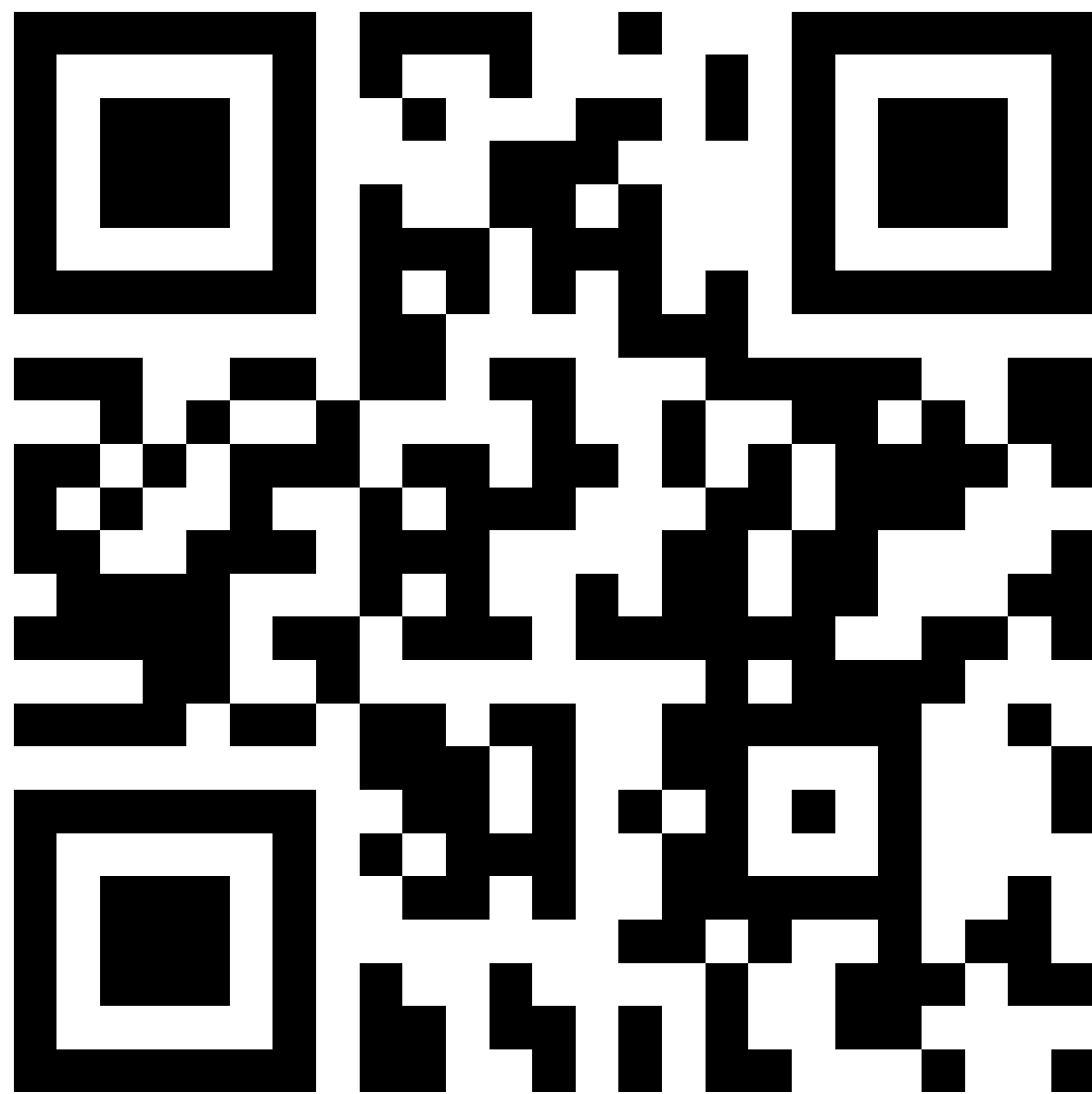


Web Engineering

DHBW Stuttgart · Informatik / Wirtschaftsinformatik

Sommersemester 2026 · Sitzung 2 — HTML & CSS

<https://librete.ch/dhbw/>



Ressourcen zum Selbstlernen

- **CODE CRISPIES** – codecrispi.es
- **Online Code-Editor** – codepen.io
- **MDN** (Mozilla Developer Network) – developer.mozilla.org/de/
- **Flexbox-Spiel** – flexboxfroggy.com
- **Grid-Spiel** – cssgridgarden.com

HTML – Hypertext Markup Language

Was ist HTML? (1/2)

HyperText Markup Language

- **Keine** Programmiersprache
- Am ehesten eine Formattierungssprache
- Beschreibt **Struktur und Aufbau** einer Webseite

```
<p>Das ist ein Absatz.</p>  
<input type="date" />
```

Erfunden 1989 von Tim Berners-Lee. Aktuelle Version: HTML5 (seit 2014).

Was ist HTML? (2/2)

- Mit HTML werden **HTML-Dokumente** beschrieben (`index.html`)
- Ein HTML-Dokument beschreibt das **Document Object Model (DOM)**
- Der DOM ist eine **hierarchische Baumstruktur** (Eltern, Kinder, Geschwister)
 - *parent* nodes / elements
 - *child* nodes / elements
 - *sibling* nodes / elements

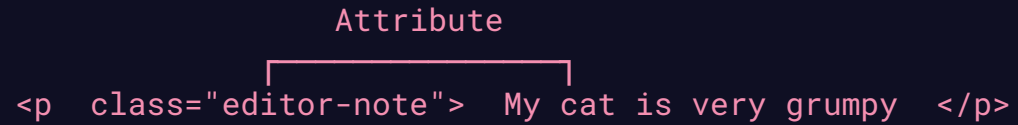
In HTML sprechen wir von **Elementen**, in JavaScript eher von **Nodes**.

HTML-Anatomie (1/2)



- **Opening tag + Content + Closing tag = Element**
- Öffnende und schließende Tags müssen identisch sein
- Schließendes Tag hat ein `/` nach der Spitzklammer

HTML-Anatomie – Attribute



The diagram shows the HTML opening tag `<p class="editor-note">` followed by the text `My cat is very grumpy` and the closing tag `</p>`. A bracket above the `class="editor-note"` part is labeled "Attribute", indicating that attributes are placed within the opening tag.

```
<p class="editor-note"> My cat is very grumpy </p>
```

- **Attribute** stehen im opening tag
- Format: `name="wert"` (Anführungszeichen empfohlen)
- Mehrere Attribute durch Leerzeichen getrennt: `<input type="text" name="email" required>`

HTML-Anatomie (2/2)

Wir unterscheiden **umschließende** und **selbst-schließende** Tags.

Umschließend:

- `<b> ... </b>` – fett-gedruckter Text (inline)
- `<a> ... </a>` – Hyperlink
- `<body> ... </body>` – sichtbarer Teil eines HTML-Dokuments
- `<details> ... </details>` – Aufklapp-Text / Akkordion

Selbst-schließend:

- `<img />` – Bild, Vektorgrafik
- `<input />` – Eingabefeld
- `<meta />` – Metadaten

Darstellungsebene und Metadaten

```
<html>
  <head>
    METADATEN
  </head>
  <body>
    DARSTELLUNGSEBENE
  </body>
</html>
```

Ein HTML-Dokument besteht aus zwei Teilen:

- `<head> ... </head>` – Metadaten für Browser, Suchmaschinen, Social Sharing
- `<body> ... </body>` – darzustellender Inhalt, Text, Formulare

HTML Metadaten

```
<html>
  <head>
    <title>DHBW Stuttgart</title>
    <meta name="description" content="Studium mit Praxis" />
    <meta property="og:image" content="https://..." />
  </head>
  ...
</html>
```

Webseiten besuchen nicht nur Menschen, sondern auch Suchmaschinen, Bots, Screen-Reader. Metadaten helfen beim "Verstehen" der Inhalte.

HTML Metadaten – Vertiefung

Tag	Funktion
<code><title></code>	Browser-Tab, Suchergebnis-Titel
<code><meta charset></code>	Zeichenkodierung (Umlaute!)
<code><meta viewport></code>	Mobile Darstellung
<code><meta description></code>	Suchergebnis-Snippet (max 160 Zeichen)

Open Graph (Social-Media-Vorschau):

```
<meta property="og:title" content="Artikel-Titel">
<meta property="og:image" content="https://example.com/preview.jpg">
```

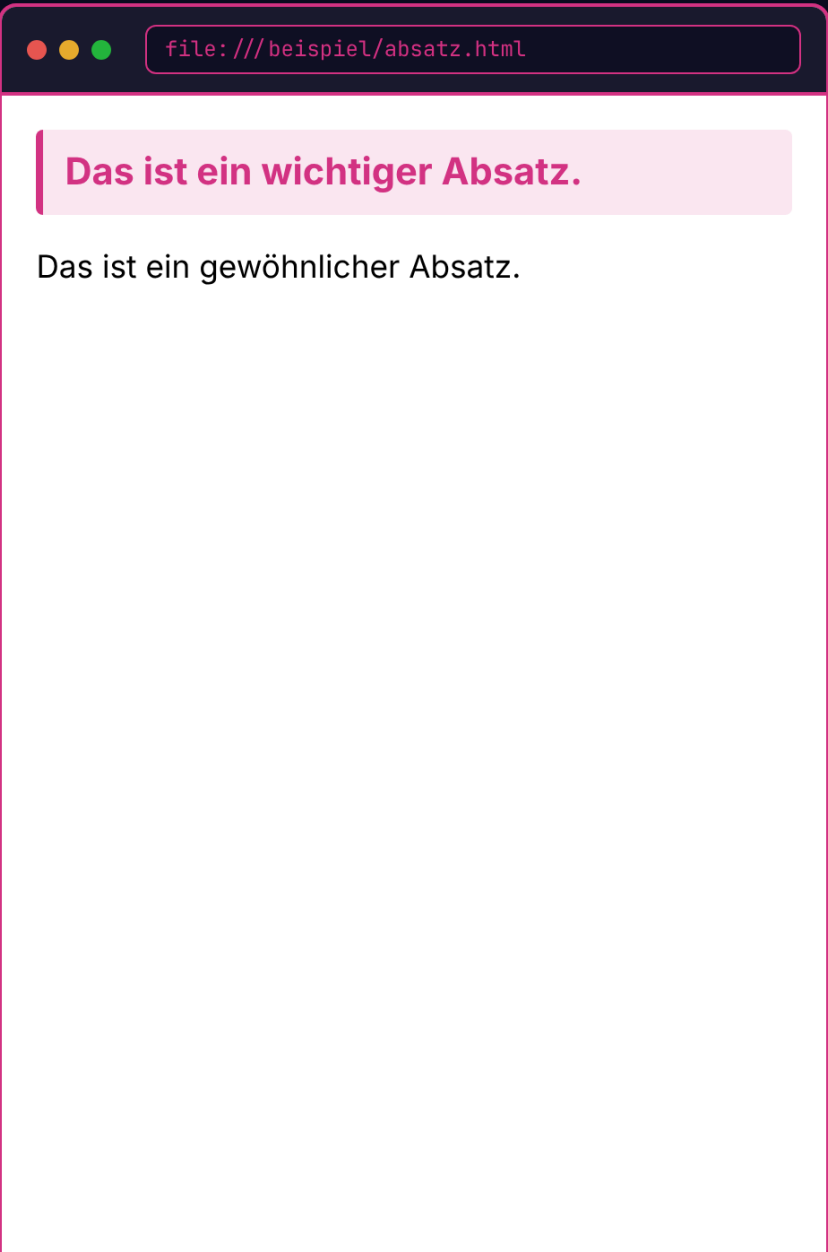
→ Wirkt auf Mastodon, LinkedIn, Signal, Slack, MS Teams ...

SEO: Google nutzt `<title>` + `<meta description>` für Ranking + Snippet.

HTML-Tags und Attribute

```
<body>
  <p class="wichtig">Das ist ein wichtiger Absatz.</p>
  <p>Das ist ein gewöhnlicher Absatz.</p>
</body>
```

Teil	Bezeichnung
<p>	Opening Tag
class="wichtig"	Attribut
Das ist ein wichtiger Absatz.	Inhalt
</p>	Closing Tag



HTML-Tags und Attribute (selbst-schließend)

```
Geburtsdatum <input type="date" />
```

Teil	Bezeichnung
<code><input ...</code>	Eingabefeld
<code>type="date"</code>	Attribut <i>type</i> mit Wert <i>date</i>
<code>... /></code>	Schließendes <code>/</code> im (öffnenden) Tag

file:///beispiel/input.html

Geburtsdatum

21.04.2026 

< April 2026 >

Mo	Di	Mi	Do	Fr	Sa	So
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3

Vollständiges Grundgerüst

```
<!DOCTYPE html>
<html lang="de">
<head>
  <title>Portfolio von ...</title>
  <meta name="description" content="..." />
  <meta property="og:image" content="https://...." />
</head>
<body>
  <h1>Hallo Welt!</h1>
  <p>Das ist mein erster Absatz.</p>
</body>
</html>
```

file:///portfolio/index.html

Hallo Welt!

Das ist mein erster Absatz.

Die wichtigsten Tags

Tag	Bedeutung
<code><h1> ... <h6></code>	Überschriften (h1 = größte)
<code><p></code>	Absatz (Paragraph)
<code> ... </code>	Hyperlink
<code></code>	Bild
<code></code> , <code></code> , <code></code>	Listen
<code><div></code>	Container / Box
<code></code>	Inline-Container
<code><input type="..."></code>	Eingabefeld

Semantische Tags (`<header>`, `<main>`, `<article>`, `<footer>`, `<nav>`, `<aside>`) wichtig für Accessibility.

Natives HTML

Was früher mit JavaScript gelöst wurde, kann HTML heute selbst. JS dafür einzusetzen ist heute oft ein **Code Smell**.

Funktion	HTML-Element
Aufklapp-Text	<code><details></code> + <code><summary></code>
Popup / Modal	<code><dialog></code>
Fortschrittsbalken	<code><progress></code>
Datum-Auswahl	<code><input type="date"></code>
Validierung	Attribute: <code>required</code> , <code>min</code> , <code>max</code> , <code>pattern</code>

Beispiel: Details/Summary

Akkordion · Accordion · Aufklappmenü · Collapsible

```
<details>
  <summary>Klick mich!</summary>
  <p>Dieser Text ist versteckt.</p>
</details>
```

Doku: developer.mozilla.org



Beispiel: Details/Summary (geöffnet)

```
<details open>
  <summary>Klick mich!</summary>
  <p>Dieser Text war versteckt.</p>
</details>
```



Beispiel: Button

```
<button>Add to favorites</button>
```

Doku: developer.mozilla.org



Beispiel: Input/Eingabefeld

```
<input  
  type="text"  
  maxlength="8"  
  size="10" />
```

Doku: developer.mozilla.org

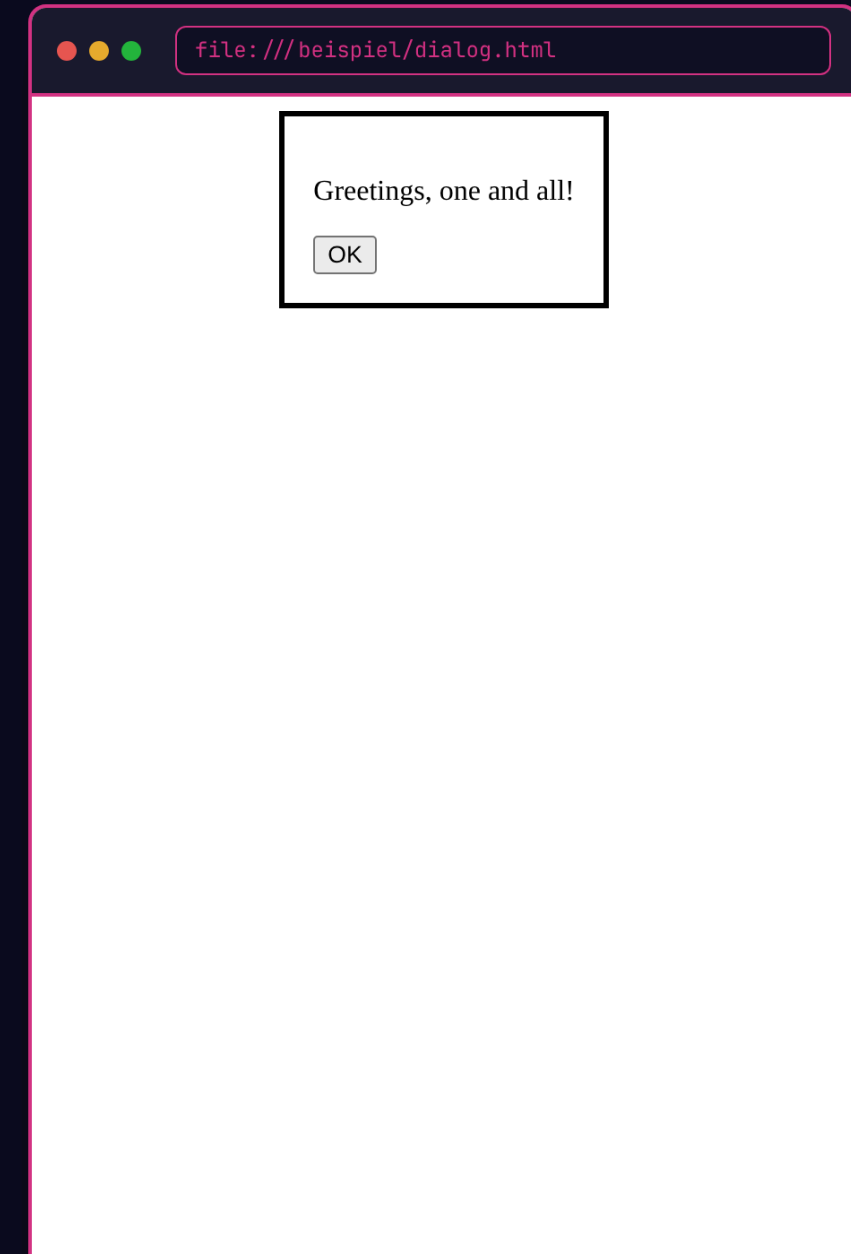


Beispiel: Dialog

Modal · Modales Fenster · Popup · Overlay · Dialogfenster

```
<dialog open>
  <p>Greetings, one and all!</p>
  <form method="dialog">
    <button>OK</button>
  </form>
</dialog>
```

Doku: developer.mozilla.org



Barrierefreiheit

Accessibility (a11y)

Wie nutzen Menschen das Web?

Eingabe	Nutzungsweise
Maus	Klicken, Scrollen
Tastatur	Tab-Navigation, Enter, Pfeiltasten
Screenreader	NVDA, VoiceOver, TalkBack, Orca
Sprachsteuerung	"Klicke auf Anmelden"
Augensteuerung	Eye-Tracking
Switch-Geräte	Ein-/Aus-Schalter (motorische Einschränkungen)

→ Nicht alle Menschen nutzen Maus oder Touchscreens gleichermaßen.

Web-Zugänglichkeit – Statistik

WHO: ca. **15 % der Weltbevölkerung** mit Behinderung – über **1 Milliarde** potenzielle Nutzer:innen.

Typ	Permanent	Temporär	Situativ
Visuell	Blindheit	Augen-OP	Grelle Sonne
Motorisch	Amputation	Gebrochener Arm	Kind auf dem Arm
Auditiv	Taubheit	Ohrenentzündung	Laute Umgebung
Kognitiv	Legasthenie	Müdigkeit	Ablenkung

Curb-Cut-Effekt: Bordsteinabsenkung für Rollstühle hilft auch Kinderwagen, Rollkoffern, Fahrrädern.

Warum Barrierefreiheit?

Rechtlich (DE / EU):

- **EAA** – European Accessibility Act, in Kraft seit 28.06.2025
- **BFSG** – DE-Umsetzung des EAA, betrifft auch Privatwirtschaft
- **BITV 2.0** – Barrierefreie Informationstechnik-Verordnung (öff. Stellen)
- Bußgelder bis **100.000 €**

Ethisch: Teilhabe und digitale Inklusion.

Praktisch:

- Bessere UX für **alle** (mobile, ältere Menschen, situative Einschränkungen)
- SEO-Bonus durch semantisches HTML, Alt-Texte
- Größerer Markt – ca. 15 % der Bevölkerung

WCAG: Der Standard

Web Content Accessibility Guidelines · Herausgeber: W3C / WAI

Aktuell: WCAG 2.2 (verbindlich seit Okt. 2023). WCAG 3 in Working-Draft.

4 Prinzipien (POUR):

Prinzip	Bedeutung
P erceivable	Wahrnehmbar – Inhalte müssen erkennbar sein
O perable	Bedienbar – Funktionen müssen nutzbar sein
U nderstandable	Verständlich – Inhalte müssen klar sein
R obust	Robust – funktioniert mit verschiedenen Technologien

Konformitätslevel: A · **AA** (gesetzlicher Standard) · AAA

→ [w3.org/WAI/WCAG22/quickref](https://www.w3.org/WAI/WCAG22/quickref)

Perceivable: Wahrnehmbar

Texte:

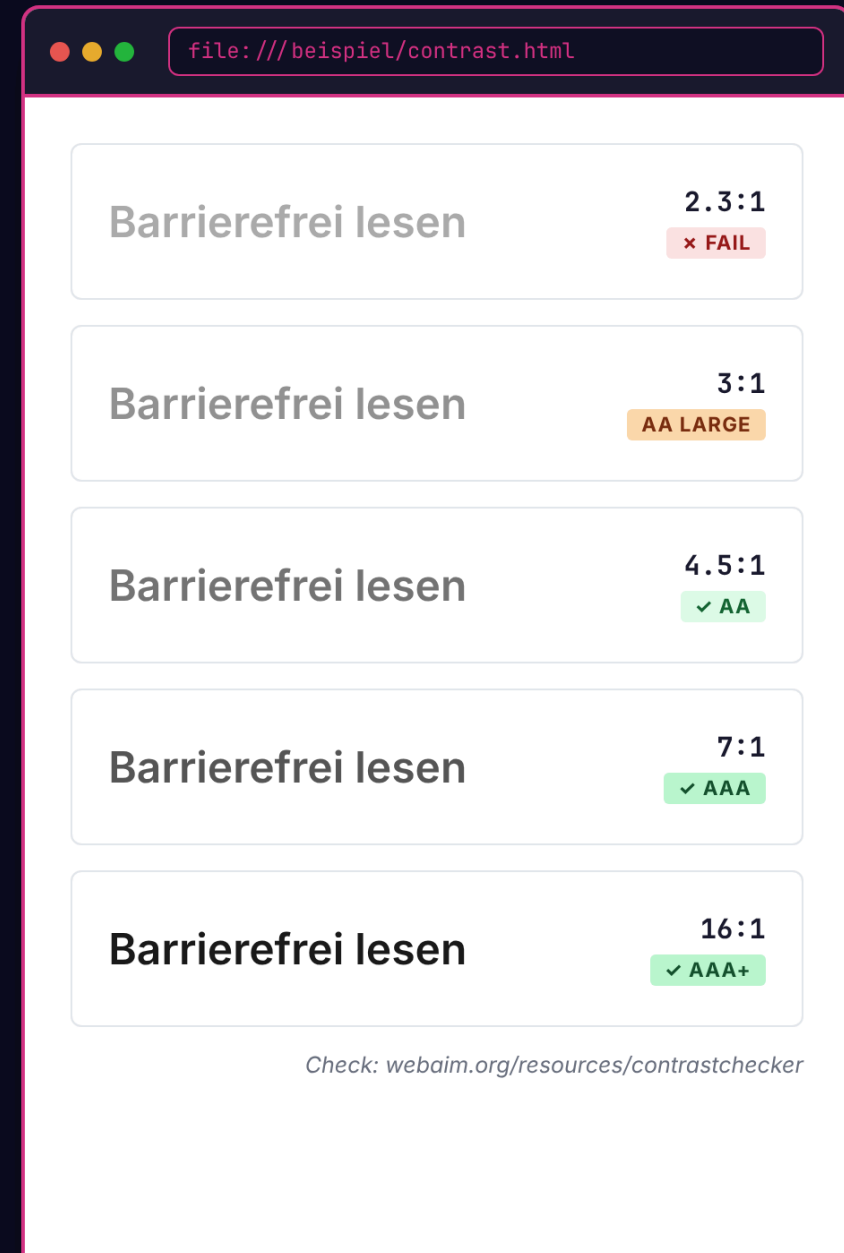
- **Kontrast:** mindestens 4,5:1 für normalen Text
- **Untertitel** für Videos
- Inhalte bei **200 % Zoom** noch nutzbar

Textalternativen für Nicht-Text:

```

```

→ Leeres `alt=""` für rein dekorative Bilder.



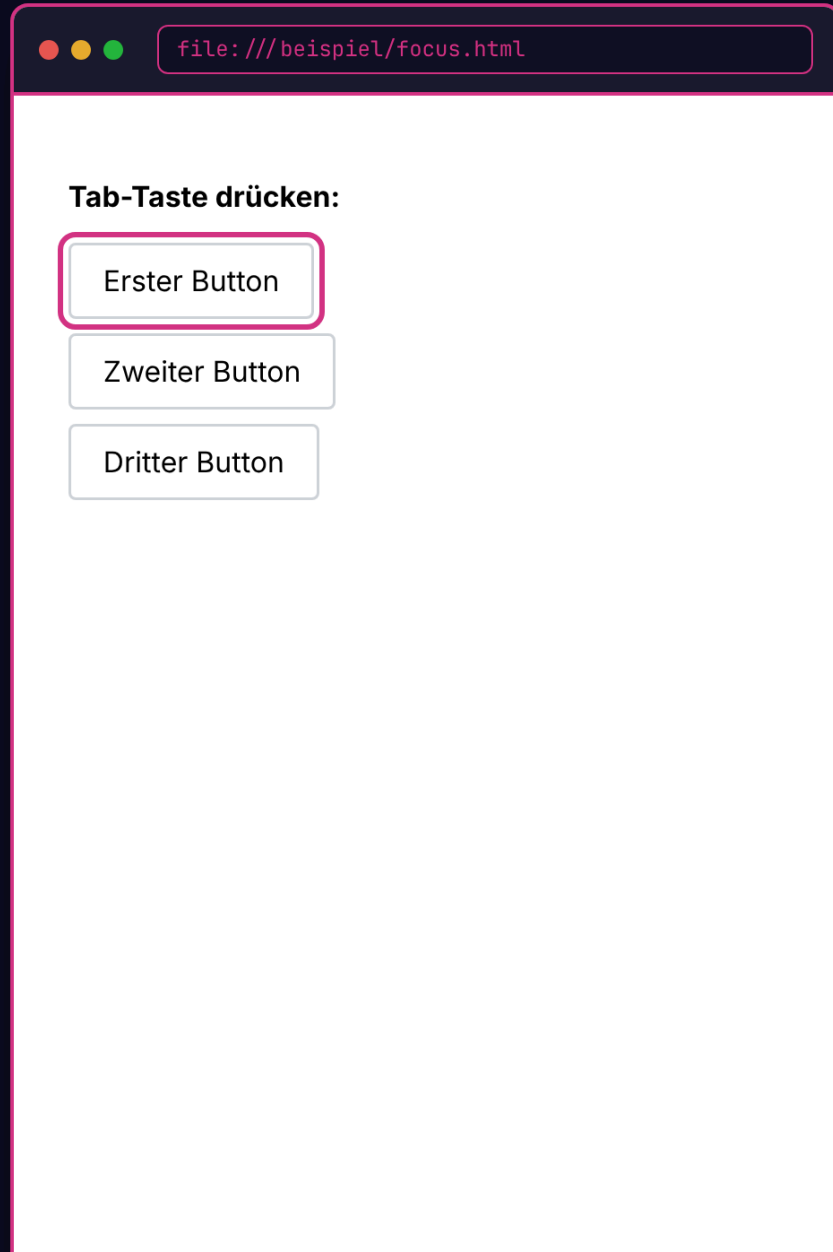
Operable: Bedienbar

Tastaturzugänglich:

```
/* Fokus-Ring nur bei Tastatur-Navigation */  
button:focus-visible {  
  outline: 3px solid #d63384;  
  outline-offset: 3px;  
  border-radius: 4px;  
}  
  
/* Anti-Pattern */  
button:focus { outline: none; }
```

Genügend Zeit: Session-Timeouts ankündigen, Verlängerung ermöglichen.

Kein Flackern: maximal 3×/Sekunde (epileptische Anfälle vermeiden).



Understandable: Verständlich

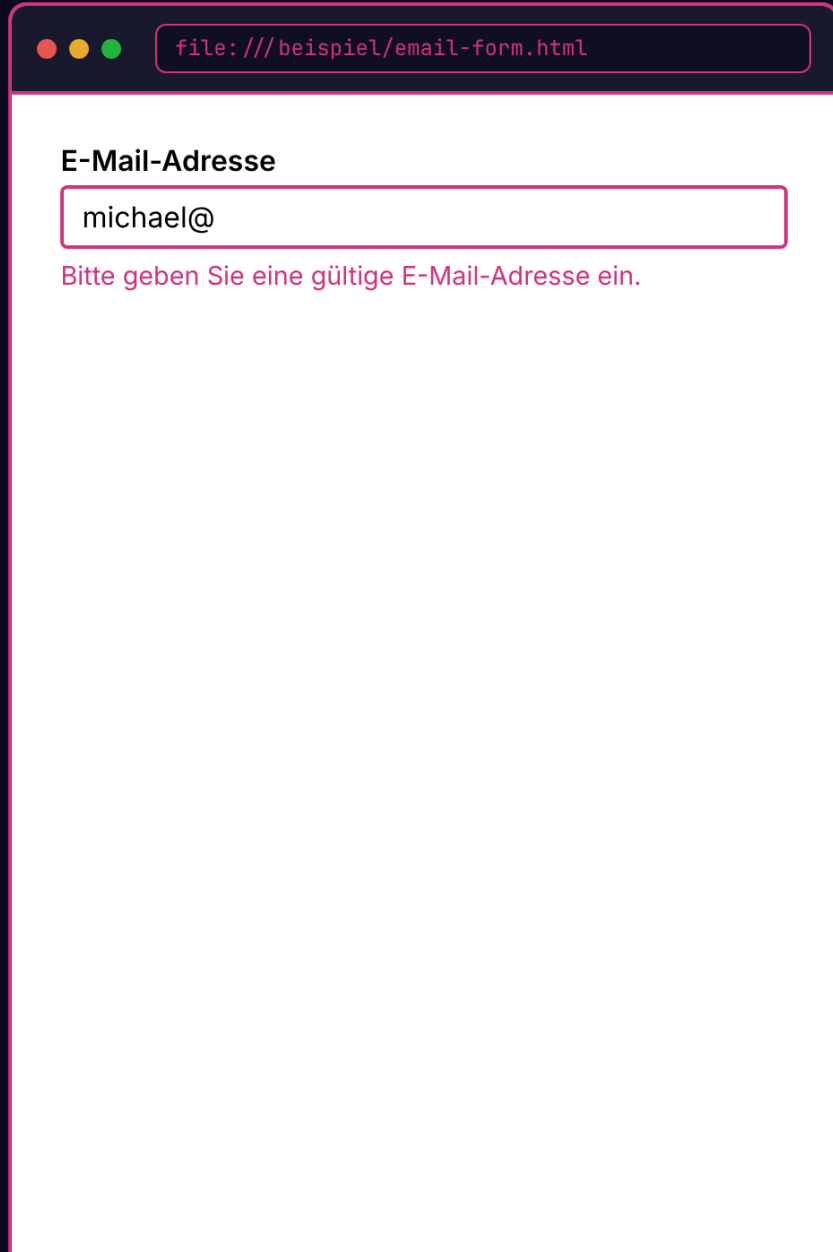
Sprache angeben:

```
<html lang="de">
```

Konsistente Navigation – gleiche Elemente an gleicher Stelle.

Fehler erklären:

```
<input type="email" aria-describedby="email-error">  
<p id="email-error" role="alert">  
  Bitte eine gültige E-Mail-Adresse eingeben.  
</p>
```



The screenshot shows a web browser window with the address bar displaying 'file:///beispiel/email-form.html'. The page content includes a label 'E-Mail-Adresse' above a text input field containing 'michael@'. Below the input field, a red error message reads: 'Bitte geben Sie eine gültige E-Mail-Adresse ein.' The browser's window controls (red, yellow, and green buttons) are visible in the top-left corner.

Robust: Technisch solide

Semantisches HTML:

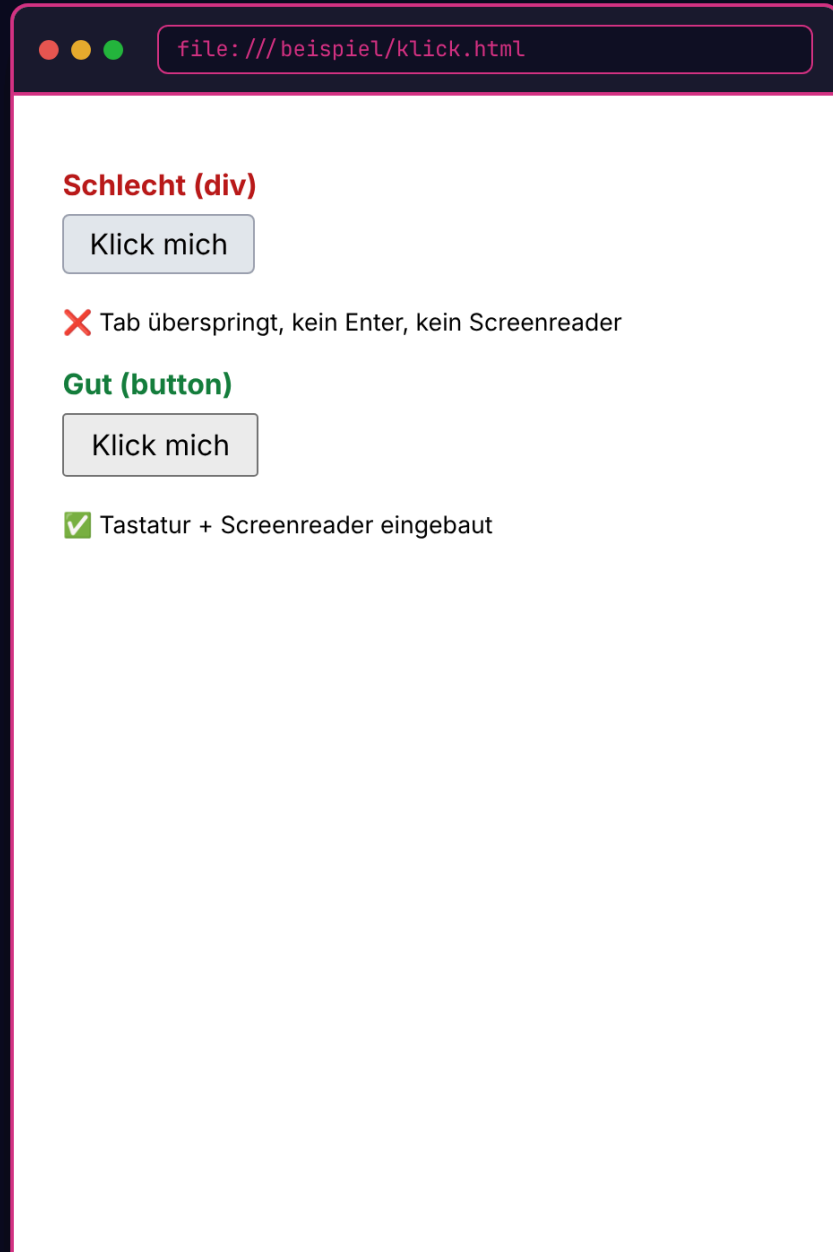
```
<!-- Schlecht -->
<div class="button" onclick="...">Klick mich</div>

<!-- Gut -->
<button type="button">Klick mich</button>
```

ARIA – nur wenn nötig:

```
<div role="tablist" aria-label="Produktkategorien">
  <button role="tab" aria-selected="true">Schuhe</button>
  <button role="tab" aria-selected="false">Jacken</button>
</div>
```

→ Erste ARIA-Regel: **wenn möglich, natives HTML nutzen.**



Barrierefreiheit testen – Werkzeuge

Automatisierte Tools (finden ca. 30 % der Verstöße):

Tool	Typ	Findet
WAVE	Browser-Extension	Kontrast, Struktur, Tab-Reihenfolge
Lighthouse	Chrome DevTools	Performance + A11y-Audit
axe DevTools	Browser-Extension	Detailanalyse, Regel-Engine
Pa11y	CLI	CI/CD-Integration

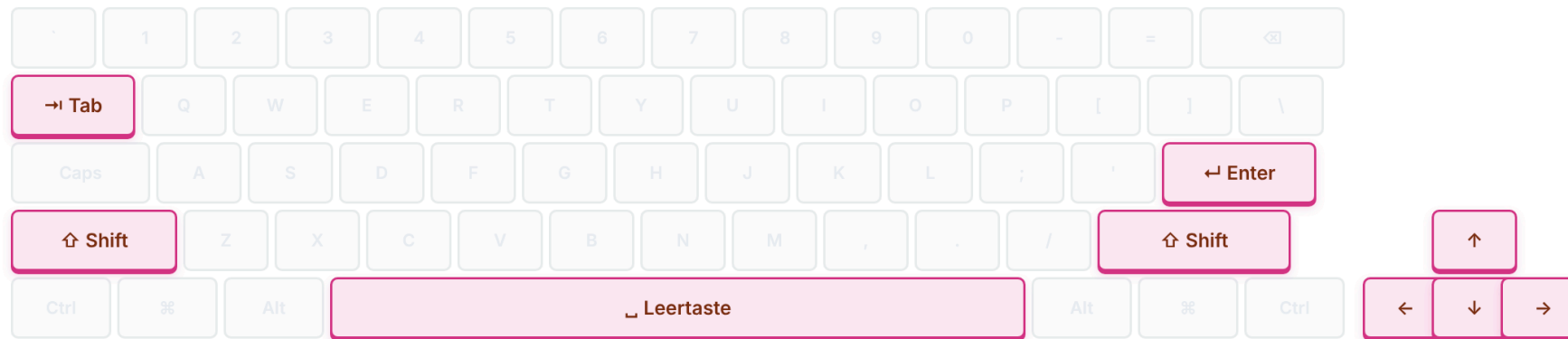
Screenreader (OS-nativ oder Open Source):

OS	Screenreader
macOS / iOS	VoiceOver (Cmd + F5) – builtin
Windows	NVDA (OSS) · Narrator (builtin)
Linux	Orca – GNOME builtin
Android	TalkBack – builtin

Barrierefreiheit testen

Automatisierte Tools finden nur etwa **30 %** der Probleme. Manuelles Testen unverzichtbar – Gold-Standard: echte Nutzer:innen mit Assistiven Technologien.

Tastatur-Test – Maus weglegen, nur Tab / Shift+Tab / Enter / Pfeiltasten:



Navigation

- Tab** – zum nächsten fokussierbaren Element
- ⇧ → Shift + Tab** – ein Element zurück
- ↑ ↓ ← → Pfeiltasten** – scrollen, innerhalb Listen/Menüs navigieren

Aktion

- ↵ Enter** – Link/Button aktivieren
- ⌵ Leertaste** – Checkbox toggle, Dropdown öffnen, scrollen

CSS – Cascading Style Sheets

Was ist CSS?

Cascading Style Sheets

```
p {  
  color: blue;  
  font-size: 16px;  
}
```

- Trennt **Inhalt** (HTML) von **Darstellung** (CSS)
- "Cascading" = Regeln können überschrieben werden
- Eine CSS-Datei kann viele HTML-Seiten stylen

CSS einbinden

Option 1: Externe Datei (empfohlen)

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

Option 2: Style-Tag

```
<style>p { color: blue; }</style>
```

Option 3: Inline (vermeiden)

```
<p style="color: blue;">...</p>
```

→ Standard ist externe Datei. Inline vermeiden (schwer zu warten, keine Wiederverwendung).

CSS-Anatomie

```
/* ID selector */  
#description {  
  color: orangered;  
  /*      |      Indicates the value of the expression  
  /*      |____ Indicates the property of the expression */  
}
```

- **Selector** – was wird gestylt (`#description` = Element mit `id="description"`)
- **Property** – welche Eigenschaft (`color`)
- **Value** – welcher Wert (`orangered`)
- **Declaration** – `property: value;` (Semikolon trennt Deklarationen)
- **Declaration block** – alles zwischen `{` und `}`

Selektoren: Element

```
/* Alle <p>-Elemente */
p {
  color: gray;
}

/* Mehrere Elemente gleichzeitig */
h1, h2, h3 {
  font-family: sans-serif;
}
```

Element-Selektoren sind die einfachsten. Komma = "oder".

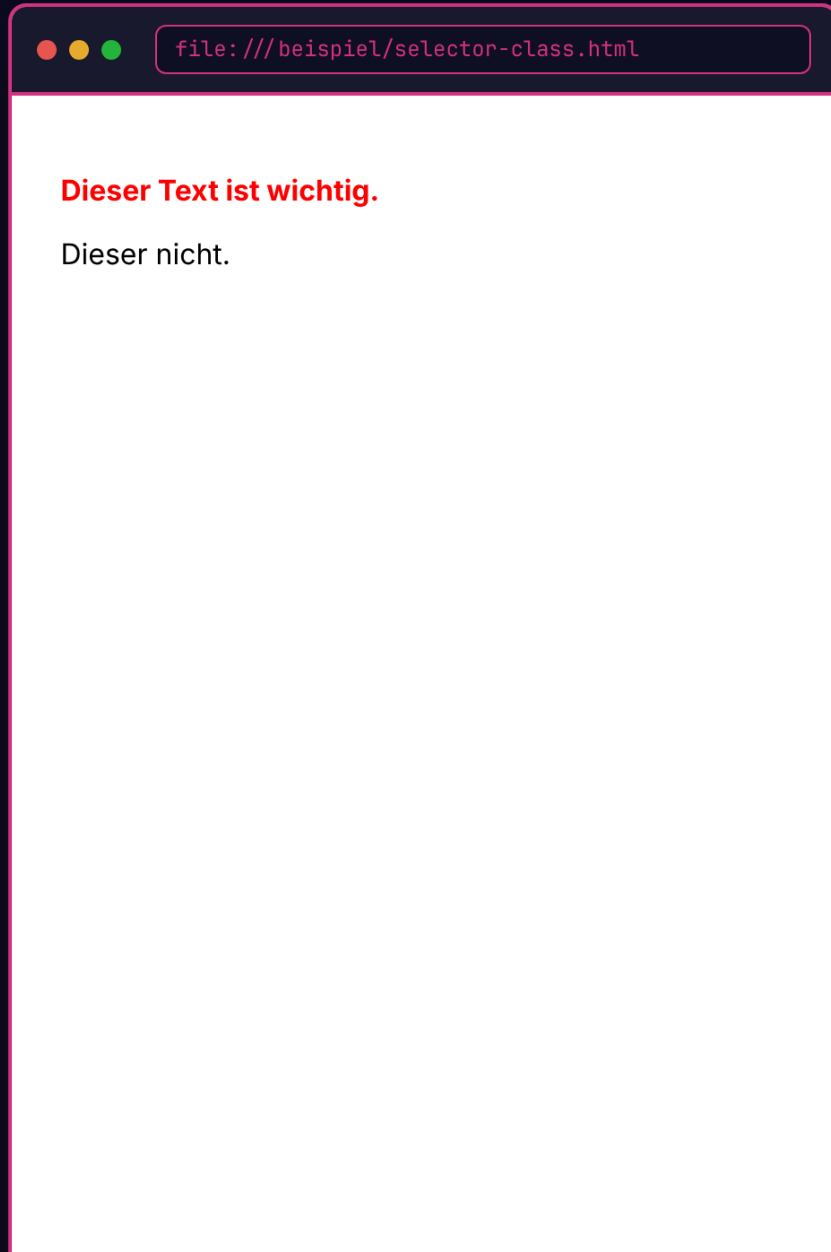


Selektoren: Klasse

```
<p class="wichtig">Dieser Text ist wichtig.</p>
<p>Dieser nicht.</p>
```

```
.wichtig {
  color: red;
  font-weight: bold;
}
```

Punkt vor dem Namen = Klasse. Beliebig oft anwendbar; ein Element kann mehrere Klassen haben.



Selektoren: ID

```
<nav id="hauptnavigation">...</nav>
```

```
#hauptnavigation {  
  background: #333;  
  padding: 1rem;  
}
```

Raute vor dem Namen = ID.

Hinweis: IDs sollten **einmalig** pro Seite sein. In der Praxis: Klassen meist besser.



Selektoren: Kombinationen

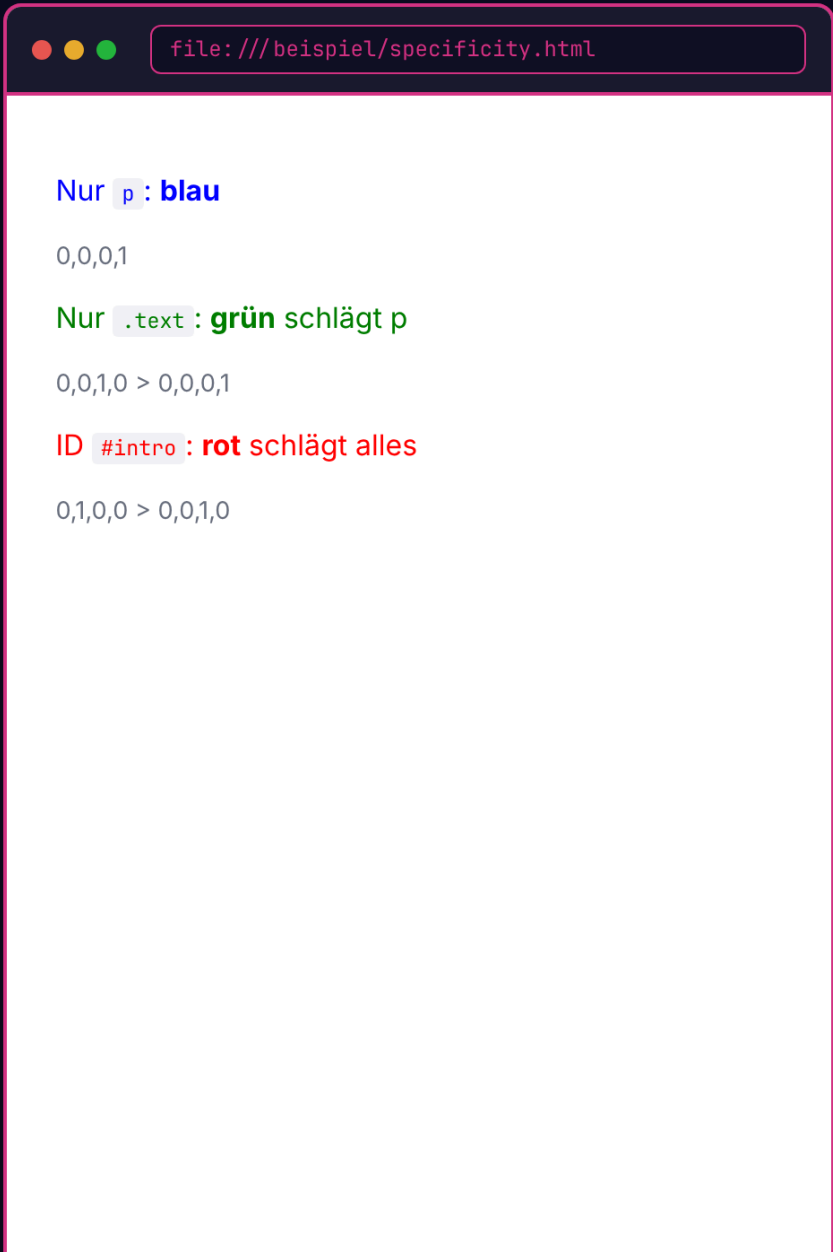
```
/* Nachfahre (beliebig tief verschachtelt) */  
article p { line-height: 1.6; }  
  
/* Direktes Kind (eine Ebene) */  
nav > a { text-decoration: none; }  
  
/* Nächstes Geschwister */  
h2 + p { font-size: 1.2rem; }  
  
/* Element mit Klasse */  
p.wichtig { color: red; }
```



Spezifität: Welche Regel gewinnt?

Selektor	Spezifität
Element (p)	0,0,0,1
Klasse (.wichtig)	0,0,1,0
ID (#header)	0,1,0,0
Inline (style="...")	1,0,0,0

```
p { color: blue; } /* 0,0,0,1 */
.text { color: green; } /* 0,0,1,0 → gewinnt */
#intro { color: red; } /* 0,1,0,0 → gewinnt über beide */
```



CSS Spezifität – Vertiefung

Selektor	Inline	IDs	Klassen	Elemente	Gesamt
<code>p</code>	0	0	0	1	0,0,0,1
<code>.info</code>	0	0	1	0	0,0,1,0
<code>p.info</code>	0	0	1	1	0,0,1,1
<code>#header</code>	0	1	0	0	0,1,0,0
<code>#header .nav a</code>	0	1	1	1	0,1,1,1

Regeln:

- Eine Klasse (0,0,1,0) schlägt **jede Anzahl** Elemente (0,0,0,99)
- Eine ID schlägt jede Anzahl Klassen
- `!important` bricht alles → vermeiden

Best Practice: Flache Spezifität. BEM-Methodik (`.block__element--modifier`) hält sie niedrig.

Box-Modell

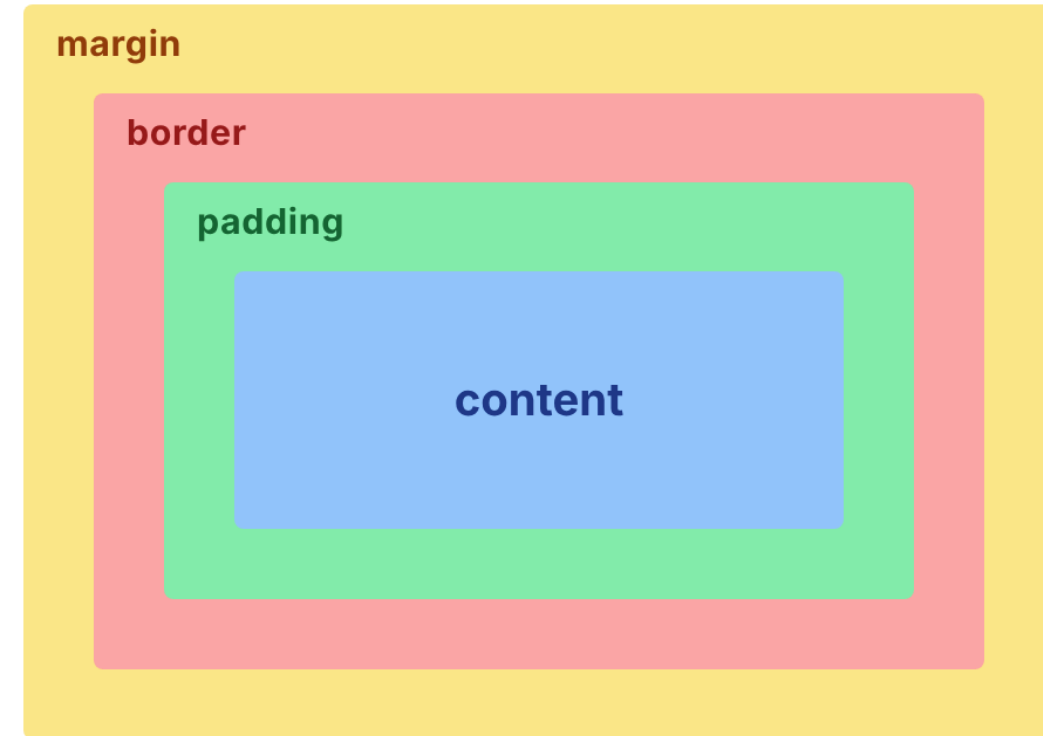
Jedes HTML-Element ist eine rechteckige **Box**.

Schichten innen → außen:

- **content** – Inhalt (Text, Bild)
- **padding** – Innenabstand
- **border** – Rahmen
- **margin** – Außenabstand

Gesamt-Breite = `width + padding + border + margin`. Tipp:

`box-sizing: border-box` rechnet padding + border mit ein.



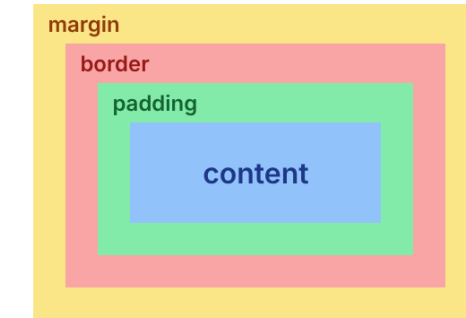
Box-Modell: CSS

```
.box {  
  width: 200px;  
  height: 100px;  
  padding: 20px;  
  border: 2px solid black;  
  margin: 10px;  
}
```

Wichtig:

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

→ width/height inkludiert padding + border.



```
.box {  
  width: 140px;  
  padding: 18px;  
  border: 18px solid;  
  margin: 18px;  
}
```

Farben

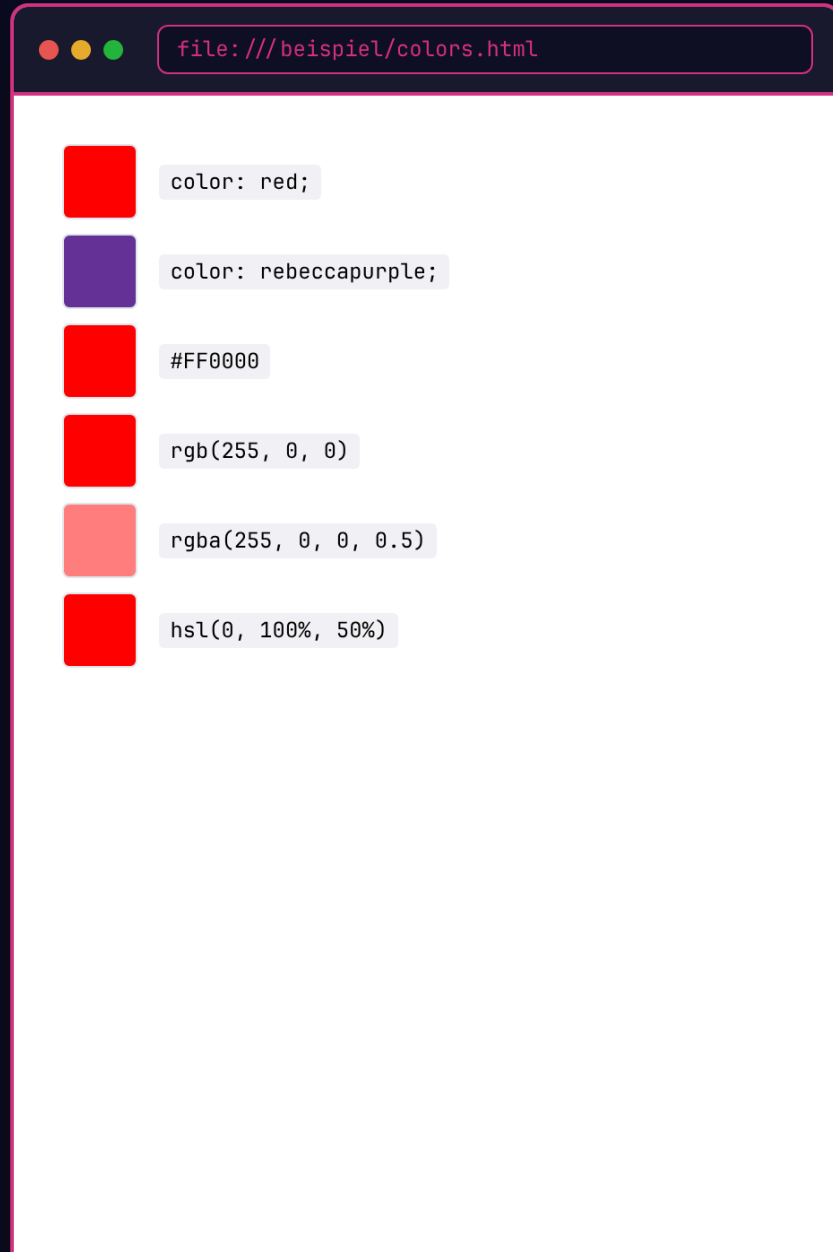
CSS kennt fünf Notationen für Farben: **Keyword**, **Hex**, **RGB**, **RGBA** (mit Transparenz) und **HSL**.

```
/* Keyword */
color: red;
color: rebeccapurple;

/* Hex */
color: #FF0000;
color: #F00;

/* RGB / RGBA */
color: rgb(255, 0, 0);
color: rgba(255, 0, 0, 0.5);

/* HSL */
color: hsl(0, 100%, 50%);
```



Einheiten

Einheit	Bedeutung
px	Pixel (absolut)
%	Prozent vom Elternelement
em	Relativ zur Schriftgröße des Elements
rem	Relativ zur Schriftgröße des Root-Elements
vw / vh	Prozent der Viewport-Breite/-Höhe

Empfehlung:

rem für Schrift, % oder vw/vh für Layout.

Pseudo-Klassen

```
/* Hover */
a:hover { color: red; }

/* Besucher Link */
a:visited { color: purple; }

/* Fokussiertes Element */
input:focus { border-color: blue; }

/* Erstes/n-tes Kind */
li:first-child { font-weight: bold; }
li:nth-child(odd) { background: #eee; }
```

: vor dem Namen = Pseudo-Klasse.

file:///beispiel/pseudo-classes.html

:hover / :visited
[Link mit Hover](#) · [Besucher Link](#)

:focus

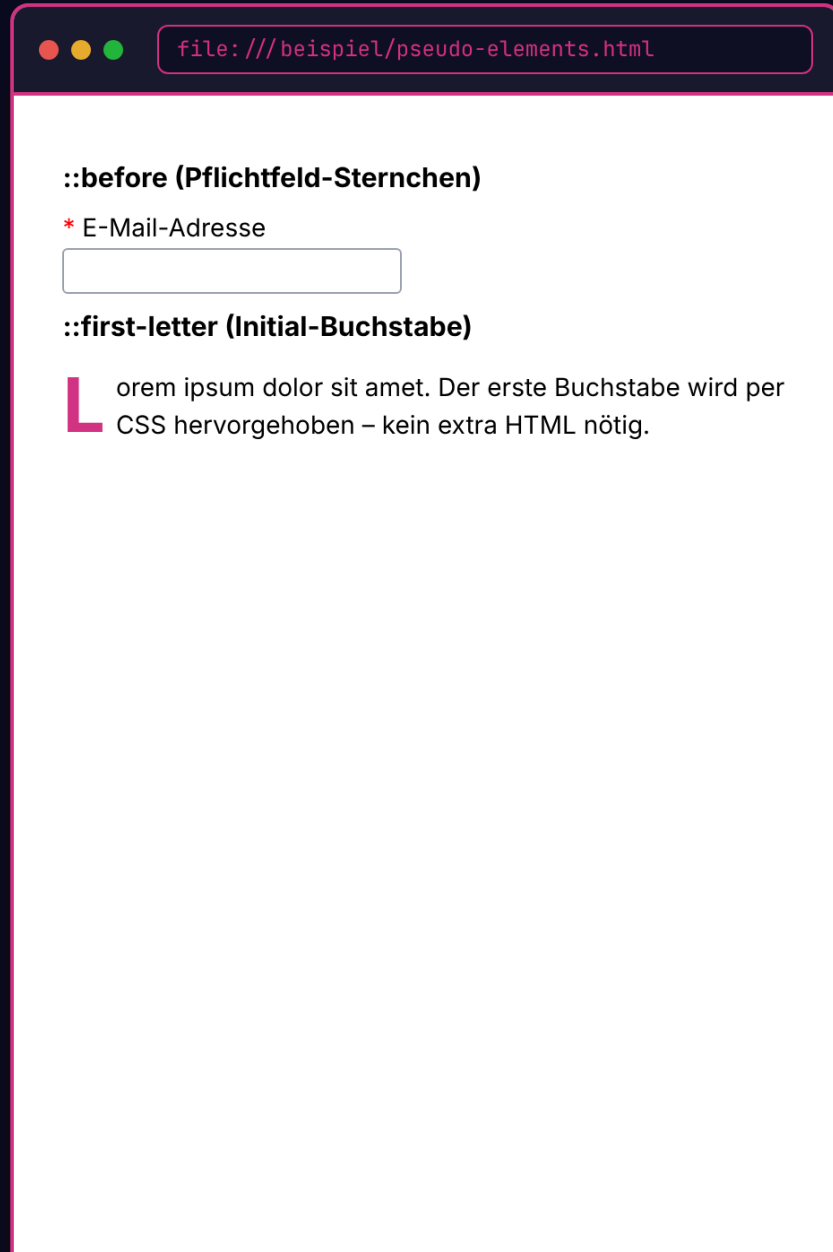
:first-child, :nth-child(odd)

Erster (bold, zebra)
Zweiter
Dritter (zebra)
Vierter
Fünfter (zebra)

Pseudo-Elemente

```
/* Vor dem Inhalt einfügen */  
.required::before {  
  content: "* ";  
  color: red;  
}  
  
/* Erster Buchstabe */  
p::first-letter {  
  font-size: 2em;  
}
```

- ::** = Pseudo-Element (erzeugt "virtuelles" Element)
- :** = Pseudo-Klasse (wählt existierendes Element im Zustand)



Layout: Flexbox

```
.container {  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
  gap: 1rem;  
}
```

→ Eindimensionales Layout (Zeile *oder* Spalte). Siehe nachfolgende Folien.

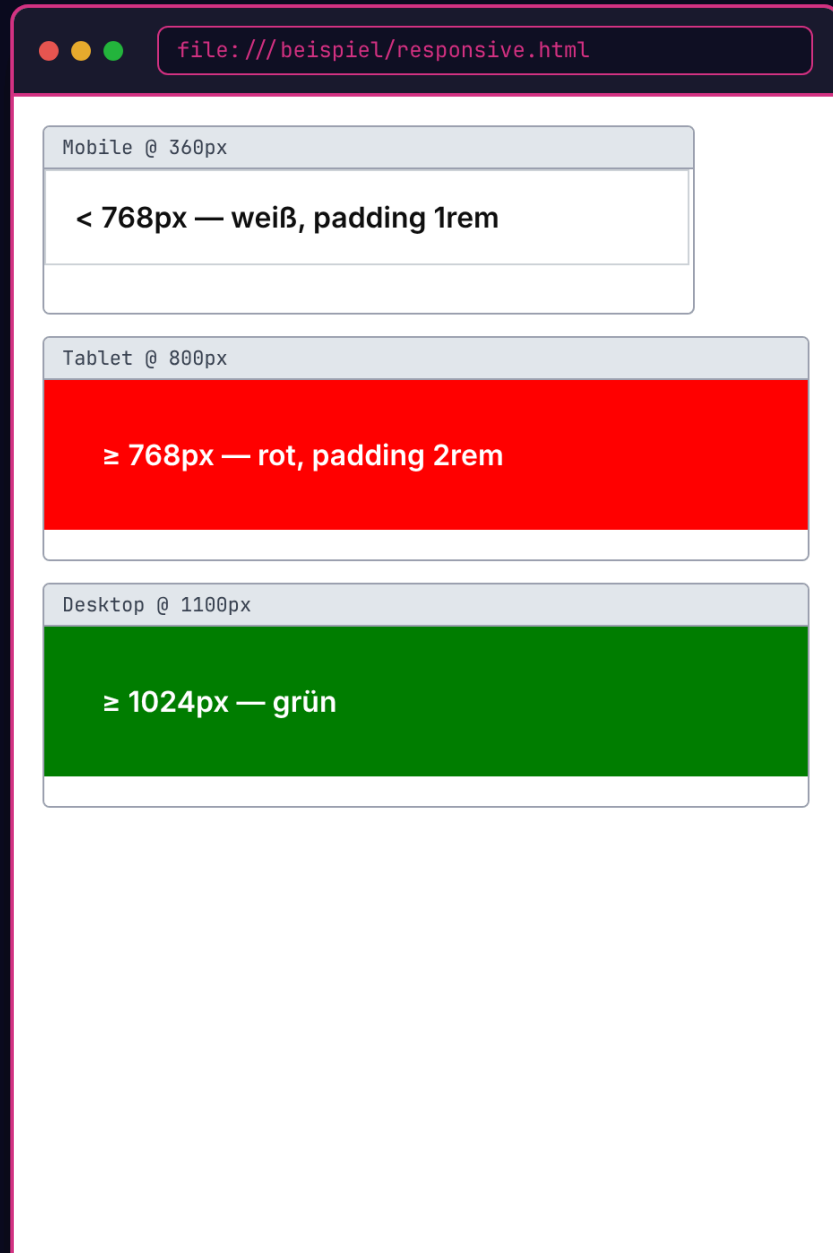
Layout: Grid

```
.grid {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  gap: 20px;  
}
```

→ Zweidimensionales Layout (Zeilen *und* Spalten gleichzeitig).

Responsive Design

```
/* Mobile First: Basis-Styles */  
.container {  
  padding: 1rem;  
}  
  
/* Ab 768px (Tablet) */  
@media (min-width: 768px) {  
  .container { padding: 2rem; }  
}  
  
/* Ab 1024px (Desktop) */  
@media (min-width: 1024px) {  
  .container { padding: 3rem; }  
}
```



Responsive Design – Vertiefung

Mobile First ist Standard: Basis-CSS für kleine Screens, dann Erweiterungen.

Breakpoint	Gerät	Media Query
< 576px	Smartphone (Portrait)	Basis
≥ 576px	Smartphone (Landscape)	@media (min-width: 576px)
≥ 768px	Tablet	@media (min-width: 768px)
≥ 992px	Desktop	@media (min-width: 992px)
≥ 1200px	Large Desktop	@media (min-width: 1200px)

Viewport-Meta-Tag (kritisch!):

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Moderne Alternativen: `clamp()`, Container Queries, CSS Grid `auto-fit` / `auto-fill`.

CSS: Extended

CSS Selectors – Attribut-Selektoren

```
/* presence */
[disabled] { opacity: 0.5; }

/* exact value */
[type="email"] { border-color: blue; }

/* contains word */
[class~="btn"] { cursor: pointer; }

/* starts with */
[href^="https"] { color: green; }

/* ends with */
[src$=".png"] { border: 1px solid #ccc; }

/* contains substring */
[class*="icon"] { padding-left: 20px; }
```

CSS Combinators

```
/* Child (direkt) */  
div > p { margin: 0; }  
  
/* Descendant (Nachkomme) */  
div p { margin: 0; }  
  
/* Next Sibling */  
h1 + p { font-size: 1.2em; }  
  
/* Subsequent Sibling */  
h1 ~ p { color: #666; }  
  
/* Selector List */  
h1, h2, h3 { font-weight: bold; }
```

Pseudo Classes – Erweitert

```
/* Form States */
:valid { border-color: green; }
:invalid { border-color: red; }
:placeholder-shown { color: #999; }

/* Content */
:before { content: "→ "; }
:after { content: " [Wichtig]"; }
:first-letter { font-size: 2em; }
:first-line { font-weight: bold; }

/* NOT */
:not(.disabled) { pointer-events: auto; }
:not(:last-child) { border-bottom: 1px solid #ccc; }
```

Shorthand Properties

```
/* margin: top right bottom left */  
margin: 10px 20px 10px 20px;  
  
/* centering */  
margin: 0 auto;  
  
/* padding same */  
padding: 20px;  
  
/* border: width style color */  
border: 1px solid #333;  
  
/* background: color image position/size repeat */  
background: #fff url(logo.png) 0 0 no-repeat;  
  
/* font: style variant weight size/line-height family */  
font: italic normal bold 16px/1.5 Arial, sans-serif;
```

Flexbox Shorthands

```
/* flex: grow shrink basis */  
flex: 1 0 auto;  
  
/* place-items: align-items justify-items */  
place-items: center stretch;  
  
/* place-content: align-content justify-content */  
place-content: space-between center;  
  
/* gap: row-gap column-gap */  
gap: 20px 40px;
```

CSS Custom Properties (Variables)

```
:root {  
  --primary-color: #d63384;  
  --spacing: 20px;  
  --radius: 8px;  
}  
  
.button {  
  background: var(--primary-color);  
  padding: var(--spacing);  
  border-radius: var(--radius);  
}
```

CSS Functions

```
/* calc */  
width: calc(100% - 40px);  
  
/* min/max/clamp */  
width: min(100%, 600px);  
font-size: clamp(14px, 2vw, 20px);  
  
/* var with fallback */  
color: var(--text-color, #333);  
  
/* rgb/rgba/hsl/hsla */  
color: hsl(200, 100%, 50%);  
background: rgba(0, 0, 0, 0.5);
```

Transition & Animation

```
/* transition: property duration timing-function */
transition: all 0.3s ease-in-out;
transition: background 0.2s, transform 0.1s;

/* animation */
@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}

.fade-in {
  animation: fadeIn 0.5s ease-out;
}
```

CSS Frameworks

Framework	Stil
Bootstrap	Komponenten + Grid, opinionated
Tailwind CSS	Utility-First, atomar
Bulma	klassisch, ohne JS
shadcn/ui	kopierbare Komponenten (Tailwind-basiert)
Flowbite	Tailwind-Komponentenbib.

Tailwind CSS – Beispiel

```
<button class="bg-blue-600 hover:bg-blue-700  
            text-white font-semibold  
            px-4 py-2 rounded-lg  
            transition shadow-sm">  
  Klick mich  
</button>
```

- Keine eigenen CSS-Klassen mehr
- JIT-Compiler erzeugt nur genutzte Klassen
- Design-System via `tailwind.config.js`