS Text Formatierung

- **text-decoration:** Text mit Linien unterstreichen, durchstreichen oder überstreichen
Werte: `overline`, `underline`, `line-through`, `none` (default), auch mehrere davon möglich.
- Spezifikation des Styles der Linie: `solid`, `double`, `dotted`, `dashed`, `wavy`
- Farbe und Dicke der Linie können ebenfalls spezifiziert werden

```
h3 {  
    text-decoration: underline 4px green double;  
}
```

This is some text.

- Diese vier Properties können auch einzeln spezifiziert werden mithilfe von
`text-decoration-line`, `text-decoration-thickness`, `text-decoration-color`,
`text-decoration-style`

CSS Layout: **display**

- **display**: gibt an, ob/wie ein Element angezeigt wird
- Jedes HTML-Element hat einen default-Wert für **display**, bei den meisten Elementen ist dies entweder `block` oder `inline`
- Ein Element auf Blockebene beginnt immer in einer neuen Zeile und nimmt die gesamte verfügbare Breite ein.

Beispiele: `<div>`, `<h1>` bis `<h6>`, `<p>`, `<form>`

- Ein Inline-Element beginnt nicht in einer neuen Zeile und nimmt nur so viel Breite wie nötig ein.

Beispiele: ``, `<a>`, ``

- Auch `none` ist ein valider Wert von **display** und wird häufig in JavaScript verwendet, um Elemente aus- und einzublenden, ohne sie zu löschen und neu zu erstellen.

CSS Layout: `position`

- Die Eigenschaft `position` gibt die Art der für ein Element verwendeten Positionierungsmethode an
- Mögliche Werte: `fixed`, `static`, `relative`, `absolute` und `sticky`
- HTML-Elemente haben standardmäßig den Wert `static` für die Eigenschaft `position`

CSS Layout: `position`

1. `Position: fixed`

- Ein Element mit `position: fixed;` ist relativ zum Viewport positioniert, d.h. es bleibt immer an der gleichen Stelle, auch wenn auf der Seite gescrollt wird.
- Die Eigenschaften `top`, `right`, `bottom` und `left` werden zur Positionierung des Elements verwendet.

CSS Layout: `position`

2. `Position: static` (Standardwert)

- Elemente mit `position: static;` werden von den Eigenschaften `top`, `right`, `bottom` und `left` nicht beeinflusst
- Ein Element mit `position: static;` wird nicht auf besondere Weise positioniert; es wird immer entsprechend dem normalen Fluss der Seite positioniert

CSS Layout: `position`

3. `Position: relative`

- Ein Element mit `position: relative;` wird relativ zu seiner normalen Position positioniert
- Wenn die Eigenschaften `top`, `bottom`, `left` oder `right` verwendet werden, um das Element von seiner normalen Position zu verschieben, wird es relativ zu der Stelle verschoben, an der es normalerweise (wie bei `static`) im Dokumentlayout erscheinen würde
- Beispiel: https://www.w3schools.com/css/tryit.asp?filename=trycss_position_relative

CSS Layout: `position`

4. `Position: absolute`

- Ein Element mit `position: absolute;` wird relativ zum nächstgelegenen **positionierten Vorgänger** positioniert (und nicht wie bei `fixed` relativ zum Viewport).
- Beispiel: https://www.w3schools.com/css/tryit.asp?filename=trycss_position_absolute

CSS Layout: position

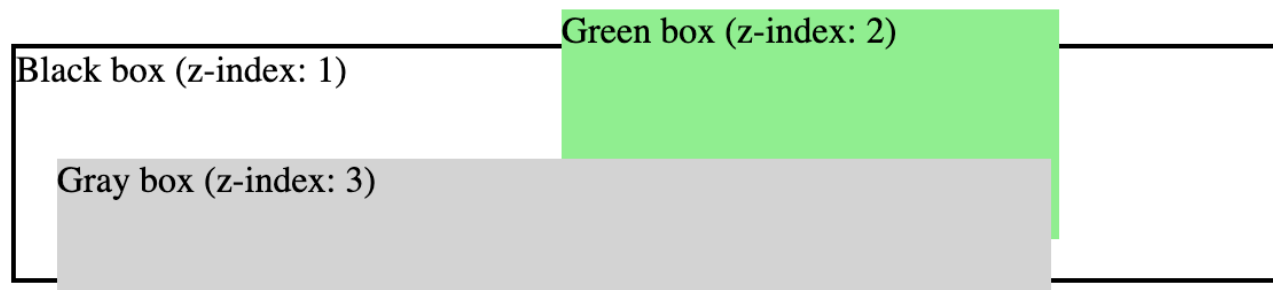
5. Position: sticky

- Ein Element mit `position: sticky;` wird basierend auf der Scroll-Position im Browser positioniert.
- Ein sticky-Element schaltet je nach Bildlaufposition zu `fixed` um
- Es wird relativ positioniert, bis eine bestimmte Position im Ansichtsfenster erreicht wird - dann "klebt" es an seinem Platz (wie `position: fixed`).
- In diesem Beispiel „klebt“ das `sticky`-Element am oberen Rand der Seite (`top: 0`), wenn Sie seine Bildlaufposition erreichen:

https://www.w3schools.com/css/tryit.asp?filename=trycss_position_sticky

CSS Layout: `z-index`

- Wenn Elemente positioniert werden, können sie andere Elemente überlappen
- Die Eigenschaft `z-index` gibt die Stapelreihenfolge eines Elements an (welches Element vor oder hinter den anderen platziert werden soll)



CSS Layout: overflow

- Die CSS-Eigenschaft `overflow` steuert, was mit dem Content geschieht, wenn dieser zu groß ist, um in einen Bereich zu passen:

`visible`

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

`hidden`

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what

`scroll`

to have better control of the layout. The overflow property specifies what happens if content overflows an element's

`auto`

to have better control of the layout. The overflow property specifies what happens if content overflows an element's

Der Wert `auto` ist ähnlich wie `scroll`, fügt aber nur bei Bedarf eine Scrollbar hinzu

```
div {  
  width: 200px;  
  height: 65px;  
  background-color: coral;  
  overflow: [...];  
}
```

CSS Pseudo-classes

- Eine Pseudoklasse wird verwendet, um einen speziellen Zustand eines Elements zu definieren.
- Syntax:

```
selector:pseudo-class {  
    property: value;  
}
```


CSS Pseudo-classes

- Beispiele:

```
/* unvisited link */
a:link {
  color: #ff0000;
}

/* visited link in the past */
a:visited {
  color: #00ff00;
}

/* mouse hover */
a:hover {
  color: #000000;
}

/* selected link (click is still active) */
a:active {
  color: #0000ff;
}
```

Task hinzufügen

Task hinzufügen



CSS Text Shadow

- Die `text-shadow` Eigenschaft wird in CSS verwendet, um einen Schatteneffekt zu einem Text hinzuzufügen
- Beispiel:

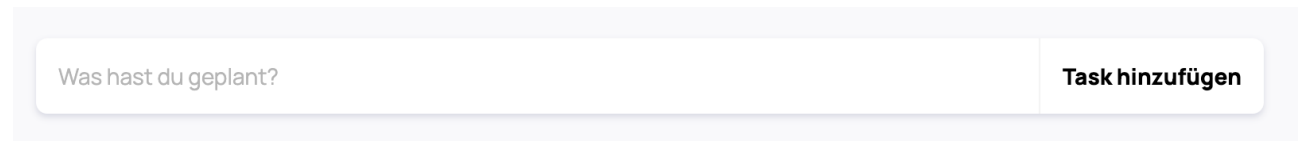
```
h1 {  
  text-shadow: 2px 2px 5px red;  
}
```

Text shadow effect!

- Der Schatten ist 2 Pixel horizontal und 2 Pixel vertikal von dem Text versetzt
- Der Schatten hat einen Unschärferadius von 5 Pixel
- Der Schatten ist rot

CSS Box Shadow

- Die `box-shadow` Eigenschaft wird in CSS verwendet, um einen Schatteneffekt zu einem Element hinzuzufügen
- Beispiel:



```
.shadowed {  
  box-shadow: 0 2px 4px rgba(0, 0, 82, 0.15);  
}
```

- Der Schatten ist 0 Pixel horizontal und 2 Pixel vertikal von dem Element versetzt
- Der Schatten hat einen Unschärferadius von 4 Pixel
- Der Schatten hat eine Farbe von `rgba(0, 0, 82, 0.15)`, was eine leicht transparente Farbe von Blau ist

CSS Flexbox

- Flexbox ist ein Layoutmodell, um die Anordnung von Elementen in Benutzeroberflächen zu vereinfachen
- In CSS ist `flex` ein Wert für das `display`-Attribut, das verwendet wird, um ein Element in ein flexibles Box-Layout zu verwandeln
- Der Hauptzweck besteht darin, einem Container (Parent-Element) die Möglichkeit zu geben, die Reihenfolge seiner Kinder-Elemente zu ändern und um den verfügbaren Platz bestmöglich auszufüllen

CSS Flexbox

- Flexbox ist richtungsunabhängig, im Gegensatz zu den normalen Layouts (`block`, das vertikal ausgerichtet ist, und `inline`, das horizontal ausgerichtet ist)
- Da es sich bei Flexbox um ein ganzes Modul und nicht um eine einzelne Eigenschaft handelt, umfasst es einige Properties

CSS Flexbox: Basics

- Einige Eigenschaften von Flexbox sind für den Container (übergeordnetes Element, bekannt als "flex container") bestimmt, während andere für die untergeordneten Elemente (die sogenannten "flex items") bestimmt sind.

Flex Container

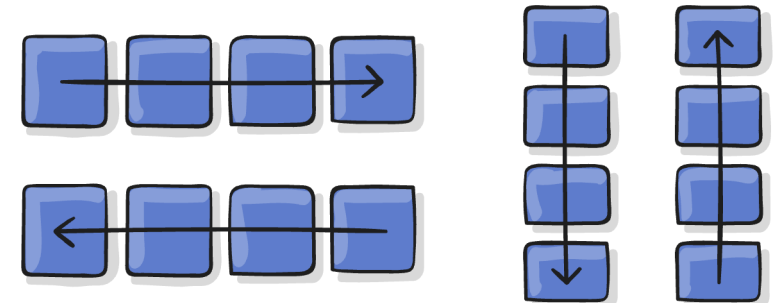


```
<div class="course-list">  
  <div class="course-element"></div>  
  <div class="course-element"></div>  
  <div class="course-element"></div>  
</div>
```

Flex Items

CSS Flexbox: `flex-direction`

- `flex-direction` ist eine Eigenschaft für das Parent-Element, die festlegt, in welche Richtung sich das Layout ausrichtet
- Es gibt vier mögliche Werte:
 1. **row** (Standardwert): Das Layout richtet sich in horizontaler Richtung aus, von links nach rechts
 2. **row-reverse**: Das Layout richtet sich in horizontaler Richtung aus, von rechts nach links
 3. **column**: Das Layout richtet sich in vertikaler Richtung aus, von oben nach unten
 4. **column-reverse**: Das Layout richtet sich in vertikaler Richtung aus, von unten nach oben



CSS Flexbox: flex-direction

- Beispiel:

Hauptachse



Katholische Theologie
Rechtswissenschaft
Wirtschaftswissenschaften
Medizin
Philosophie, Kunst-, Geschichts- und Gesellschaftswissenschaften
Humanwissenschaften
Sprach-, Literatur- und Kulturwissenschaften
Mathematik
Physik
Informatik und Data Science
Biologie und Vorklinische Medizin
Chemie und Pharmazie

```
.faculty-list {  
    display: flex;  
    flex-direction: column;  
}
```


CSS Flexbox: `flex-wrap`

- Standardmäßig werden alle Flex-Elemente versuchen, in eine Zeile zu passen
- Man kann dieses Verhalten jedoch mit der Eigenschaft `flex-wrap` ändern und den Elementen erlauben, sich automatisch auf mehrere Zeilen zu verteilen
- Die Eigenschaft `flex-wrap` gibt an, wie Flex-Elemente in mehrere Zeilen umgebrochen werden sollen, wenn sie nicht genügend Platz in der aktuellen Zeile haben

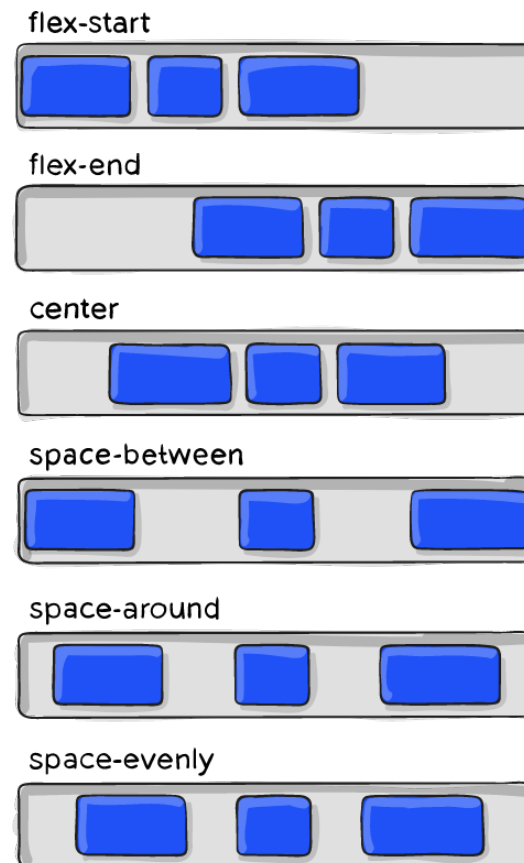
Demo: <https://css-tricks.com/almanac/properties/f/flex-wrap/>

CSS Flexbox: **flex-wrap**

- Mögliche Werte von **flex-wrap**:
 1. **nowrap** (Standardwert): Flex-Elemente werden nicht umgebrochen und bleiben alle in einer Zeile
 2. **wrap**: Flex-Elemente werden umgebrochen und in mehrere Zeilen angeordnet, wenn sie nicht genügend Platz in der aktuellen Zeile haben
 3. **wrap-reverse**: Flex-Elemente werden umgebrochen und in mehrere Zeilen angeordnet, wenn sie nicht genügend Platz in der aktuellen Zeile haben, aber in umgekehrter Reihenfolge

CSS Flexbox: `justify-content`

- Damit wird die Ausrichtung entlang der Hauptachse (`flex-direction`) festgelegt
- Sie hilft bei der Verteilung des zusätzlichen freien Platzes, der möglicherweise übrig bleibt
- Standardwert: `flex-start`



CSS Flexbox: `align-items`

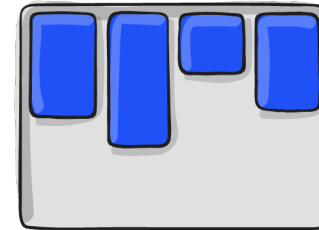
- Mit `align-items` kann das Standardverhalten für die Anordnung von Flex-Elementen entlang der Querachse in der aktuellen Zeile festgelegt werden
- Die Querachse ist die Achse, die senkrecht zur Hauptachse (`flex-direction`) verläuft

CSS Flexbox: `align-items`

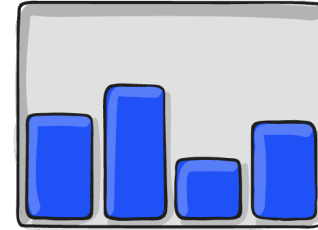
- Es gibt viele mögliche Werte für `align-items`, darunter:
 1. **`stretch`** (Standardwert): Flex-Items werden auf die volle Höhe des flexiblen Flex-Containers gestreckt
 2. **`flex-start`**: Ausrichtung am Anfang der Querachse
 3. **`flex-end`**: Ausrichtung am Ende der Querachse
 4. **`flex-center`**: Ausrichtung in der Mitte der Querachse

(`flex-direction: row;`)

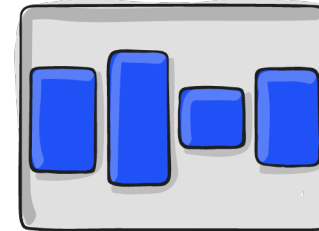
`flex-start`



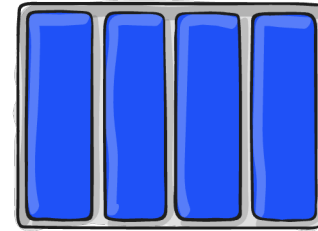
`flex-end`



`center`



`stretch`



CSS Flexbox: gap

- Die Eigenschaft `gap` steuert den Abstand zwischen Flex-Items
- Dieser Abstand wird **nur** zwischen den Elementen und nicht an den Außenkanten angewendet

CSS Flexbox: `flex-grow`

- Definiert die Fähigkeit eines Flex-Items, bei Bedarf zu wachsen
- `flex-grow` akzeptiert einen einheitenlosen Wert, der als Proportion dient
- Er gibt an, wie viel des verfügbaren Platzes innerhalb des Flex-Containers das Element einnehmen soll
- Wenn für alle Elemente `flex-grow` auf 1 gesetzt ist, wird der verbleibende Platz im Container gleichmäßig auf alle Kinder verteilt
- Wenn eines der untergeordneten Elemente den Wert 2 hat, nimmt dieses Element doppelt so viel Platz ein wie die anderen

CSS Media Queries

- Media Queries sind CSS-Regeln, die es ermöglichen, das Layout und Design einer Website für verschiedene Bildschirmgrößen anzupassen
- In diesem Beispiel wird eine Media Query erstellt, die angibt, dass der Text auf Geräten mit einer Bildschirmbreite $\leq 600\text{px}$ die Schriftgröße 14px haben soll:

```
@media only screen and (max-width: 600px) {  
  p {  
    font-size: 14px;  
  }  
}
```


Weiterführende Literatur / Quellen

- Ausführliches CSS Tutorial mit anschaulichen Beispielen: <https://www.w3schools.com/css>
- Flexbox in CSS: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- Demo zu **flex-wrap**: <https://css-tricks.com/almanac/properties/f/flex-wrap/>
- Animationen mit CSS: <https://blog.hubspot.com/website/css-animation-examples>
- Weitere CSS-Selektoren: https://www.w3schools.com/cssref/css_selectors.php
- CSS transform: <https://developer.mozilla.org/en-US/docs/Web/CSS/transform>