

Cascading Style Sheets

Wie aus Struktur eine gestaltete Seite wird

Informatik – 3. Semester

Warum ist dieser Button rot?

Leitfrage: Warum ist der Button rot? ▾

CSS prüfen

Zurücksetzen

HTML

```
<nav class="leiste">
  <button id="senden"
    class="knopf gefahr">
    Senden
  </button>
</nav>
```

CSS

5 DEKLARATIONEN

```
.leiste .gefahr {
  background: red;
}

button {
  background: blue;
  color: white;
  font-size: 2rem;
}

.knopf {
  background: green;
}
```

BROWSER

Senden

1 Regeln und Selektoren

3 Boxen

5 Selbst gestalten & Ausblick

2 Die Kaskade

4 Layout

Wie sage ich, welches Element gemeint ist?

Startseite ohne CSS

CSS prüfen

Zurücksetzen

CSS0 DEKLARATIONEN

```
/* Ihre Regeln kommen hierher.  
Zum Beispiel:  
body { font-family: system-ui, sans-serif; } */
```

BROWSER

Mitfahrbörse TH Lübeck

- [Fahrt suchen](#)
- [Fahrt anbieten](#)
- [Mein Konto](#)

Heute zum Campus

Bahnhof → Campus

Abfahrt **7:45 Uhr**, noch 2 Plätze frei.

Travemünde → Campus

Abfahrt **7:20 Uhr**, noch 1 Platz frei.

Fahrt suchen

Ziel

Abfahrtsort



- Der **Selektor** bestimmt, welche Elemente gemeint sind
- Jede **Deklaration** setzt eine **Eigenschaft** auf einen **Wert**, abgeschlossen mit `;`
- Eigenschaften heißen in CSS nicht Attribute. Attribute stehen im HTML: `class`, `href`, `alt`

Extern: so machen Sie es

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

- Eine Datei für alle Seiten
- Der Browser lädt sie einmal und cacht sie

Im Dokument oder am Element: Ausnahme

```
<style>
  p { color: green; }
</style>
<p style="color: blue">Inline</p>
```

- `<style>` gilt nur für diese eine Seite
- `style=""` mischt Darstellung ins HTML und schlägt fast jede Regel

Selektor	trifft	Beispiel
Typ	alle Elemente dieses Typs	<code>li</code>
Klasse	alle Elemente mit dieser Klasse	<code>.passed</code>
ID	das eine Element mit dieser <code>id</code>	<code>#leben-am-limes</code>
Attribut	Elemente mit diesem Attribut oder Wert	<code>[type="email"]</code>
Pseudoklasse	Elemente in einem Zustand oder an einer Position	<code>:hover</code> , <code>:focus</code> , <code>:first-child</code>
Universal	alle Elemente	<code>*</code>

```
<ul>
  <li class="passed">1234: 1,0</li>
  <li class="passed">1235: 1,7</li>
  <li class="passed" id="leben-am-limes">1236: 4,0</li>
  <li class="not-passed">1238: 5,0</li>
</ul>
```

```
.passed      { color: green; }
.not-passed  { color: red; }
#leben-am-limes { font-weight: bold; }
```

- Eine **Klasse** dürfen viele Elemente tragen, ein Element auch mehrere: `class="passed knapp"`
- Eine **ID** gibt es pro Seite genau einmal
- Für Stil fast immer Klassen. IDs für Sprungmarken, `label for` und JavaScript

Schreibweise	Name	trifft
<code>nav a</code>	Nachfahre	jedes <code>a</code> irgendwo innerhalb von <code>nav</code>
<code>ul > li</code>	Kind	nur <code>li</code> direkt in <code>ul</code>
<code>h2 + p</code>	Nachbar	das <code>p</code> direkt nach einem <code>h2</code>
<code>h2 ~ p</code>	Geschwister	alle <code>p</code> nach einem <code>h2</code> , gleiche Ebene
<code>li.passed</code>	zusammengesetzt	<code>li</code> , das auch die Klasse <code>passed</code> hat
<code>h1, h2</code>	Liste	<code>h1</code> und <code>h2</code> , das Komma heißt „und“

li

4 Treffer

Typselektor: alle Elemente dieses Typs.

HTML

```
<h1>Klausurergebnisse</h1>
<nav><a href="#start">Start</a> <a href="#noten">Noten</a></nav>
<p>Stand: <strong>Oktober</strong></p>
<ul>
  <li class="passed">1234: Note 1,0</li>
  <li class="passed">1235: Note 1,7</li>
  <li class="passed" id="leben-am-limes">1236: Note 4,0</li>
  <li class="not-passed">1238: Note 5,0</li>
</ul>
<h2>Einsicht anfragen</h2>
<form>
  <input type="text" name="name" placeholder="Name">
  <input type="email" name="mail" placeholder="E-Mail">
</form>
```

BROWSER

Klausurergebnisse

[Start](#) [Noten](#)

Stand: **Oktober**

- 1234: Note 1,0
- 1235: Note 1,7
- 1236: Note 4,0
- 1238: Note 5,0

Einsicht anfragen

Kurz testen: Was trifft der Selektor?

WIE VIELE ELEMENTE IN DIESEM HTML TRIFFT DER SELEKTOR?

a

```
<nav>
  <a href="/">Start</a>
  <a href="/fahrten" class="aktiv">Fahrten</a>
</nav>
<main>
  <p>Heute <a href="/neu">neu</a></p>
  <p class="hinweis">Abfahrt 7:45</p>
</main>
```

0

1

2

3

1/7



swe1.mylab.th-luebeck.de/web-03

Auf dem Handy mitmachen

2 Min

1 Regeln und Selektoren

3 Boxen

5 Selbst gestalten & Ausblick

2 **Die Kaskade**

4 Layout

Wer gewinnt, wenn sich Regeln streiten?

1 Wichtigkeit
!important vor normalen Regeln, Inline-Style vor Stylesheet

2 Spezifität
ID vor Klasse vor Typ: $(1,0,0) > (0,2,0) > (0,0,3)$

3 Reihenfolge
Gleich spezifisch? Die spätere Regel gewinnt

4 Vererbung
Nur wenn keine Regel das Element direkt trifft

Entschieden wird auf der ersten Stufe, auf der sich die Regeln unterscheiden

Selektor	IDs	Klassen, Attribute, Pseudoklassen	Typen, Pseudoelemente
#senden	1	0	0
.leiste .gefahr	0	2	0
nav button	0	0	2
nav button.knopf	0	1	2

Vergleich von links nach rechts, wie bei Versionsnummern: 1,0,0 schlägt 0,9,9

Duell 1 von 8: Typ gegen Klasse

REGEL 1 · STEHT WEITER OBEN

TRIFFT DAS ELEMENT

```
.hinweis { color: blue; }
```

0

IDs

1

Klassen, Attribute, :pseudo

0

Typen

HTML

FETT = DAS ELEMENT

```
<div class="karte">  
  <p class="hinweis">Abgabe am Sonntag</p>  
</div>
```

REGEL 2 · STEHT WEITER UNTEN

TRIFFT DAS ELEMENT

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

0

IDs

0

Klassen, Attribute, :pseudo

1

Typen

BROWSER

Abgabe am Sonntag

Regel 1 gewinnt. Höhere Spezifität: (0,1,0) schlägt (0,0,1). Verglichen wird Stelle für Stelle von links.

```
body {  
  font-family: system-ui, sans-serif;  
  color: #222;  
  border: 1px solid red;  
}
```

Jeder Absatz hat danach diese Schrift und Farbe, aber keinen eigenen Rahmen.

- **Erben:** Text-Eigenschaften wie `color`, `font-*`, `line-height`, `text-align`
- **Nicht erben:** Box-Eigenschaften wie `margin`, `padding`, `border`, `width`, `background`
- Erzwingen mit `inherit`, zurücksetzen mit `initial`
- Vererbt verliert immer gegen eine Regel, die das Element direkt trifft

Ungültiges CSS ignoriert der Browser still

Was stimmt hier nicht? ▾

CSS prüfen

Zurücksetzen

CSS7 DEKLARATIONEN

```
#d1 {
  color: white;
  background-color: darkblue;
}

#d2 {
  background-image: url("https://http.cat/images/100.jpg");
  background-size: cover;
  height: 200;
  width: 250;
  colour: white;
}
```

1. **Z. 9** „height: 200“: Einheit fehlt, z. B. 200px. Der Browser ignoriert die Zeile.
2. **Z. 10** „width: 250“: Einheit fehlt, z. B. 250px. Der Browser ignoriert die Zeile.
3. **Z. 11** „colour“ gibt es nicht (gemeint: color). Der Browser überspringt die Zeile.

BROWSER

Das ist div eins.

Kurz testen: Welche Regel gewinnt?

BEIDE REGELN TREFFEN DAS ELEMENT. WELCHE SETZT SICH DURCH?

Welche Farbe hat der Absatz?

```
<p class="hinweis">Abfahrt 7:45</p>
```

```
/* A */ p      { color: blue; }
```

```
/* B */ .hinweis { color: red; }
```

Regel A

Regel B

1 / 6



swe1.mylab.th-luebeck.de/web-03

Auf dem Handy mitmachen

2 Min

1 Regeln und Selektoren

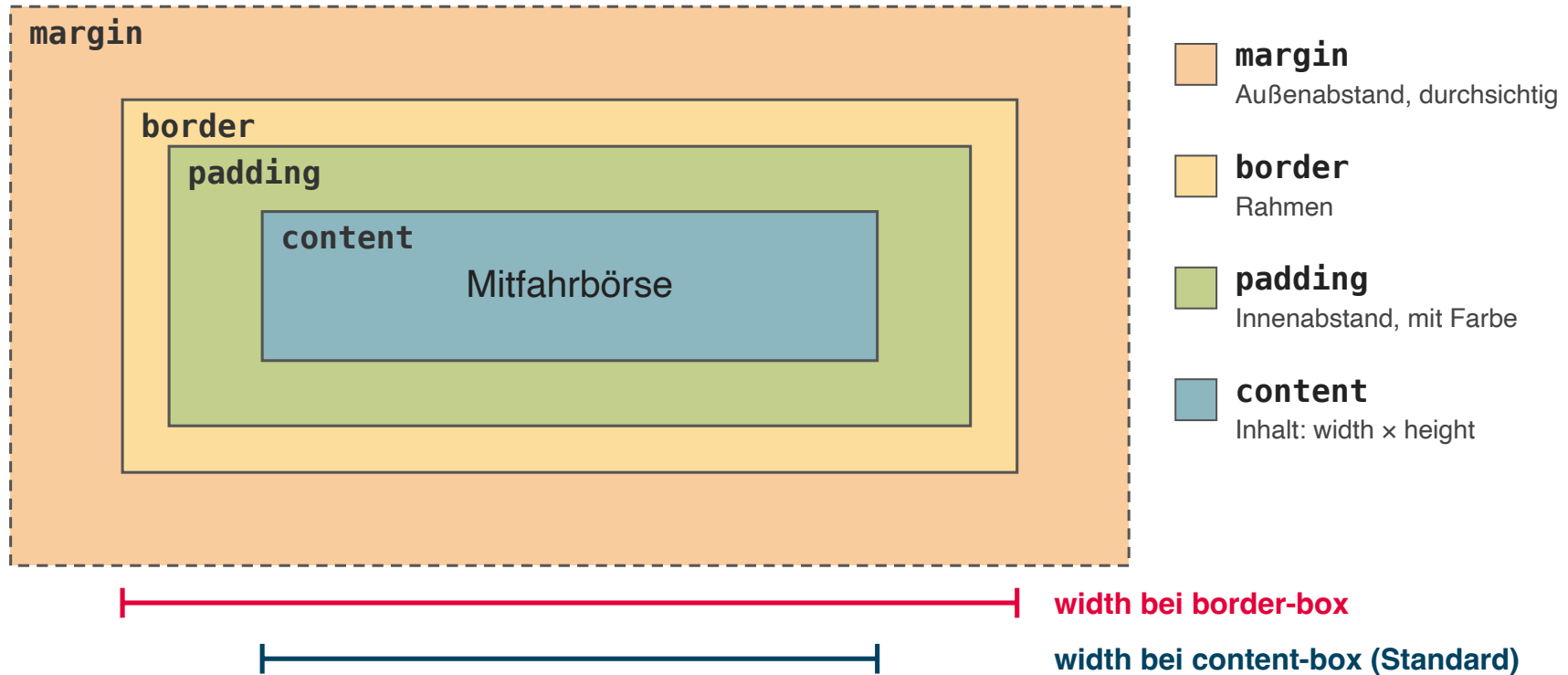
2 Die Kaskade

3 Boxen

4 Layout

5 Selbst gestalten & Ausblick

Wie viel Platz braucht ein Element?



```
*, *::before, *::after {  
  box-sizing: border-box;  
}
```

Steht in fast jedem Stylesheet ganz oben, auch in der Projektvorlage.

```
width: 200px; padding: 20px;  
border: 5px solid
```

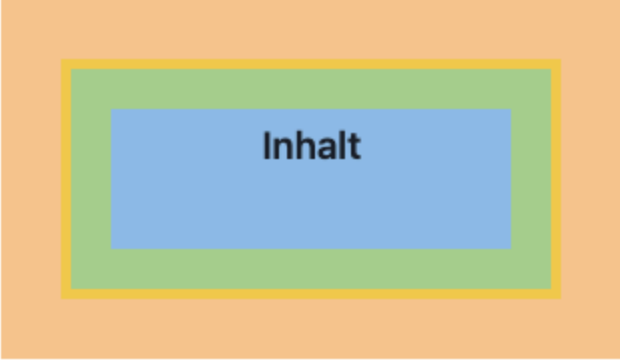
box-sizing	sichtbare Breite
content-box (Standard)	250 px
border-box	200 px

- Mit `border-box` ist `width` das, was man sieht
- Zwei Spalten mit `width: 50%` passen dann nebeneinander, auch mit Padding

BROWSER

marginborderpaddingInhalt

Text davor, damit Sie sehen, wie die Box im Fluss steht.



Text danach. Und noch ein bisschen mehr Text, der weiterläuft.

sichtbare Breite = width + 2·padding + 2·border
= 200 + 2·20 + 2·5 = **250 px** (gemessen: 250 px)
mit margin belegt die Box 310 px Platz

widthpaddingbordermargin

widthpaddingbordermargin

200px20px5px30px

box-sizing

content-box

border-box

display

block

inline

```
#box {  
  display: block;  
  box-sizing: content-box;  
  width: 200px;  
  height: 70px;  
  padding: 20px;  
  border: 5px solid;  
  margin: 30px;  
}
```

Einheit	bedeutet	nehmen Sie es für
px	Pixel (CSS-Pixel, nicht Bildschirmpunkte)	Rahmen, feine Details
rem	Vielfaches der Grundschrift (meist 16 px)	Schrift, Abstände
em	Vielfaches der Schrift des Elements	Abstände, die mit dem Text wachsen
%	Anteil am umgebenden Element	Breiten
vw , vh	Prozent der Fensterbreite, -höhe	bildschirmfüllende Bereiche
fr	Anteil am freien Platz im Grid	Grid-Spalten, kommt gleich

Farben

- Gleichwertig: `#e4003a`, `rgb(228 0 58)`, `hsl(345 100% 45%)`, `crimson`
- Text braucht Kontrast zum Hintergrund, mindestens 4,5 : 1

```
.status {  
  background: url("https://http.cat/100.jpg")  
              center / cover;  
  width: 250px;  
  height: 200px;  
}
```

Schrift

- `font-family` immer mit Ausweichschrift: `"Inter"`, `system-ui`, `sans-serif`
- `font-size`, `font-weight`, `line-height`, `text-align`
- Web-Fonts selbst ausliefern (`@font-face`), nicht von Google laden. Lizenz prüfen
- Am einfachsten: `system-ui`, die Schrift des Betriebssystems

1 Regeln und Selektoren

2 Die Kaskade

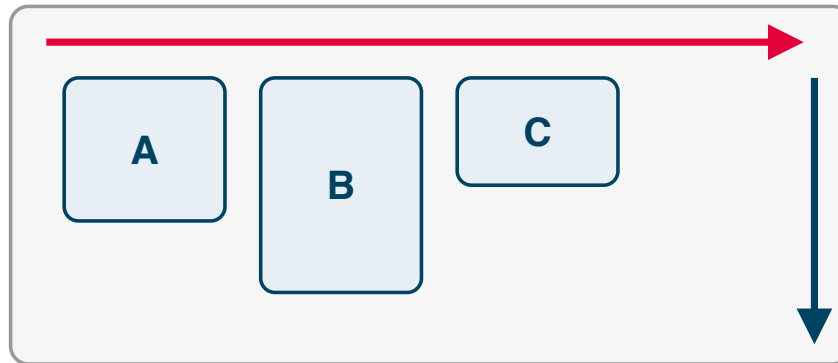
3 Boxen

4 **Layout**

5 Selbst gestalten & Ausblick

Wie kommt etwas dahin, wo ich es will?

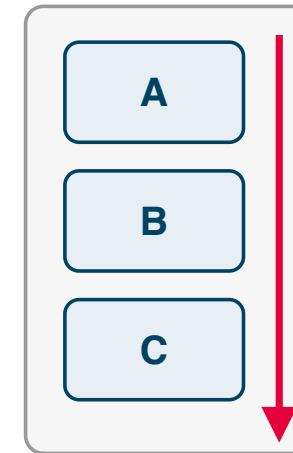
`flex-direction: row`



Hauptachse → `justify-content`

Querachse
→ `align-items`

`column`



Hauptachse jetzt senkrecht

- `display: flex` am **Container**, die Kinder ordnen sich auf der Hauptachse an
- `justify-content` verteilt entlang der Hauptachse, `align-items` quer dazu, `gap` setzt Abstände

Flexbox

Grid

flex-direction

row

justify-content

space-between

align-items

center

flex-wrap

nowrap

gap

8px

```
.container {  
  display: flex;  
  flex-direction: row;  
  justify-content: space-between;  
  align-items: center;  
  flex-wrap: nowrap;  
  gap: 8px;  
}
```

BROWSER

Breite: 772px

Hauptachse · justify-content

Querachse · align-items

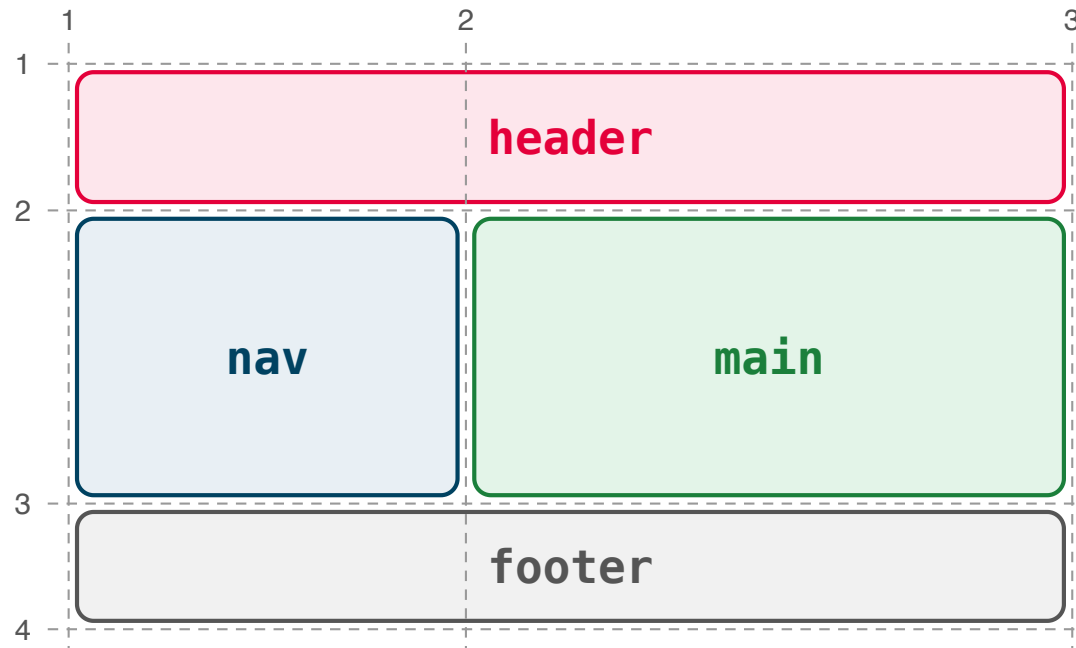
1

2

3 länger

4

5



```
.seite {  
  display: grid;  
  grid-template-columns:  
    190px 1fr;  
  grid-template-areas:  
    "header header"  
    "nav    main"  
    "footer footer";  
}
```

Gestrichelt: Grid-Linien, nummeriert ab 1

Flexbox

Grid

```
.seite {  
  display: grid;  
  grid-template-columns: 160px 1fr;  
  grid-template-areas:  
    "kopf kopf"  
    "nav inhalt"  
    "fuss fuss";  
  gap: 8px;  
}  
header { grid-area: kopf; }  
nav { grid-area: nav; }  
main { grid-area: inhalt; }  
footer { grid-area: fuss; }  
@media (max-width: 600px) {  
  .seite {  
    grid-template-columns: 1fr;  
    grid-template-areas:  
      "kopf" "nav" "inhalt" "fuss";  
  }  
}
```

BROWSER

Breite: 722px → Media Query inaktiv

header
kopf

nav
nav

main
inhalt

footer
fuss

```
.seite {  
  display: grid;  
  grid-template-areas:  
    "header" "nav" "main" "footer";  
}  
  
@media (min-width: 48rem) {  
  .seite {  
    grid-template-columns: 12rem 1fr;  
    grid-template-areas:  
      "header header"  
      "nav    main"  
      "footer footer";  
  }  
}
```

- **Mobil zuerst:** Die Grundregeln gelten fürs Handy, `@media` ergänzt für breite Fenster
- Die `viewport`-Zeile aus dem Grundgerüst gehört dazu, sonst rechnet das Handy mit 980 px
- Testen in den DevTools: Gerätesymbol, Breite ziehen
- Mit `auto-fill` und ohne feste Breiten braucht man oft gar keine Media Query

float

- Lässt Text um ein Bild fließen, wie in der Zeitung
- Dafür ist es gedacht, und nur dafür
- Spaltenlayouts mit float stammen aus der Zeit vor Flexbox

position

- `relative` : verschiebt, der Platz bleibt reserviert
- `absolute` : relativ zum nächsten positionierten Vorfahren, für Badges und Overlays
- `fixed` hängt am Fenster, `sticky` klebt beim Scrollen

1 Regeln und Selektoren

2 Die Kaskade

3 Boxen

4 Layout

5 Selbst gestalten & Ausblick

Jetzt Ihre Startseite

```
:root {  
  --accent: #003366;  
  --border: #c8ccd2;  
}  
@media (prefers-color-scheme: dark) {  
  :root {  
    --accent: #7fb2e5;  
    --border: #3a4048;  
  }  
}  
h1 { color: var(--accent); }
```

- `--name` deklariert eine Variable, `var(--name)` liest sie
- In `:root` deklariert, gilt sie überall
- Eine Farbe ändern, die ganze Seite passt sich an. Dark Mode ist dann ein Block
- DRY im Front-End: steht so im Leitfaden für P4

- **Kontrast:** Text mindestens 4,5 : 1 zum Hintergrund. Die DevTools zeigen es im Farbwähler
- **Fokus sichtbar lassen:** nie `outline: none` ohne Ersatz. Gestalten mit `:focus-visible`
- **Nicht nur Farbe:** Fehler rot *und* mit Text oder Symbol
- **Schrift in `rem`**, damit die Einstellung im Browser greift
- **`:hover` ist kein Muss:** Auf dem Handy gibt es keinen Mauszeiger

Ihre HTML-Seite von letzter Woche, jetzt mit CSS:

1. Eine Farbpalette aus drei Variablen in `:root`
2. Die Navigation als Flexbox-Reihe
3. Ein Grid-Layout für `header`, `nav`, `main`, `footer`
4. Eine Media Query: schmal einspaltig
5. In den DevTools nachsehen, welche Regel wo gewinnt

12 Min

allein oder zu zweit



swe1.mylab.th-luebeck.de/web-03

dort „CSS-Werkstatt“. Eigenes HTML hineinkopieren, „Als Datei speichern“ nicht vergessen

Auflösung: Warum ist der Button rot?

Leitfrage: Warum ist der Button rot? ▾

CSS prüfen

Zurücksetzen

HTML

```
<nav class="leiste">
  <button id="senden"
    class="knopf gefahr">
    Senden
  </button>
</nav>
```

CSS

5 DEKLARATIONEN

```
.leiste .gefahr {
  background: red;
}

button {
  background: blue;
  color: white;
  font-size: 2rem;
}

.knopf {
  background: green;
}
```

BROWSER

Senden

Verschachteln

```
.karte {  
  padding: 1rem;  
  & h2 { margin: 0; }  
}
```

Wie in Sass, jetzt direkt im Browser

:has()

```
form:has(:invalid) button {  
  opacity: .5;  
}
```

Der Eltern-Selektor, den es lange nicht gab

Container Queries

```
@container (min-width: 30rem) {  
  .karte { display: flex; }  
}
```

Reagiert auf den Platz der Komponente

- Eine Regel ist Selektor plus Deklarationen. Eigenschaften gehören ins CSS, Attribute ins HTML
- Spezifität vor Reihenfolge. Klassen statt IDs, kein `!important`
- `box-sizing: border-box`, dann ist `width` das, was man sieht
- Flexbox für eine Reihe, Grid für die Fläche, `@media` für schmale Fenster
- Farben und Abstände als Variablen in `:root`
- Sieht etwas anders aus als gedacht: DevTools, Styles

1. P2 läuft: Anforderungen bis **So 15.11.**, Leitfaden Phase 2 lesen
2. Ihre Startseite mit CSS fertig machen und aufheben, nächste Woche kommt JavaScript dazu
3. Zum Üben: CSS Escape Room und Selektor-Spielplatz unter `swe1.mylab.th-luebeck.de/web-03`

Di 20.10. bei Leif

- Projekt und Anforderungen, Teil 1

Mi 21.10.

- JavaScript, Teil 1