CSS - Cascading Style Sheets

CSS

Formatierungssprache für HTML-Dokumente

- HTML zur Strukturierung und semantischen Auszeichnung
- CSS zum Formatieren/Stylen des Dokuments (auch XML)
- Möglichkeit für verschiedene Ausgabemedien (Bildschirm, Papier) unterschiedliche Darstellungen vorzugeben: **Responsives Design**!

Trennung von Inhalt und Design

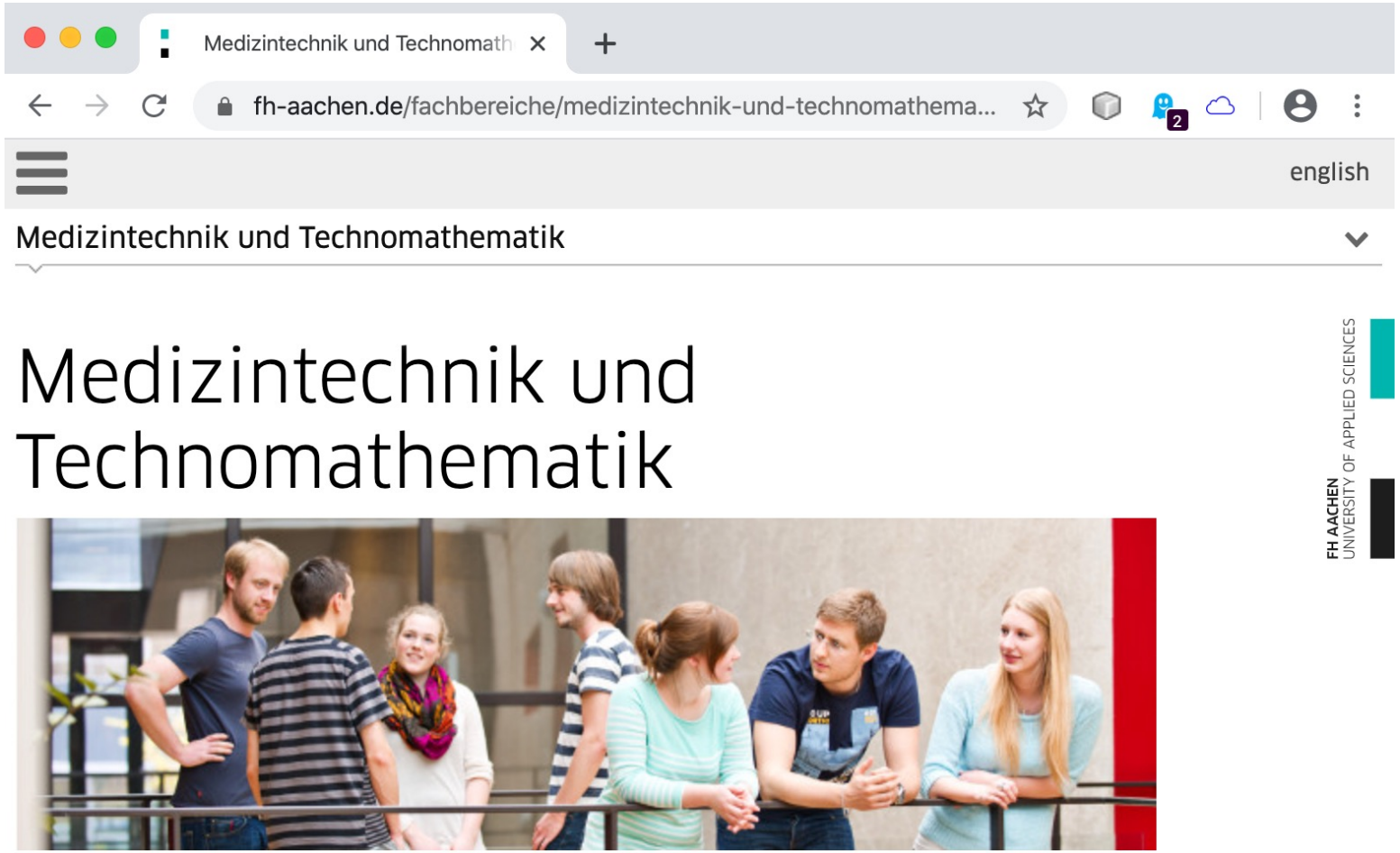
- Optimalfall: Austausch des Designs ohne Veränderung des Dokuments
- „inline"-HTML-Style vermeiden, also keine font-Tags, o.Ä. verwenden
- Elemente gruppieren und mit Bezeichner versehen: Klassen und Ids mit Formatierungen versehen
- Wiederverwendbare Stil-Elemente schaffen

Beispiele

- Farben (Vordergrund, Hintergründe, Rahmen)
- Schriftgrößen, -farben, -arten, -stile
- Textformatierungen, Rahmen, Ränder
- Positionierung (pixelgenau!)
- …

CSS – Wie kann ich responsives Design etablieren?

CSS oder: Wie kann ich responsives Design etablieren?

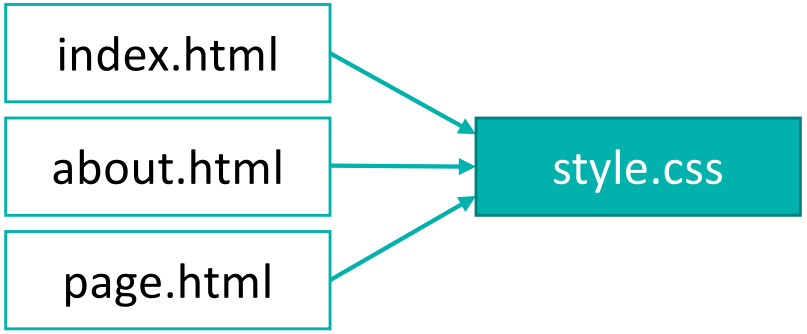


CSS
Einbinden in das Dokument

1. Einbinden im <head> (empfohlen)

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

> Wiederverwenden der Formatierungen über mehrere Dokumente hinweg
> Hier wird ein persistentes Stylesheet eingebunden (wird immer geladen und ist der Default
> Style-Dateien liegt hier auf dem gleichen Server



2. style-Element im <head>

```
<style type="text/css">
  h1 { font-size: 200%; }
</style>
```

3. style-Attribut (unschön, nicht wiederverwendbar)

```
<h1 style="font-size:200%;">Überschrift</h1>
```

Aufbau eines CSS-Dokumentes

```
Selektor {
    Eigenschaft: Wert;
    Eigenschaft: Wert;
}
Selektor1, Selektor2 {
    Eigenschaft: Wert;
    /* Kommentar */
}
```

> Selektoren sprechen eine Gruppe von HTML-Elementen an
> Spezifizierte Eigenschaften werden auf diese HTML-Elemente angewendet
> Es gibt keine einfachen einzeiligen Kommentare, aber /* Kommentar */

CSS
Selektoren

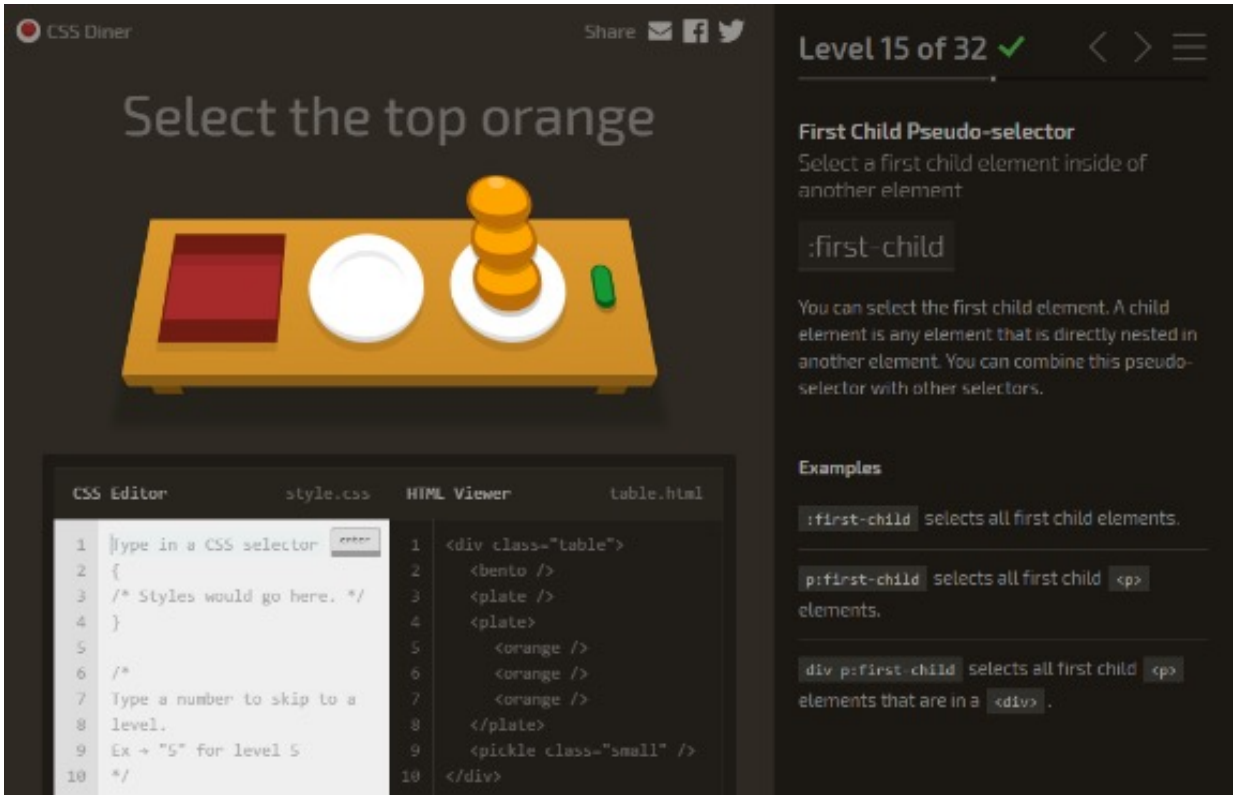
Beispiele

```
h1 { font-size:200%; }      /* <h1>...</h1> */
p strong { ... }             /* <p><strong>...</strong></p> */
p.first  { ... }             /* <p class="first">...</p> */
p, li { ... }                /* p- und li-Tags */
input[name=pw] { ... }       /* <input name="pw" .../> */
.hl { ... }                  /* Alle Tags mit class="hl" */
div#navi { ... }             /* <div id="navi">...</div> */
a:hover { ... }              /* a-Tag beim "Mouseover" */
```

- Es existieren noch viele weitere CSS-Selektoren
  - > https://wiki.selfhtml.org/wiki/CSS/Selektoren
  - > https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

Spiel zum Lernen der CSS-Selektoren

- https://flukeout.github.io/



Werte und Maßeinheiten

- Meist besteht eine Angabe aus zwei Teilen:
  - > Wert und Einheit, Beispiel: 20px = (20, Pixel)
- Pixel (px) entspricht der absoluten Basiseinheit
- Wir sollten aber sinnvollere Einheiten wählen, gerade im Zusammenhang mit responsiven Design

Relative Längenangaben

Einheit	Beschreibung	Beispiel
%	Angabe relativ zum Elternelement (bzw. Referenzelement)	`width: 50%`
em	1em entspricht der Schriftgröße des Elementes auf das die Eigenschaft angewendet wird. Oftmals wird hier in dem Selektor auch ein Font angegeben	`border-width: 1em;`
vw, vh	Viewport Units (*viewport width* und *viewport height*) sind relativ zur Größe des Browserfensters	`font-size: 3vw;` `height: 100vh;`

Der Viewport ist der sichtbare Bereich im Browser. Wenn ein Browserfenster verkleinert wird, so ändert sich der Wert

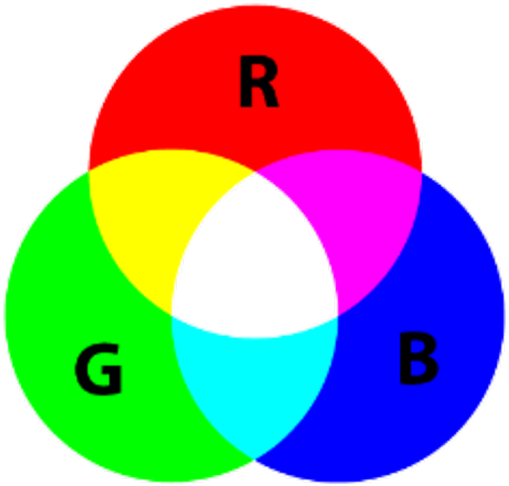
Weitere Einheiten:

- Für Druckmedien: cm, mm, in
- Weitere relative Einheiten: ex, ch, rem, vmin, vmax

CSS
Wertangaben

Farben

- RGB-Farbmodell definiert eine Farbe durch die Angabe der Anteile an (rot, grün, blau)
  - > Werte im Bereich von 0 bis 255
  - > Beispiel: (255 rot, 255 grün, 0 blau) = gelb
  - > Hexadezimale Schreibweite: `#ffff00`
  - > Kurzschreibweise: `#ff0`



- RGBA erlaubt zusätzlich eine Deckkraftangabe
  - > Beispiel: rgba(255, 255, 0, 0.5)
    - – Gelb mit 50% Deckkraft

- Ergänzend gibt es einige fest vorgegebene Farbwerte
  - > Beispiele: red, blue, cyan, yellow

Farbbeispiele (auf weißem Hintergrund)

```
.a {
    color: red;
}
.b {
    color: cyan;
}
.c {
    color: #000;
}
.d {
    color: #ffffff;
}
.e {
    color: #f00;
}
.f {
    color: #32dfa2;
}
.g {
    color: rgba(255, 0, 0, 0.5);
}
```

Text mit Klasse a

Text mit Klasse b

Text mit Klasse c

Text mit Klasse e

Text mit Klasse f

Text mit Klasse g

CSS
Eigenschaften

Text- und Schriftformatierung

```
font-family: Arial, sans-serif;   /* Schriftart */
font-style: italic;               /* kursiv */
font-size: 20px;                  /* Schriftgröße */
font-weight: bold;                /* fett */
color: blue;                      /* Textfarbe (blau) */
text-align: center;               /* zentrieren */
font: 20px Arial, sans-serif;     /* zusammenfassend */
```

- Seltener benötigt:
  - > text-decoration, letter-spacing, font-strech, text-outline, text-stroke
  - > …
  - > https://wiki.selfhtml.org/wiki/CSS

CSS
Eigenschaften

Rahmenformatierung

```
border-width: 10px;      /* Breite */
border-style: solid;     /* Durchgezogene Linie */
border-color: red;       /* Farbe (rot) */
border: 10px solid red;  /* Zusammengefasst */
border-radius: 5px;      /* Rundung der Ecken */
```

- Aufschlüsselung der Eigenschaften in (top, right, bottom, left) möglich
  - > border-top, border-right, border-bottom, border-left
- Besser relative angaben!

Hintergrundformatierung

```
background-color: red;              /* Farbe (rot) */
background-image: url(logo.png);    /* Datei */
background-position: 10px 20px;     /* Verschiebung */
background-repeat: no-repeat;       /* Bild nicht "kacheln" */
```

- Weitere Eigenschaften:
  - > -size, -attachment, -clip, -origin, …
  - > https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Hintergrundfarben_und_-bilder

Absolute Positionierung

- Element exakt an der Stelle (100, 100) positionieren:

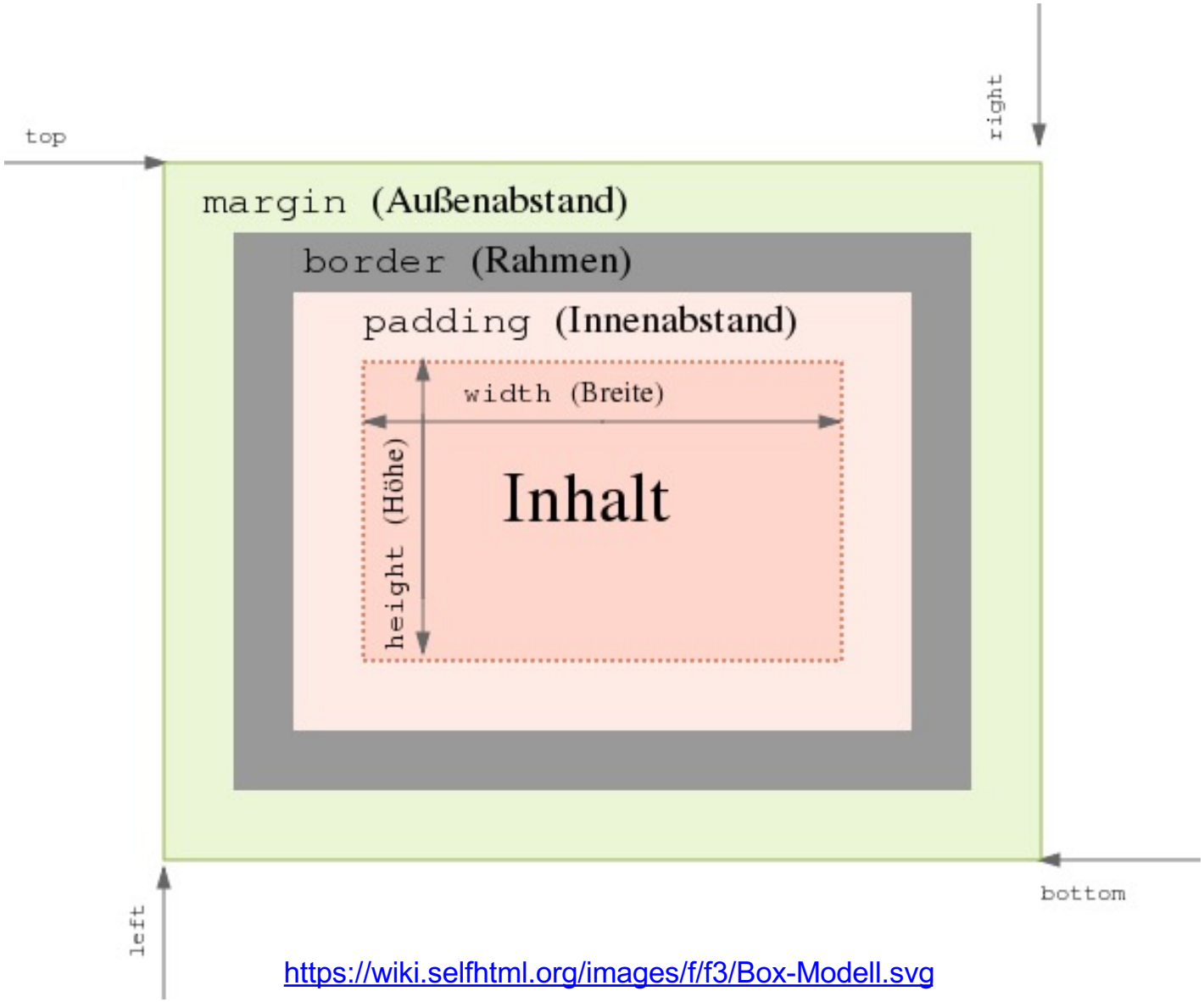
```
position: absolute;
left: 100px;
top: 100px;
```

> nur in seltenen Fällen notwendig, tatsächlich sogar kontraproduktiv

Formatierung einer Box

```
width: 100px;                    /* Breite */
height: 100px;                   /* Höhe */
margin: 10px;                    /* Außenabstand */
padding: 5px 10px 5px 10px;      /* Innenabstand */
display: none;                   /* "Display"-Typ */
```

CSS
Anzeige & Positionierung



https://wiki.selfhtml.org/images/f/f3/Box-Modell.svg

Formatierung einer Box

Auch hier gilt:

- keine festen Werte für die Breite und Höhe
- Festlegen der Abstände über padding
- Nutzen relativer Längenmaße die sich an die Schriftgröße anpassen (em)

Box-Sizing Property

- Standardkalkulation:

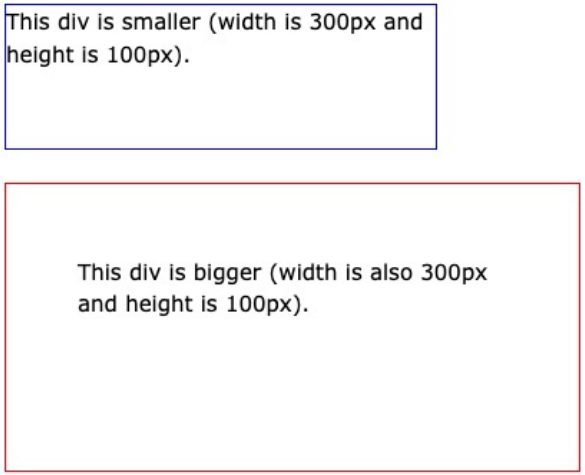
```
width + padding + border = actual width of an element
height + padding + border = actual height of an element
```

- Konsequenz: Padding und Border beeinflussen die Größe der Box!
- Mittels box-sizing kann dies verhindert werden, hier beinhaltet die Größe der Box auch Padding und Border:

```
box-sizing: border-box
```

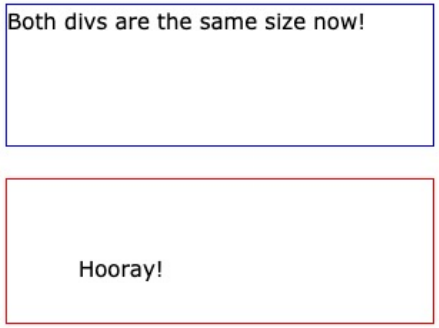
CSS
Anzeige & Positionierung

```
.div1 {
  width: 300px;
  height: 100px;
  border: 1px solid blue;
}

.div2 {
  width: 300px;
  height: 100px;
  padding: 50px;
  border: 1px solid red;
}
```




```
.div1 {
  width: 300px;
  height: 100px;
  border: 1px solid blue;
  box-sizing: border-box;
}

.div2 {
  width: 300px;
  height: 100px;
  padding: 50px;
  border: 1px solid red;
  box-sizing: border-box;
}
```

Noch besser wäre die Verwendung von em statt px!



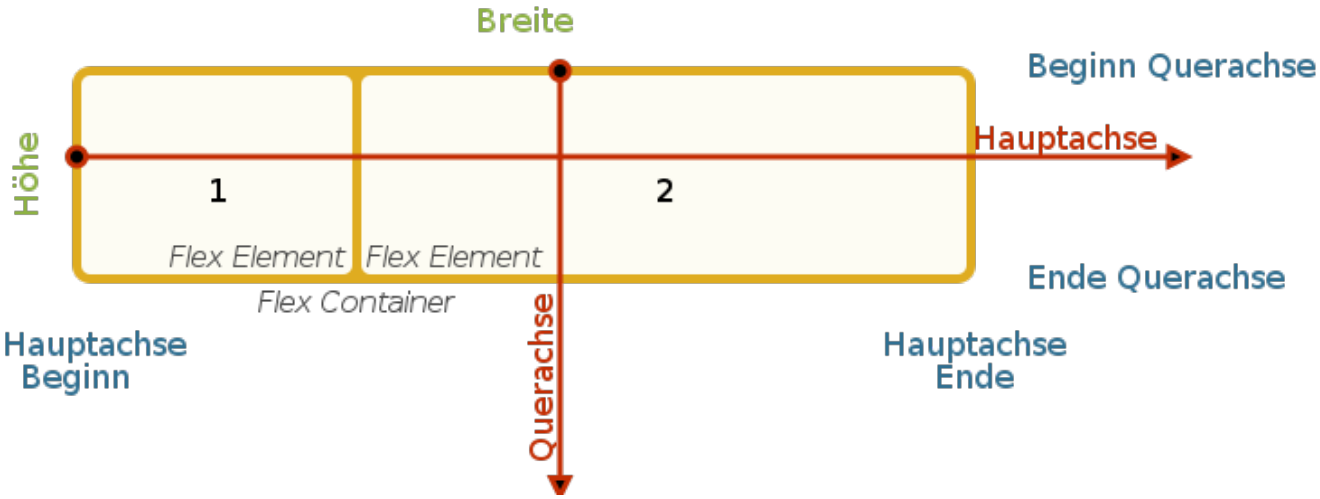
Empfehlung: Alle Elemente so setzen

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

https://www.w3schools.com/css/css3_box-sizing.asp

Erst mit dem Flexbox-Container lassen sich responsive Anordnungen multipler Elemente entwerfen

- Ein Flexbox-Container beinhaltet Flex-Items
- Diese werden innerhalb des Flex-Containers nach den Anforderungen des Layouts angeordnet
- Flex-Items werden dabei an einer Flex Line ausgerichtet, die im Standard linksbündig und horizontal ausgerichtet ist



https://wiki.selfhtml.org/extensions/Selfhtml/frickl.php/Beispiel:CSS-Flexbox-1.html

CSS
Flexbox

https://wiki.selfhtml.org/extensions/Selfhtml/frickl.php/Beispiel:CSS-Flexbox-1.html#view_result

```
<div class="flex-container">
    <p class="flex-item">1</p>
    <p class="flex-item">2</p>
    <p class="flex-item">3</p>
    <p class="flex-item">4</p>
    <p class="flex-item">5</p>
</div>

 <div class="flex-container">
    <section class="flex-item">
        <h2>Flexbox</h2>
        <p>Flexbox ist eine moderne und einfache Möglichkeit, responsive und flexible Layouts
            zu erstellen, ohne feste Größenangaben und weitere CSS-Einstellungen wie <code>position</code>,
            <code>float</code> oder <code>clear</code> nutzen zu müssen. </p>
    </section>
    <section class="flex-item">
        <h2>Was kann Flexbox?</h2>
        <p>Flexbox verzichtet auf feste Breiten, wobei sich das flexible Element an den tatsächlich
            verfügbaren Platz anpasst, indem es die Leerräume zwischen den Elementen gleichmäßig
            verteilt. </p>
    </section>
    <section class="flex-item">
        <h2>flexibler Container</h2>
        <p>Flexbox benötigt nur einen CSS-Regelsatz:</p> <pre><code>.flex-container {
  display: flex;
}</code></pre> </section>
    </div>
```

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

.flex-item {
    border: 1px solid;
    border-radius: 0 .5em .5em;
    margin: .5em;
    padding: .5em;
    background: #ffebe6;
}

p.flex-item {
    font-weight: bold;
    text-align: center;
}

.flex-item:nth-of-type(2) {
    background: #fdfcf3;
}

.flex-item:nth-of-type(3) {
    background: #ebf5d7;
}

.flex-item:nth-of-type(4) {
    background: #e6f2f7;
}

.flex-item:nth-of-type(5) {
    background: hsla(277, 53%, 73%, 0.3);
}
```

1 2 3 4 5

Flexbox

Flexbox ist eine moderne und einfache Möglichkeit, responsive und flexible Layouts zu erstellen, ohne feste Größenangaben und weitere CSS-Einstellungen wie `position`, `float` oder `clear` nutzen zu müssen.

Was kann Flexbox?

Flexbox verzichtet auf feste Breiten, wobei sich das flexible Element an den tatsächlich verfügbaren Platz anpasst, indem es die Leerräume zwischen den Elementen gleichmäßig verteilt.

flexibler Container

Flexbox benötigt nur einen CSS-Regelsatz:

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

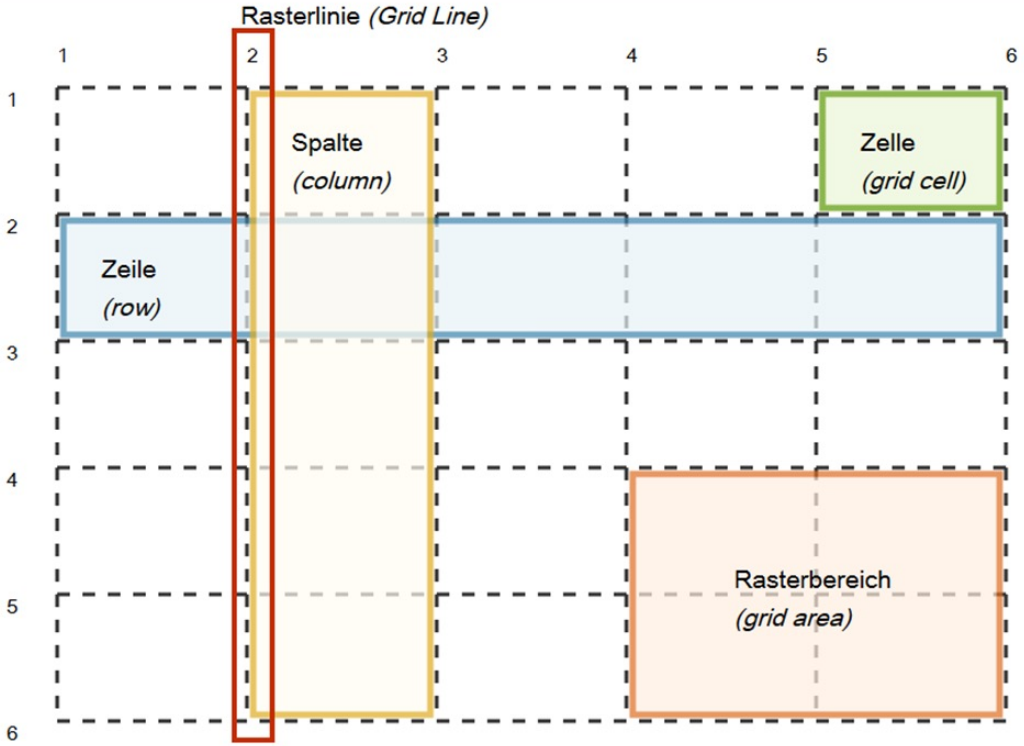
https://wiki.selfhtml.org/extensions/Selfhtml/frickl.php/Beispiel:CSS-Flexbox-1.html#view_result

https://www.w3schools.com/css/css3_flexbox.asp

CSS
Grids

Mittels Flexbox können eindimensionale Strukturen verwaltet werden, CSS Grids ermöglichen zweidimensionale Layouts

- Anlegen eines Rasters
- Kindelemente werden hier ohne feste Größenangaben positioniert



https://wiki.selfhtml.org/wiki/CSS/Tutorials/Grid/Einführung

Responsives Design wird zumeist mittels existierender Frameworks erreicht

- Bootstrap ist der populärste CSS-Framework (nicht nur CSS)
- Benötigt ab Version 4 aktuellere Browser (kein IE9)

Bootstrap verfolgt über den viewport einen Mobile-First-Ansatz

- Die Layout-Stile wurden zunächst für kleine Bildschirme entworfen, um sie dann den größeren anzupassen
- `<meta name="viewport" content="width=device-width, initial-scale=1">`
- Viewport-Meta-Tag als Basis zur Behandlung des sichtbaren Bereichs

Bootstrap wird im Head-Element zumeist von einem externen Anbieter (Content Delivery Network) eingebunden und so mit geladen

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
```

CSS

CSS bietet noch weitere Möglichkeiten zur Formatierung

- Media Queries als Werkzeug zur Realisierung eines respsonsiven Designs: media="screen"
- Transitions (animierter Übergang von einem Status in einen anderen)
- Animations (Animation mittels CSS)
- Flexbox
- Grafikfilter
- @-Regeln (**@charset** "utf-8", **@viewport** für responsive Design)

- Zusätzlich wird bei Formatierungsschwierigkeiten empfohlen das Box-Modell zu studieren:
  > https://wiki.selfhtml.org/wiki/CSS/Box-Modell

Ergänzend gibt es auch Sprachen, die auf CSS aufsetzen und zusätzliche Features implementieren

- SASS
- LESS