CSS – Grundlagen II

Übersicht

- Kombination von Selektoren
- CSS Kaskade
- Position
  - Absolute
  - Relative
  - Fixed
  - Inherit
  - Static
- Layout
  - Display
  - Float & Clear
  - Media Queries
  - Flexbox
  - Grid

CSS – Kombination von Selektoren

Kombination von Selektoren: Element spezifizieren

- Indem man Selektoren ohne Abstand aneinander schreibt, kann man ein Element genauer spezifizieren. Die kombinierten Selektoren betreffen dann das selbe Element.
- Wird ein Tag Selektor dabei kombiniert, so muss dieser an erster Stelle stehen (da kein Prefix-Zeichen davor steht).

```html
<p>Lorem</p>
<p class="qwer">Ipsum</p>
<pre class="qwer">Dolor</pre>
```

```css
p.qwer {
    color: red;
}
```

Kombination von Selektoren: ODER

- Man kann Selektoren auch miteinander kombinieren dabei gibt es verschiedene Möglichkeiten, sie miteinander zu verknüpfen.
- Trennt man zwei Selektoren mit , (Komma) so ist das ein ODER.

```html
<h1>CSS Demo</h1>
<h2>Grundlagen</h2>
```

```css
h1, h2 {
    color: red;
}
```

Kombination von Selektoren: INNERHALB

- Trennt man Selektoren mit leerzeichen, so wird das zweitere ausgewählt, welches sich in ersterem befindet.
- Es muss dabei kein direktes Children sein, sondern kann auch tiefer liegen

```html
<p>Asdf</p>
<section>
    <h2>Demo</h2>
    <p>Test</p>
</section>
<section>
    <h2>Demo 2</h2>
    <div>
        <p>Deep</p>
    </div>
</section>
```

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

Kombination von Selektoren: DIREKT INNERHALB

- Trennt man Selektoren mit >, so wird das zweitere ausgewählt, welches sich direkt in ersterem befindet.
- Es muss dabei ein direktes Children sein!

```html
<p>Asdf</p>
<section>
    <h2>Demo</h2>
    <p>Test</p>
</section>
<section>
    <h2>Demo 2</h2>
    <div>
        <p>Deep</p>
    </div>
</section>
```

```css
section > p {
    color: red;
}
```

Kombination von Selektoren: UNTERHALB

- Trennt man Selektoren mit ~ (Tilde), so wird das zweitere ausgewählt, welches sich unterhalb von ersterem befindet.
- Es muss dabei nicht direkt anschließend stehen, es können andere Elemente dazwischen stehen.

```html
<p>Asdf</p>
<h2>Demo</h2>
<p>Test</p>
<p>Test more</p>
<h2>Demo 2</h2>
<h3>Demo 3</h3>
<p>Deep</p>
```

```css
h2 ~ p {
    color: red;
}
```

Kombination von Selektoren: DIREKT UNTERHALB

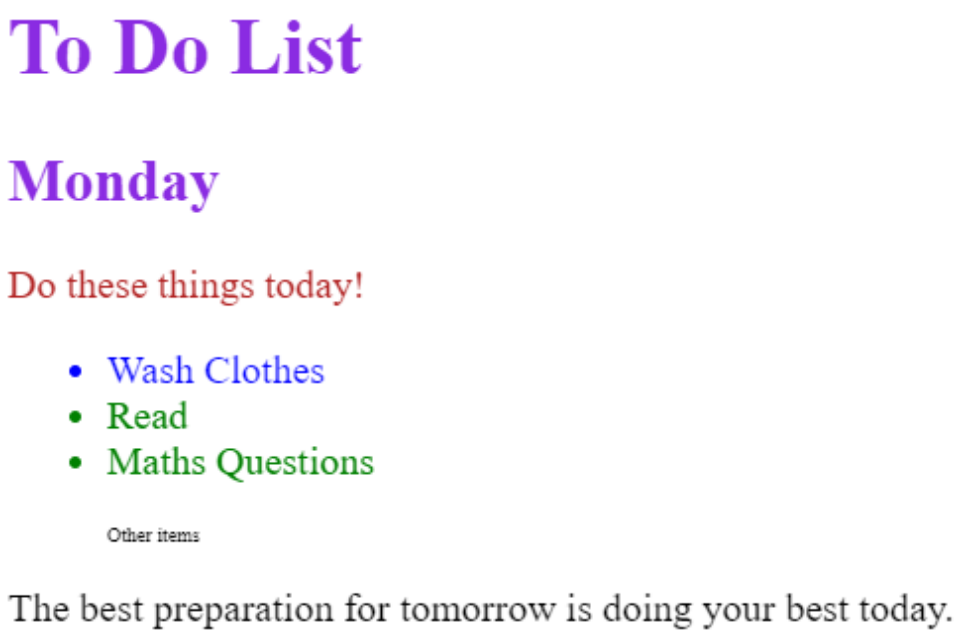
- Trennt man Selektoren mit + (Plus), so wird das zweitere ausgewählt, welches sich direkt unterhalb von ersterem befindet.
- Es muss dabei direkt anschließend stehen!

```html
<p>Asdf</p>
<h2>Demo</h2>
<p>Test</p>
<p>Test more</p>
<h2>Demo 2</h2>
<h3>Demo 3</h3>
<p>Deep</p>
```

```css
h2 + p {
    color: red;
}
```

Übung zu Kombination von Selektoren

- Übungsunterlagen: CSS/Combining Selectors
- Style das Dokument in einer Weise, die dem Bild rechts entspricht.
- Verändere dabei nicht das bestehende html sondern schreibe ein externes CSS File um das Styling zu erzielen



CSS – Kaskade

CSS Kaskade

Position

```
li {
  color: red;
  color: blue;
}
```

Specificity

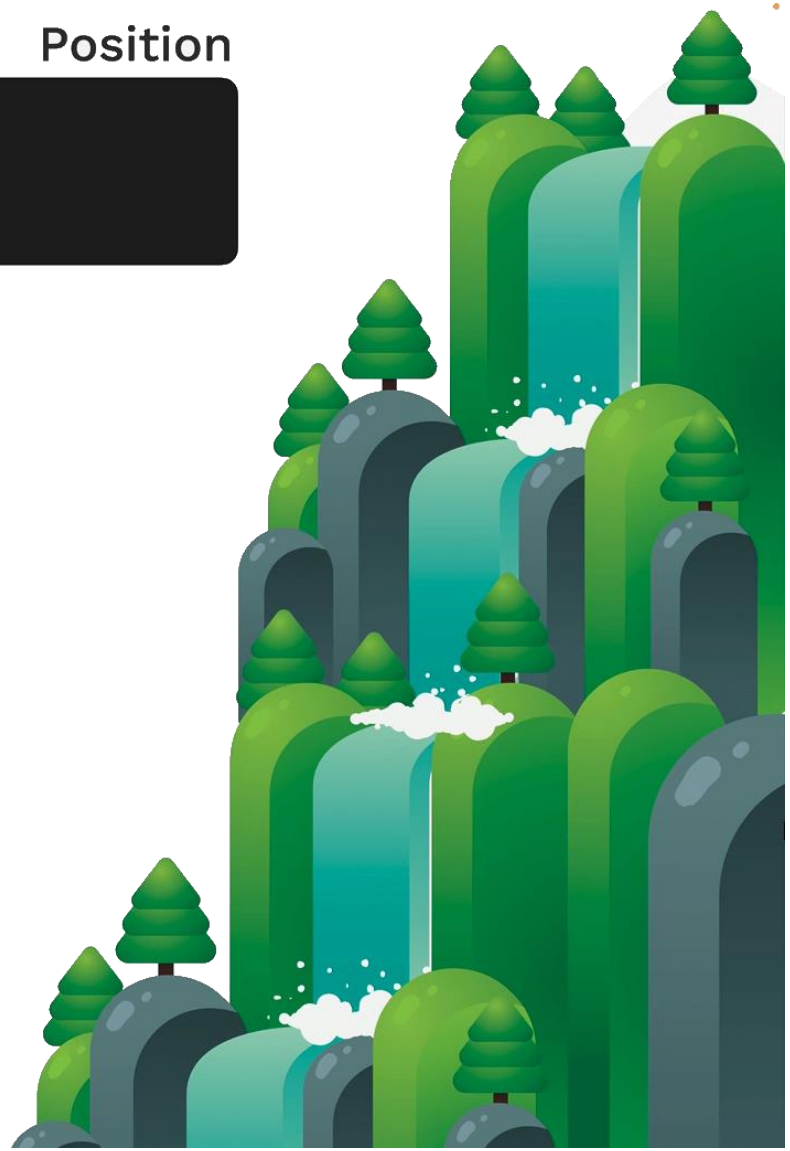
```
li {color: blue;}
.first-class {color: red;}
li[draggable] { color: purple;}
#first-id {color: orange;}
```

Type

```
<link rel="stylesheet" href="./style.css">
<style> </style>
<h1 style=" ">Hello</h1>
```

Importance

```
color: red;
color: green !important;
```



CSS Kaskade

- Die CSS-Kaskade ist das Konzept, wie CSS-Regeln auf HTML-Elemente angewendet werden, wenn mehrere Regeln konkurrieren.
- Konflikte werden durch die **Reihenfolge** der Regeln, die **Spezifität** von Selektoren und die Verwendung von **!important** gelöst
- Es gibt 4 Kategorien, die das Spezifitätsniveau eines Selektors definieren:
  - Inline-Stile - Beispiel: <h1 style="color: pink;">
  - IDs - Beispiel: #navbar
  - Klassen, Pseudoklassen, Attributselektoren - Beispiel: .test, :hover, [href]
  - Elemente und Pseudo-Elemente - Beispiel: h1, ::before

```
<html>
<head>
  <style>
    #demo {color: blue;}
    .test {color: green;}
    p {color: red;}
  </style>
</head>
<body>

<p id="demo" class="test" style="color: pink;">Hello World!</p>

</body>
</html>
```

Berechnung der Spezifität

Selector	Specificity Value	Calculation
p	1	1
p.test	11	1 + 10
p#demo	101	1 + 100
<p style="color: pink;">	1000	1000
#demo	100	100
.test	10	10
p.test1.test2	21	1 + 10 + 10
#navbar p#demo	201	100 + 1 + 100
*	0	0 (the universal selector is ignored)

https://www.w3schools.com/css/css_specificity.asp

CSS - Position

CSS Position

- Die Eigenschaft "position" legt die Art der Positionierungsmethode für ein Element fest.
- Es gibt fünf verschiedene Werte für "position":
  - "static" (statisch)
  - "relative" (relativ)
  - "fixed" (fixiert)
  - "absolute" (absolut)
  - "sticky" (klebend)
- Die Positionierung von Elementen erfolgt mithilfe der Eigenschaften
  - "top", "bottom", "left" und "right".

CSS Position: static

- HTML-Elemente sind standardmäßig statisch positioniert.
- Statisch positionierte Elemente werden nicht von den Eigenschaften "top", "bottom", "left" und "right" beeinflusst.
- Ein Element mit "position: static;" wird auf keine besondere Weise positioniert; es folgt immer dem normalen Fluss der Seite:

```
div.static {
  position: static;
  border: 3px solid #73AD21;
}
```

CSS Position: relative

- Ein Element mit "position: relative;" wird relativ zu seiner normalen Position positioniert.
- Wenn die Eigenschaften "top", "right", "bottom" und "left" eines relativ positionierten Elements festgelegt werden, wird es von seiner normalen Position verschoben.
- Es wird kein anderer Inhalt angepasst, um in eventuelle Lücken zu passen, die durch das Element entstehen

```
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
```

CSS Position: absolute

- Ein Element mit "position: absolute;" wird relativ zum nächstgelegenen positionierten Elternelement positioniert (statt relativ zum Viewport wie bei "fixed").
- Wenn ein absolut positioniertes Element jedoch keine positionierten Vorelemente hat, wird das Dokumentbody verwendet, und es bewegt sich beim Scrollen der Seite mit.
- Hinweis: Absolut positionierte Elemente werden aus dem normalen Fluss genommen und können sich überlappen.

```
div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
```

CSS Position: fixed

- Ein Element mit "position: fixed;" wird relativ zum Viewport positioniert, was bedeutet, dass es immer an derselben Stelle bleibt, auch wenn die Seite gescrollt wird. Die Eigenschaften "top", "right", "bottom" und "left" werden verwendet, um das Element zu positionieren.
- Ein fixiertes Element hinterlässt keine Lücke auf der Seite, an der es normalerweise positioniert gewesen wäre.

```
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
```

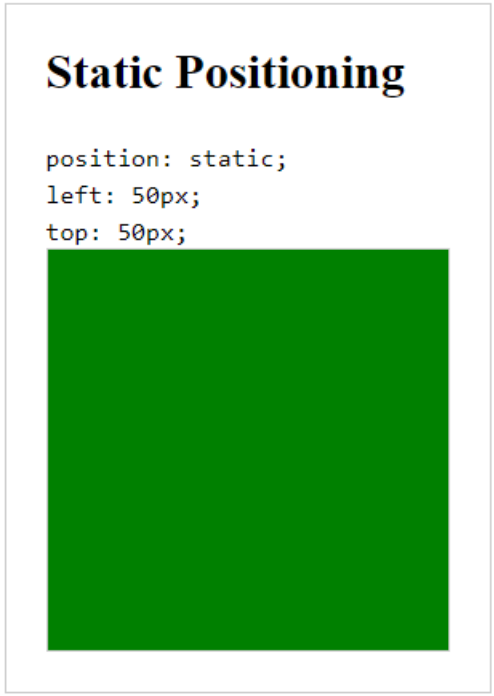
CSS Position: sticky

- Ein Element mit "position: sticky;" wird basierend auf der Scrollposition des Benutzers positioniert.
- Ein "sticky" Element wechselt zwischen relativ und fixiert, abhängig von der Scrollposition.
- Es wird relativ positioniert, bis eine bestimmte Offset-Position im Viewport erreicht ist. Dann bleibt es an dieser Stelle "kleben" (ähnlich wie "position: fixed").

```css
div.sticky {
  position: -webkit-sticky; /* Safari */
  position: sticky;
  top: 0;
  background-color: green;
  border: 2px solid #4CAF50;
}
```

CSS Position Demo

Static Positioning | Relative Positioning | Absolute Positioning | Fixed Positioning



https://appbrewery.github.io/css-positioning/

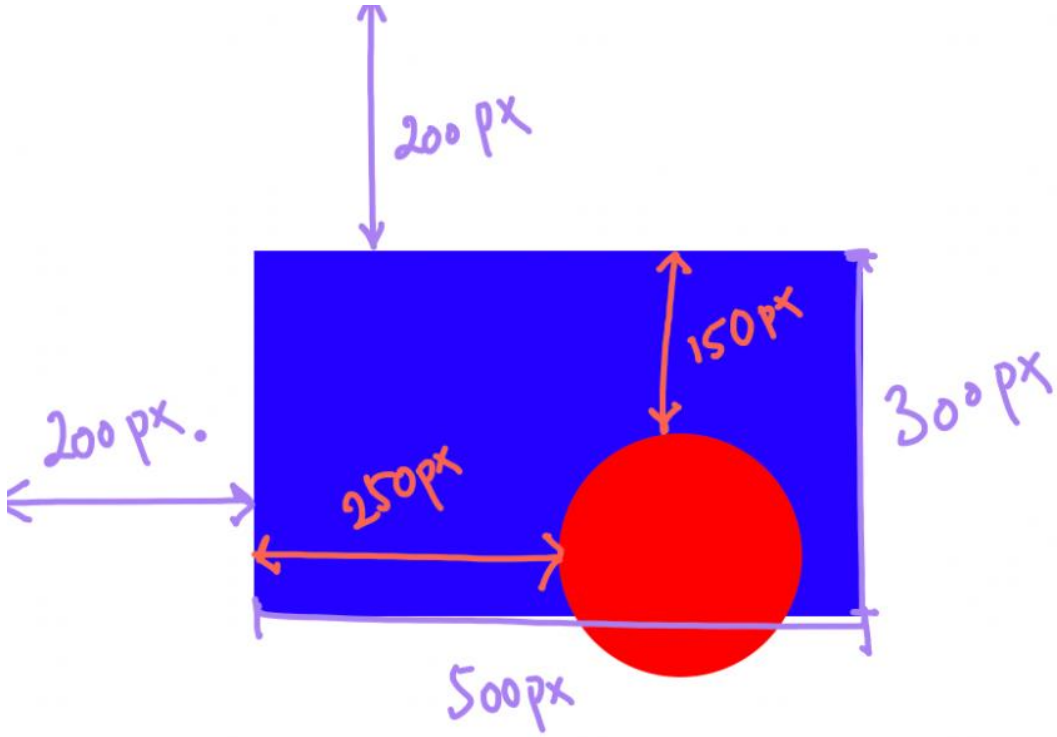
CSS Position: z-Index

- Wenn Elemente positioniert werden, können sie sich überlappen.
- Die Eigenschaft "z-index" legt die Stapelreihenfolge eines Elements fest (welches Element vor oder hinter den anderen platziert werden soll).
- Ein Element kann eine positive oder negative Stapelreihenfolge haben.


```css
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
```

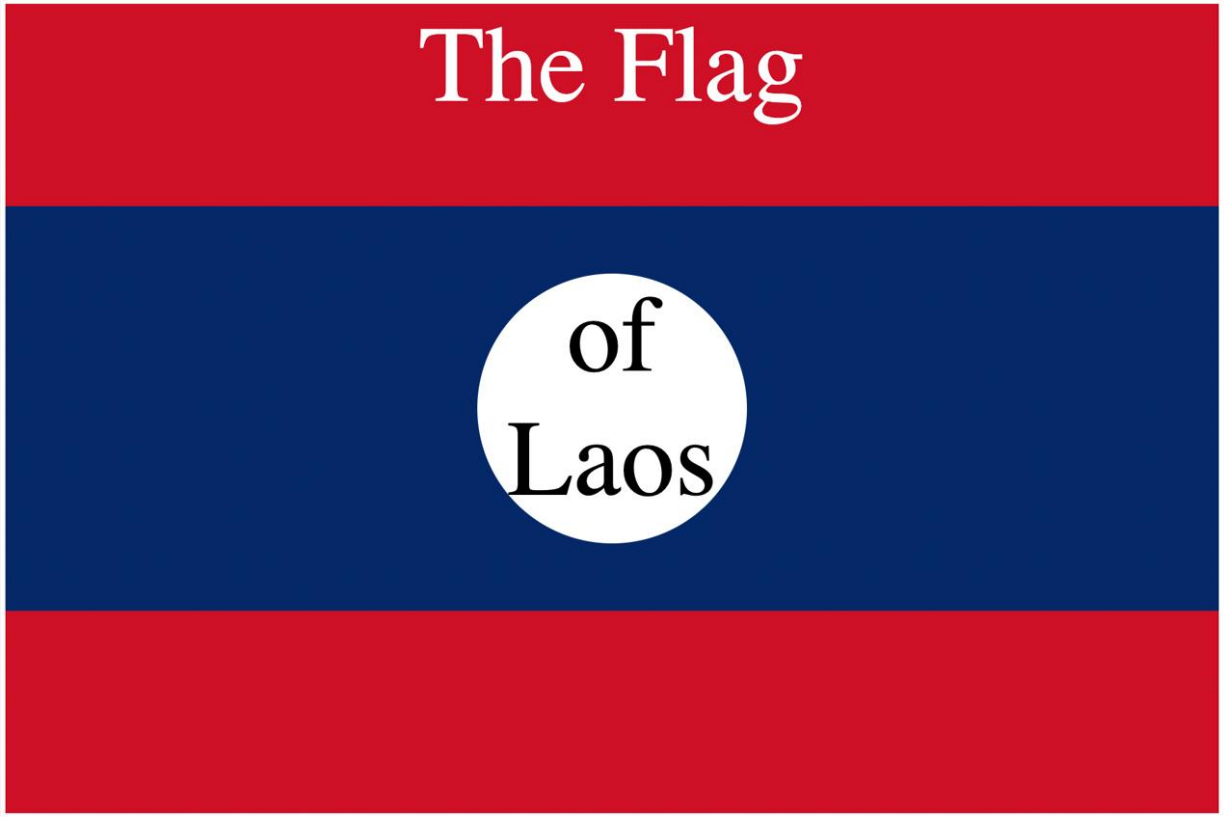
CSS Position Übung Positionierung (gemeinsam)

- Übungsunterlagen CSS/Positionierung/
- Schreibe CSS Code um das Bild unterhalb zu erzielen
- Tipp: Um einen Kreis zu erzielen nutze die Eigenschaft border-radius: 50%
- Zusatz: Lass den Kreis hinter dem blauen Element verschwinden



CSS Übung Flagge von Laos

- Im Repository findest du ein file „flag.html“ im CSS Folder deiner persönlichen Codebeispiele
- Definiere den Style des Dokuments im head sodass die Flagge von Laos erscheint
- WICHTIG! Ändere nicht das HTML. Füge keine Klassen/IDs/Elemente hinzu. Nutze dein Wissen über das Kombinieren von Selektoren, Positionierung und die Spezifität von CSS stattdessen.



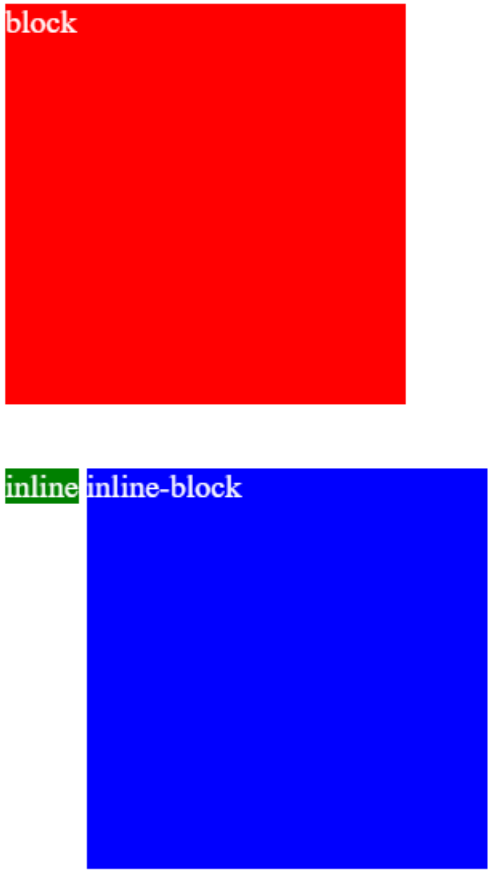
CSS - Layout

CSS Display

- Die Eigenschaft "display" legt fest, wie ein Element angezeigt wird bzw. ob es angezeigt wird.
- Jedes HTML-Element hat einen standardmäßigen Anzeigewert, der von seinem Typ abhängt. Der Standard-Anzeigewert für die meisten Elemente ist "block" oder "inline".
- Häufig verwendete Werte:
  - none (Element wird nicht angezeigt und nimmt keinen Platz ein)
  - block (kennen wir bereits)
  - inline (kennen wir bereits)
  - inline-block
  - flex, flexbox (wird in kommenden Kapiteln behandelt)
  - grid (wird in kommenden Kapiteln behandelt)

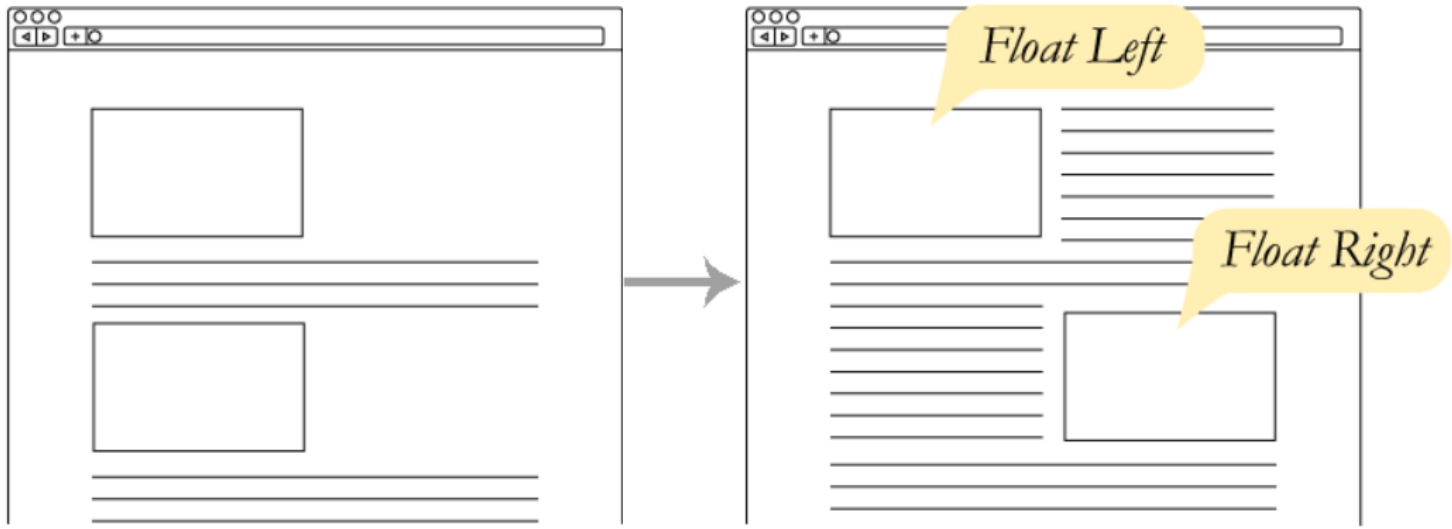
CSS Inline-block

- **block** Elemente nehmen die gesamte Breite des Parent Elements ein und erzeugen einen Zeilenumbruch. Zusätzlich können Breite und Höhe angegeben werden
- **inline** Elemente nehmen immer nur so viel Platz ein wie sie benötigen. Angabe von Breite und Höhe haben keine Auswirkungen
- "**inline-block**„ kombiniert die Eigenschaften von block und inline indem:
  - …es ermöglicht, eine Breite und Höhe für das Element festzulegen
  - …keinen Zeilenumbruch erzeugt, Elemente können nebeneinander platziert werden



CSS Float & Clear

- Die CSS-Eigenschaft "float" legt fest, wie ein Element in das restliche HTML eingebettet (umfließend) werden soll
- Die CSS-Eigenschaft "clear" gibt an, neben welchen Elementen das geklärte Element schweben kann und auf welcher Seite es dies tun sollte.

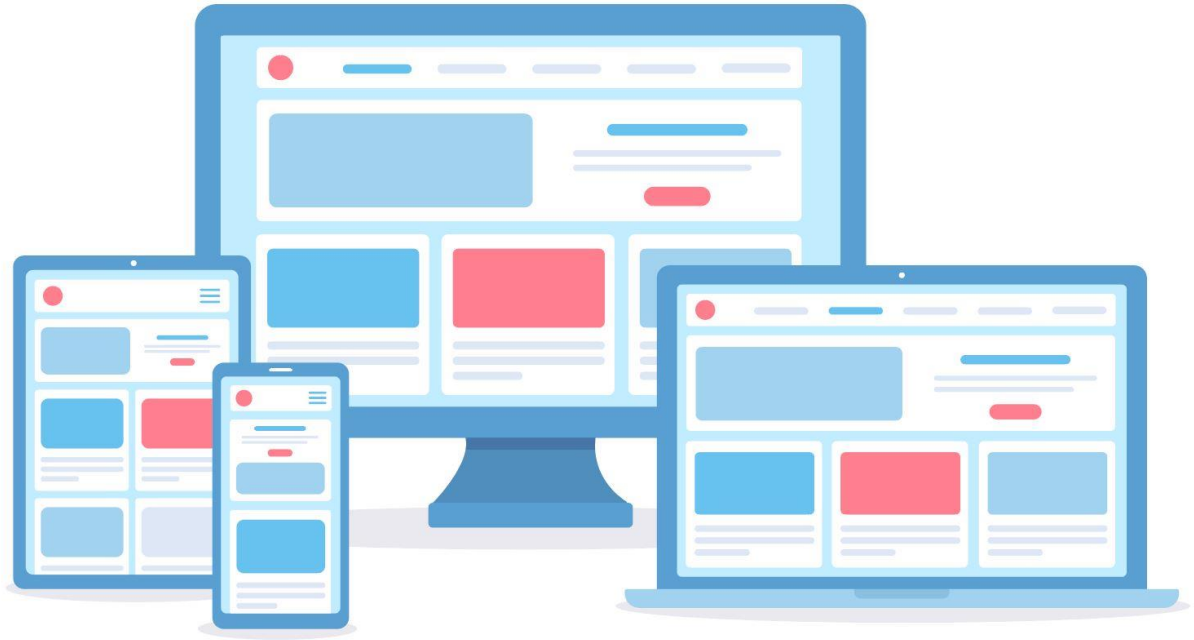


CSS Layout früher „Newspaper“

- "Block" wird verwendet für Abschnitte auf einer Webseite.
- "Inline" wird für Text verwendet.
- "Table" wird für zweidimensionale Tabellendaten verwendet.
- "Positioned" wird für die explizite Positionierung eines Elements verwendet.
- „Float" wird für Umfließende Inhalte verwendet, und für Ausrichtung von Elementen.
- Wenig Bedarf nach Responsive Webdesign

CSS Layout heute „Responsive Webdesign“

- Media Queries
- Flexbox
- Grid
- Frameworks
  - Bootstrap
  - Material UI
  - …



CSS Media Queries

- Media Queries werden verwendet, um auf Eigenschaften der Anzeige zu reagieren
  - Breite und Höhe der Anzeige
  - Ausrichtung (befindet sich das Tablet/Handy im Quer- oder Hochformat?)
  - Auflösung
- Die Verwendung von Media Queries ist eine beliebte Technik, um ein maßgeschneidertes Stylesheet für Desktop-Computer, Laptops, Tablets und Mobiltelefone (wie iPhone und Android-Telefone) bereitzustellen.

```css
@media (min-width: 400px) and (max-width: 600px){
  /* CSS für Anzeigen zwischen 400px und 600px Breite */
  div {
    height: 200px;
    width: 200px;
  }
}
```

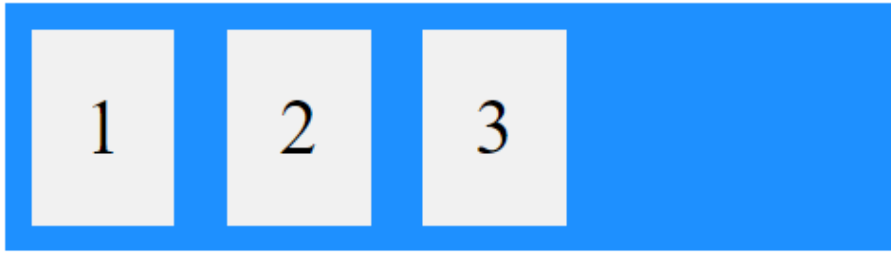
CSS Übungsprojekt Webdesign

- Übungsbeispiel CSS/Web Design Agency Project
- Schreibe CSS Code um ein Layout, ähnlich dem Layout in der Lösung zu erzielen.
- Wichtige Layout Elemente:
  - 2 Blöcke nebeneinander
  - Fließender Text um Bild
  - Responsive Design (Blöcke untereinander bei schmaler Bildschirm-Breite)
- Lösungsurl: https://holdo89.github.io/Web-Design-Agency-Project/

CSS Flexbox

- Das Modul für das flexible Box-Layout erleichtert die Gestaltung flexibler, reaktionsfähiger Layoutstrukturen.
- Anwendungsbereich eher bei 1-dimensionale Strukturen
- Ein flexibles Layout muss ein übergeordnetes Element haben, bei dem die Eigenschaft "display" auf "flex" gesetzt ist.
- Die direkten Child-Elemente des flexiblen Containers werden automatisch zu flexiblen Elementen.
- Über weitere Flex-Eigenschaften kann das Layout genau definiert werden
  - Flex-direction
  - Align-items
  - Justify-content
  - Align-content

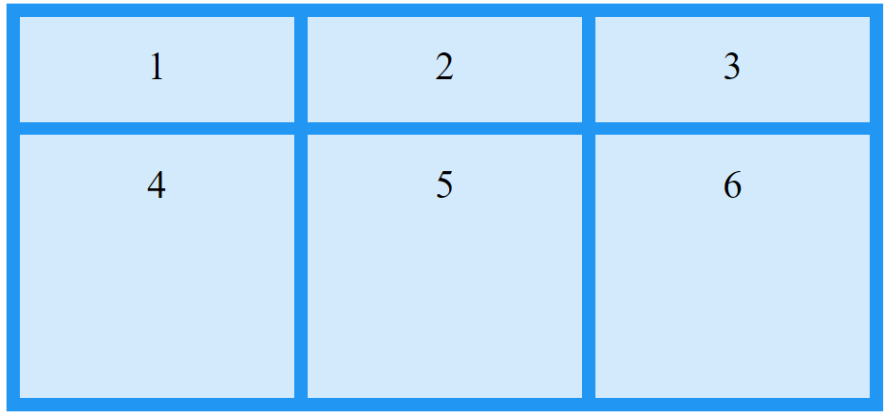
```css
.flex-container {
  display: flex;
  background-color: DodgerBlue;
}

.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
```



CSS Grid

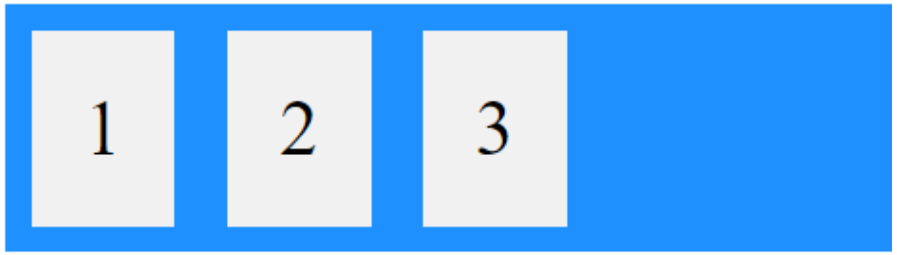
- Das CSS Grid Layout-Modul bietet ein Rastersystem für das Layout, mit Zeilen und Spalten,
- Anwendung eher bei Layout-Strukturen in 2 Dimensionen.
- Die Eigenschaft "**grid-template-columns**" definiert die Anzahl der Spalten in Ihrem Rasterlayout und kann die Breite jeder Spalte festlegen.
- Die Eigenschaft "grid-template-columns" kann auch verwendet werden, um die Breite der Spalten festzulegen.
- Die Eigenschaft "**grid-template-rows**" definiert die Höhe jeder Zeile.
- Mit der Eigenschaft „**gap**“ wird der Abstand zwischen Zeilen und Spalten definiert (erster Wert = Zeile, zweiter Wert = Spalte)

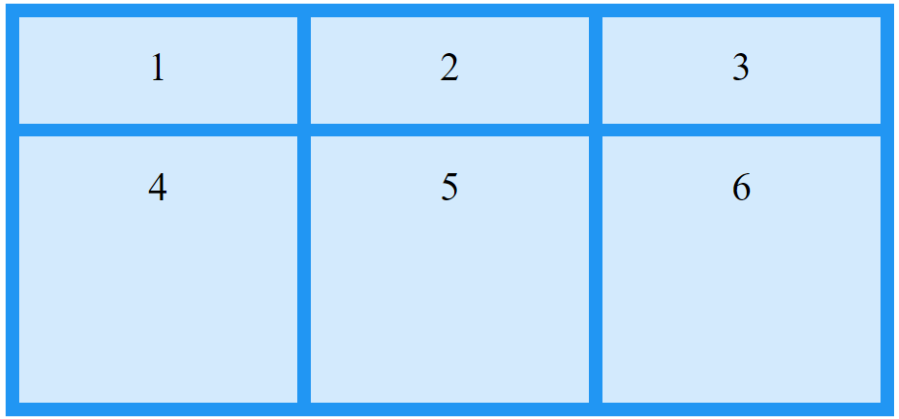
```css
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  grid-template-rows: 80px 200px;
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
```



„Hausaufgabe“ Flexbox & Grid

- Flexbox und Grid bieten viele Eigenschaften mit denen man das Layout definieren kann. Versuche dir einen groben Überblick zu verschaffen, über die Möglichkeiten die sich mit diesen beiden Modulen ergeben.
  In der kommenden Einheit werden wir die beiden Module im Detail gemeinsam besprechen.
- Flexbox:
  https://www.w3schools.com/css/css3_flexbox.asp
- Grid:
  https://www.w3schools.com/css/css_grid.asp





Viel Erfolg beim Entwickeln!