

**CODING  
SCHULE**

# CSS & JAVASCRIPT

Grundlagen und Vertiefung

CSS



CSS VERTIEFUNG

# CSS WIEDERHOLUNG

Grundlegende Strukturen und Eigenschaften

- HTML-Einbindung
- Selektoren
- Eigenschaften
- Werte / Optionen

# ATTRIBUTE

*HTML Attribute enthalten zusätzliche Informationen zu einem Tag*

*Diese stehen in den Tags drin nach  
name="value"*

*Grob in bezeichnende und funktionelle trennbar*

# ATTRIBUTE

```
<a href="https://www.w3schools.com">Visit W3Schools</a>
```

# ATTRIBUTE

Liste einzelner Attribute

Liste globaler, gemeinsam genutzer  
Attribute

Liste Eventhandler-Attribute

# AUFGABE 0 - EINE EINFACHE SEITE

Bearbeite die Aufgabe zur Wiederholung von HTML.

# AUFGABE 1.1



- Erzeuge eine H1 Überschrift und passe die Schriftfarbe an.
- Der erste Buchstabe der Überschrift soll jeweils eine andere Farbe haben.
- Nutze zum Styling CSS Klassen in einer CSS-Datei.

**Hello World**



# CSS CUSTOM PROPERTIES

Vordefinierte Werte (Variablen)  
Wiederverwendbarkeit im Dokument

```
:root {  
  --main-bg-color: brown;  
}  
  
element {  
  background-color: var(--main-bg-color)  
}
```

*Beginn mit "--" und Case-Sensitive*

# CSS GRADIENT

Mit CSS Gradient lassen sich Hintergrund-Farbverläufe erzeugen.

```
#grad {  
  background-image: linear-gradient(red, yellow);  
}
```

Es wird zwischen linearen und radialen Farbverläufen unterschieden.

```
background-image: radial-gradient(red, yellow, green);
```

# CSS GRADIENT

Auch Text lässt sich mit einem Farbverlauf versehen.

Codingschule

```
.textcolor {  
  background-image: linear-gradient(90deg, blue, yellow, red, purple);  
  -webkit-background-clip: text;  
  background-clip: text;  
  -webkit-text-fill-color: transparent;  
}
```

# CSS GRADIENT GENERATOREN

Zur Erleichterung bei der Nutzung von Farbverläufen gibt es zahlreiche CSS Generatoren im Internet.

<https://cssgradient.io/>

<https://mycolor.space/gradient>

<https://www.css-gradient.com/>

# AUFGABE 1.2



Erweitere die Webseite um eine H2 Überschrift. Die Überschrift soll einen Farbverlauf haben.

Die Farben des Farbverlaufs sollen in Custom Properties gespeichert werden.



# PSEUDOKLASSEN

Erweiterung eines Selektors um einen bestimmten Zustand.

*Einfacher Doppelpunkt*

```
h2:hover {  
  text-decoration: underline;  
}
```

# PSEUDOKLASSEN

```
li:first-child {  
  color: blue;  
}  
input:focus {  
  background-color: red;  
}
```

Pseudoklassen

# CSS SELEKTOREN

## PROBLEMSTELLUNG

Die erste Zeile eines Absatzes soll in einer anderen Farbe dargestellt werden.

*Lorem ipsum dolor sit amet, consetetur  
s adipscing elit, sed diam nonumy eirmod  
tempor invidunt ut labore et dolore magna  
aliquyam erat, sed diam voluptua. At vero  
eos et accusam et*



# PSEUDOELEMENTE

Erweiterung eines Selektors um einen Teil des Elements zu gestalten.

*Zwei Doppelpunkte*

```
p::first-line {  
  color: blue;  
}  
p::selection {  
  background-color: red;  
}
```

Pseudoelemente

# AUFGABE 1.3



Füge zwei Absätze (<p>) ein.

Der erste Buchstabe eines Absatzes soll jeweils in einer größeren Schriftgröße dargestellt werden.

Nutze hierfür ein Pseudoelement.

Textgenerator: <https://www.loremipsum.de/>



# AUFGABE 1.4



Über das Pseudoelement `::after` kann CSS-Code nach dem Inhalt eines HTML-Elements eingefügt werden.

Füge den folgenden CSS Code an das Ende des letzten Paragraphs. Nutze hierfür eine Pseudoklasse und `::after`.

```
content: '';  
display: block;  
height: 10px;  
background-color: red;
```



# VIEWPORT

Größe des sichtbaren Bereichs einer Webseite in Prozent.  
Nutzung der Höhe und Breite:

```
#main {  
  width: 90vw; /* View Width (%) */  
  height: 80vh; /* View Height (%) */  
}
```

# CSS POSITION

Über die Eigenschaft **Position** lässt sich die Art der Poositionierung von Elemente definieren.

```
position: static; /* Standard - keine explizite Positionierung */  
position: relative;  
position: fixed;  
position: absoulte;  
position: sticky;
```

# CSS POSITION: RELATIVE

Relative Positionierung zur "normalen" Position.

```
.card {  
  position: relative;  
  top: 30px;  
  left: 30px;  
  /* right, bottom */  
}
```

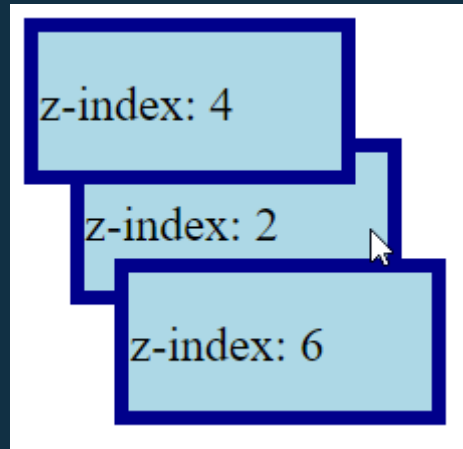
# CSS POSITIONEN

- Fixed
  - Relative Positionierung zum "Viewport"
- Absolute
  - Freie Positionierung, abhängig vom "non static" Parent-Element
- Sticky
  - Relative Positionierung bis zum Erreichen einer Scroll-Position, dannach **fixed**.

# CSS Z-INDEX

Z-Reihenfolge der Elemente.

Elemente mit höherem Z-Index sind im Vordergrund.



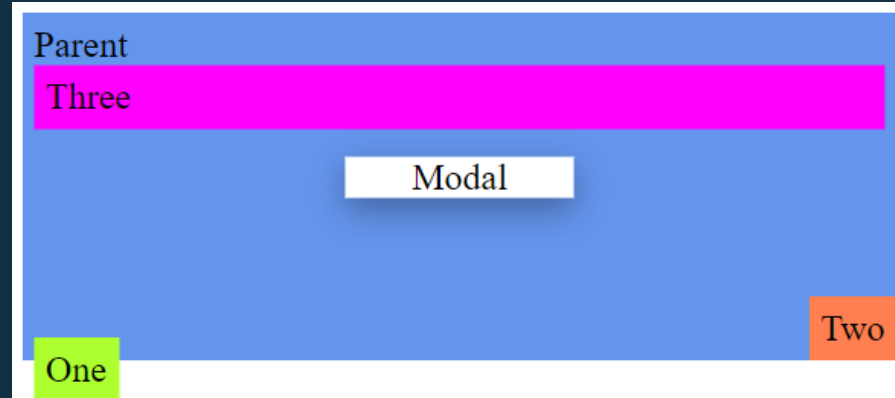


# AUFGABE 2



## POSITION UND Z-INDEX

Passe die CSS Eigenschaften an um folgende Anordnung darzustellen.



Aufgabenstellung

# AUFGABE 2.1

## EINE EINFACHE NAVIGATIONSLEISTE

Aufgabenstellung

# CSS GAME DINNER

<https://flukeout.github.io/>



# AUFGABE 3



## KEYFRAMES

Passe den CSS Code an, um die Keyframe Animation zu verändern.

Aufgabenstellung

# AUFGABE 4



## ROBOTFRIEND

Passe den CSS Code an, um den Roboter darzustellen.

Aufgabenstellung

# FLEXBOX

## Flexible Box Model

Anordnung von Elementen (Items) in einem Container  
*zeilen- oder spaltenweise*

```
.container {  
  display: flex;  
}
```

# AUFGABE 5



Was passiert bei Anwendung des Flexbox Layouts mit den Elementen?

```
<div class="container">  
  <div class="child one">Element</div>  
  <div class="child two">Element</div>  
  <div class="child three">Element</div>  
  <div class="child four">Element</div>  
</div>
```

Element

Element

Element

Element

# FLEXBOX EIGENSCHAFTEN

## FLEX-DIRECTION

Zeilenweise oder spaltenweise Darstellung der Elemente.  
*vorwärts oder rückwärts*

```
.container {  
  flex-direction: row | row-reverse | column | column-reverse;  
}
```



# FLEXBOX EIGENSCHAFTEN

## FLEX-WRAP

Über den Bildschirm hinausgehende Items werden in die nächste Zeile/Spalte geschoben.

```
.container {  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

# FLEXBOX EIGENSCHAFTEN

## JUSTIFY-CONTENT

Horizontale Anordnung von Elementen

```
.container {  
  justify-content: flex-start | flex-end | center |  
    space-between | space-around | space-evenly |  
    start | end | left | right ... + safe | unsafe;  
}
```

# FLEXBOX EIGENSCHAFTEN

## ALIGN-ITEMS

Vertikale Anordnung von Elementen

```
.container {  
  align-items: stretch | flex-start | flex-end | center |  
    baseline | first baseline | last baseline | start |  
    end | self-start | self-end + ... safe | unsafe;  
}
```

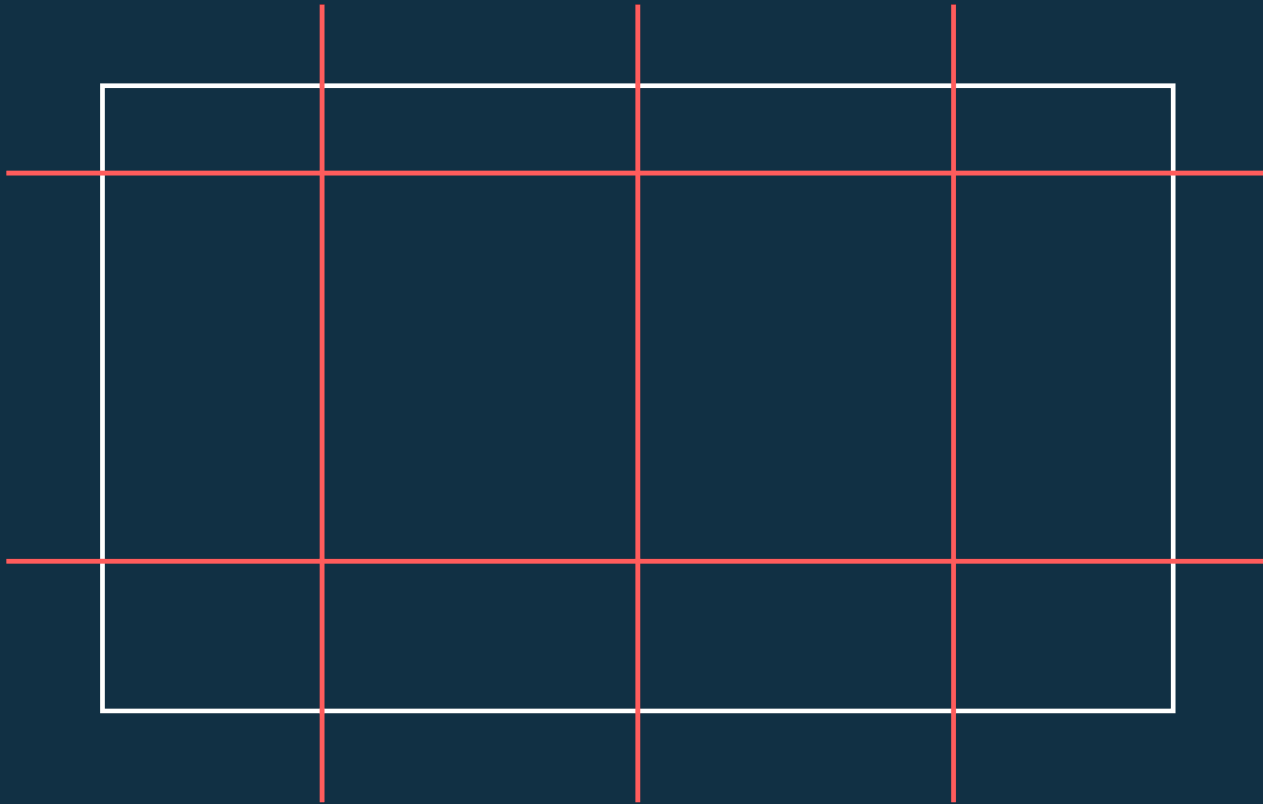
# FLEXBOX EIGENSCHAFTEN

## Weitere Flexbox Eigenschaften

```
.container {  
  align-content: flex-start | flex-end | center |  
    space-between | space-around | space-evenly |  
    stretch | start | end | baseline | first baseline |  
    last baseline + ... safe | unsafe;  
  
  gap: 10px 20px; /* row column */  
}
```

# GRID LAYOUT

Über die CSS Eigenschaft **display: grid;** lassen sich Elemente in einem Raster darstellen.



# GRID LAYOUT

```
.container {  
  display: grid;  
  grid-template-rows: 200px 1fr 100px;  
  grid-template-columns: 25% 25% 25% 25%;  
}
```

```
<div class="container">  
  <div>Element 1</div>  
  <div>Element 2</div>  
  ...  
  <div>Element 11</div>  
  <div>Element 12</div>  
</div>
```

*Wie sind die Elemente im Raster angeordnet?*

# GRID LAYOUT

A diagram illustrating a grid layout. It features a 3x4 grid of cells. The grid is defined by a white border and three vertical red lines. Two horizontal red lines are positioned between the rows of cells. The cells are numbered as follows:

1	2	3	4
5	7	8	9
9	10	11	12

# RASTER ANORDNUNG

Auch spaltenweise Anordnung möglich

```
.container {  
  ...  
  grid-auto-flow: column; /* Standard: row */  
  ...  
}
```

Mit der zusätzlichen Option **dense** werden Lücken aufgefüllt.



# FRACTION

```
.container {  
  ...  
  grid-template-rows: 1fr 1fr 2fr;  
  ...  
}
```

Einheit für den Teil des verfügbaren Platzes in einem Grid Layout.

# ELEMENTE IM RASTER POSITONIEREN

```
<div class="container">
  <div class="one">Element 1</div>
  <div class="two">Element 2</div>
  <div class="three">Element 3</div>
  <div class="four">Element 4</div>
</div>
```

```
.one {
  grid-column-start: 1;
  grid-column-end: 3; /* bis zur dritten Rasterlinie */
  /* grid-column: 1 / 3; grid-row: 1 / 3; */
  /* Gezählt werden die Rasterlinien */
}
```

# ABSTÄNDE ZWISCHEN ZELLEN

```
.container {  
  grid-gap: 10px; /* zwei Werte (Row/Column) möglich */  
  /* grid-row-gap: 10px; */  
  /* grid-column-gap: 10px; */  
}
```

# AUTO-ROWS / AUTO-COLUMNS

Standard Höhe bzw. Breite

```
.container {  
  grid-auto-rows: 100px;  
  grid-auto-column: 100px;  
  /* grid-auto-rows: min-content; */  
  /* grid-auto-rows: minmax(30px, auto) */  
}
```

**minmax()** ist eine CSS Funktion die einen Größenbereich definiert

# AUSRICHTUNGEN

Standardmäßig nehmen die Zellen die komplette verfügbare Fläche ein.

Über die Eigenschaften **justify-items** und **align-items** lassen sich die Zellen horizontal / vertikal ausrichten.

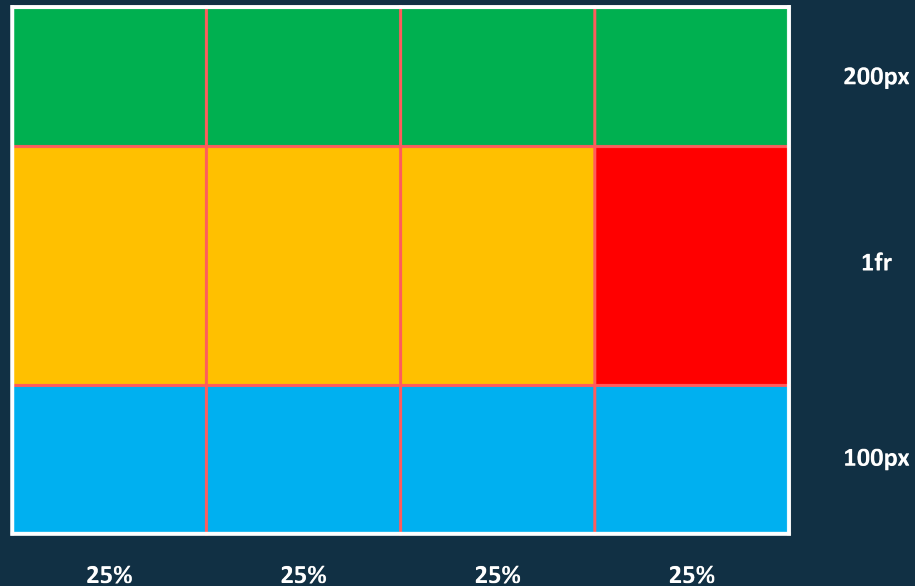
```
.container {  
  justify-items: center; /* start, end */  
  align-items: end; /* start, center */  
}
```

Einzelne Zellen können auch individuell ausgerichtet werden: **justify-self** und **align-self**

# AUFGABE 6



Überführe das folgende Grid Layout in CSS Code



Vorlage

# GRID LINKS:

- [Grid Garden](#)
- [Layoutit!](#)
- [GRIDDY](#)
- [CSS GRID Generator](#)
- [CSSGR.ID](#)

# AUFGABE 7.1



Passe die vorliegende CSS Datei an, um die Elemente  
anzuordnen.

Zieldarstellung:



Aufgabenstellung



# FLEXBOX ITEM EIGENSCHAFTEN

Neben Einstellungen, die die gesamte Flexbox betreffen, können auch individuelle Items angepasst werden.

- order
- flex-grow / flex-shrink
- flex-basis
- align-self

## AUFGABE 7.2



Erweitere die Aufgabe um die Eigenschaften *flex-grow* und *order* für die Elemente mit der Hintergrundfarbe *aquamarine*.

```
flex-grow: 2;  
order: 2;
```

*Was passiert mit der Darstellung?*

# RESPONSIVE WEBDESIGN

Durch die flexible Anordnung lassen sich **Grid** und **Flexbox** gut für responsive Webseiten verwenden.

## RESPONSIVE WEBDESIGN

*Gestalterisches und technisches Paradigma zur Erstellung von Webseiten, um diese auf das jeweilige Endgerät anzupassen.*

***Grundsatz: Mobile First***

# MEDIA QUERIES

Über CSS Media Queries lassen sich Style-Sheets je nach Ausgabegerät definieren.

```
@media only screen and (min-width: 768px) {  
  .container {  
    flex-direction: row;  
    /* Bei größeren Bildschirmen ändert sich die Darstellung  
       von Spalten zu Zeilen */  
  }  
}
```

Zusätzlich lassen sich Eigenschaften für gedruckte Seiten (print) sowie Sprachausgaben (speech) definieren.

## AUFGABE 7.3



Erweitere die letzte Aufgabe um CSS Media Queries.  
Verändere bei einer Bildschirmbreite von 768px die  
Schriftfarbe und Schriftgröße der Elemente.



# FLEXBOX LINKS

- [CSS-Tricks](#)
- [Flexbox Froggy](#).
- [Flexbox Zombies](#)

# CSS FRAMEWORKS

Sammlungen von Gestaltungselementen und Hilfsmitteln zum einfachen und standardisierten Webdesign.

## POPULÄRE FRAMEWORKS

- [Bootstrap](#)
- [Tailwind.CSS](#)

# TAILWIND CSS

Sammlung von CSS Klassen mit vordefinierten Eigenschaften.

```
/* CSS Code */  
margin: 1.5rem;
```

```
<!-- HTML mit Tailwind Klasse -->  
<div class="m-6">Tailwind</div>
```

<https://tailwindcss.com/docs>



# TAILWIND CSS

## Einbindung von Tailwind

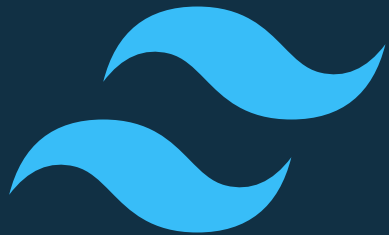
```
<head>  
  <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css"  
</head>
```

[Dokumentation](#)

## AUFGABE 7.4



Passe das Styling des `.parent-Elements` von der nativen CSS Konfiguration auf Tailwind CSS Klassen an.



# tailwindcss

# BOOTSTRAP



Dokumentation

# LIVECODING



# AUFGABE 8

Gestalte dein eigenes Frontend für eine Todo-App. Diese sollte auf jeden Fall beinhalten:

- eine Überschrift
- eine sehbare Liste der Todos
- ein Eingabefeld
- einen Button
- Bootstrap basierende Elemente

Aufgabe

# CSS LAYOUT GENERATOREN

- [CSSLAYOUT.IO](#)
- [Loading.io](#)
- [Fancy Border Radius](#)
- [CSS Generator](#)
- [CSS Filter Generator](#)
- [Webcode.tools](#)