Spickzettel HTML & CSS · Webseiten erstellen

Grundlagen · Text · Links · Semantik · Bilder · Hilfe

Holger Sebastião · www.sebastiao.org
Version 1.1 · 2026-10-25

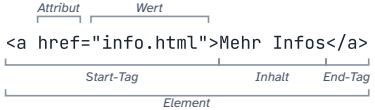
Grundgerüst — index.html

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content=
    "width=device-width, initial-scale=1">
  <title>Meine Seite</title>
  <link rel="stylesheet"
        href="css/style.css">
</head>
<body>
  <h1>Hallo Welt</h1>
</body>
</html>
```

`<!DOCTYPE html>`	Das ist HTML5. Immer die erste Zeile.
`<head>`	Infos für den Browser, unsichtbar: Zeichensatz, Titel im Tab, CSS.
`<body>`	Alles, was man auf der Seite sieht.

Die Startseite heißt `index.html`.

Aufbau eines Elements



Der End-Tag hat einen `/`. Attribute stehen im Start-Tag: `name="wert"`, Wert in Anführungszeichen.

Leere Elemente haben keinen Inhalt und keinen End-Tag: `<br>` `<hr>` `<img>` `<meta>` `<link>`

Richtig verschachteln, zuerst innen schließen: `<p><strong>…</strong></p>`, nicht `<p><strong>…</p></strong>`

Tags klein schreiben und mit Einrückung zeigen, was ineinander steckt.

Sonderzeichen

`<`	<	`>`	>
`&`	&	` `	Leerzeichen ohne Umbruch
`©`	©		

Umlaute wie ä, ö, ü, ß direkt schreiben, dank `charset="UTF-8"`.

Text

```
<h1>Hauptüberschrift</h1>
<h2>Unterüberschrift</h2>   <!-- bis h6 -->

<p>Ein Absatz. Hier steht Text,
der im Browser einfach weiterläuft.</p>

<p>Erste Zeile<br>
zweite Zeile im selben Absatz</p>

<strong>wichtig</strong>  <em>betont</em>
<hr>   <!-- Trennlinie -->
```

`h1` … `h6`	Überschriften. `h1` nur einmal pro Seite, keine Stufe überspringen.
`p`	Absatz (paragraph)
`br`	Zeilenumbruch innerhalb eines Absatzes
`strong`, `em`	wichtig (fett), betont (kursiv)

Leerzeichen und Zeilenumbrüche im Code zeigt der Browser nicht. `<br>` nicht für Abstände benutzen, dafür gibt es CSS.

Kommentar: `<!-- wird nicht angezeigt -->`

Listen

```
<ul>
  <li>Milch</li>
  <li>Brot</li>
</ul>
```

```
<ol>
  <li>Wecker</li>
  <li>Duschen</li>
</ol>
```

`ul`	Liste mit Punkten (unordered list)
`ol`	nummerierte Liste (ordered list)
`li`	ein Listenpunkt (list item), nur in `ul` oder `ol`

Verschachteln: Die innere Liste steht **im** `<li>` der äußeren.

Tabelle

```
<table>
  <tr><th>Name</th><th>Note</th></tr>
  <tr><td>Ada</td><td>1</td></tr>
  <tr><td>Max</td><td>2</td></tr>
</table>
```

`tr` Zeile (table row), `th` Überschriftzelle, `td` Datenzelle. Tabellen nur für Daten, nie für das Seitenlayout.

Links — a href

```
<a href="kontakt.html">Kontakt</a>
```

ZIEL	HREF
Seite im selben Ordner	`kontakt.html`
Seite im Unterordner	`seiten/team.html`
eine Ebene höher	`../index.html`
andere Website	`https://visucode.de`
Stelle auf der Seite	`#kontakt`
E-Mail	`mailto:info@schule.de`

```
<!-- in neuem Tab öffnen -->
<a href="https://de.wikipedia.org"
   target="_blank">Wikipedia</a>

<!-- Sprungziel: id ohne # -->
<h2 id="kontakt">Kontakt</h2>

<!-- Bild als Link -->
<a href="index.html"><img src="logo.png"
   alt="Zur Startseite"></a>
```

Relativer Pfad (`seiten/team.html`): vom Ordner der aktuellen Datei aus, funktioniert auch auf dem Server. **Absoluter Pfad** beginnt mit `https://`.

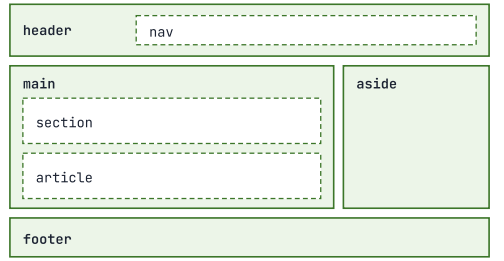
Nie `C:\Users\…` oder `/Users/…` als Ziel: Das gibt es nur auf deinem Rechner.

Navigation — nav

```
<header>
  <nav>
    <ul>
      <li><a href="index.html">Start</a></li>
      <li><a href="team.html">Team</a></li>
      <li><a href="info.html">Info</a></li>
    </ul>
  </nav>
</header>
```

Ein Menü ist eine **Liste von Links** in `<nav>`. Dasselbe Menü steht auf jeder Seite. Die Leiste daraus macht CSS (Seite 2).

Semantisches HTML5 — Seitenaufbau



`header`	Kopf: Logo, Titel, Menü
`nav`	Hauptnavigation
`main`	Hauptinhalt, nur einmal pro Seite
`section`	Abschnitt mit eigener Überschrift
`article`	steht für sich allein, z. B. ein Beitrag
`aside`	Nebeninfo, z. B. Kasten am Rand
`footer`	Fuß: Impressum, Kontakt

Semantisch heißt: Das Tag sagt, **was** der Inhalt ist, nicht wie er aussieht. Das hilft Screenreadern und Suchmaschinen. `div` und `span` nur, wenn kein Tag passt (z. B. nur für CSS).

Bilder — img

```
<figure>
  <img src="bilder/hund.jpg"
       alt="Brauner Hund im Gras"
       width="300">
  <figcaption>Unser Schulhund</figcaption>
</figure>
```

`alt` ist Pflicht: beschreibt das Bild für Blinde und wenn es nicht lädt. `figure` + `figcaption` = Bild mit Bildunterschrift.

Typische Fehler

- End-Tag vergessen oder falsch verschachtelt
- Dateinamen mit Leerzeichen, Umlauten oder Großbuchstaben: besser `ueber-uns.html`. Der Server unterscheidet `Bild.JPG` und `bild.jpg`.
- Pfad falsch: Liegt die Datei wirklich in dem Ordner?
- Prüfen: `validator.w3.org`, im Browser F12 bzw. Rechtsklick → Untersuchen.

Spickzettel HTML & CSS · Webseiten erstellen

Holger Sebastião · www.sebastiao.org
Version 1.1 · 2026-10-25

CSS einbinden

3 Wege

1 · EXTERN: SO MACHEN WIR ES

```
<!-- im head jeder HTML-Seite -->
<link rel="stylesheet" href="css/style.css">
```

```
/* Datei css/style.css: nur CSS, kein HTML */
body {
  font-family: Arial, sans-serif;
}
```

Eine Datei für alle Seiten: Änderst du sie, ändern sich alle Seiten.

2 · INTERN IM HEAD

```
<head>
  <style>
    h1 { color: navy; }
  </style>
</head>
```

Gilt nur für diese eine Seite.

3 · DIREKT IM TAG (INLINE)

```
<p style="color: red;">Achtung</p>
```

Gilt nur für dieses eine Element. Nur zum schnellen Testen.

> **Wer gewinnt?** Inline vor intern und extern. Sonst gewinnt, was später kommt.

Aufbau einer Regel



Der **Selektor** wählt aus, **was** gestaltet wird. In `{ }` stehen Deklarationen, jede endet mit `;`. Kommentar: `/* … */`

Vererbung in CSS

```
body {
  font-family: Arial, sans-serif;
  color: #1d2433;
}
```

Schrift und Textfarbe von `body` gelten für alles darin. Abstände, Rahmen und Hintergrund werden **nicht** vererbt.

Selektoren

CSS	TRIFFT	IM HTML
`p`	alle Absätze	`<p>`
`.hinweis`	Klasse, beliebig oft	`class="hinweis"`
`#kopf`	id, nur einmal pro Seite	`id="kopf"`
`nav a`	Links **in** nav	
`h1, h2`	h1 und h2	
`a:hover`	Link unter der Maus	
`*`	alle Elemente	

```
<p class="hinweis">Morgen kein Unterricht</p>
```

```
.hinweis {
  background-color: #fff3c4;
}
```

Wenn mehrere Regeln passen: id schlägt Klasse, Klasse schlägt Element. Bei Gleichstand gewinnt die spätere Regel.

Farben & Text

```
h1 {
  color: #1f5fae;       /* Textfarbe */
  background-color: white;
  font-family: Arial, sans-serif;
  font-size: 2rem;
  font-weight: bold;
  text-align: center;   /* oder left */
  line-height: 1.5;     /* Zeilenabstand */
}
a {
  text-decoration: none; /* kein Strich */
}
```

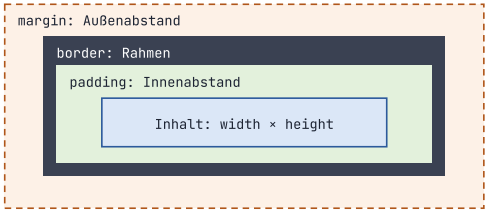
Farbe	`red` · `#ff0000` · `rgb(255, 0, 0)`
durchsichtig	`rgba(0, 0, 0, 0.5)`
Einheiten	`px` fest · `%` vom Elternteil · `rem` vom Grundtext (16px)

Links gestalten

```
a:link    { color: #1f5fae; } /* neu */
a:visited { color: #6b4aa6; } /* besucht */
a:hover   { color: #b0561a; } /* Maus */
a:active  { color: #b3261e; } /* Klick */
```

Reihenfolge einhalten, sonst überschreiben sie sich. Merkhilfe: **LoVe HAte**.

Box-Modell




```
.karte {
  width: 300px;
  padding: 1rem;
  border: 2px solid #3a4152;
  border-radius: 8px;  /* runde Ecken */
  margin: 20px auto;   /* auto = mittig */
  box-sizing: border-box;
}
```

`box-sizing: border-box`: `width` zählt padding und border mit. Kurzform `margin: 10px 20px;` = oben/unten 10, links/rechts 20. Vier Werte: oben, rechts, unten, links (im Uhrzeigersinn).

Navigation als Leiste

Flexbox

```
nav ul {
  list-style: none;   /* keine Punkte */
  margin: 0;
  padding: 0;
  display: flex;      /* nebeneinander */
  gap: 1rem;
}
nav a {
  color: white;
  text-decoration: none;
  padding: 0.5rem 1rem;
}
nav a:hover {
  background-color: #1f5fae;
}
```

`display: flex` stellt die Kinder nebeneinander. Dazu: `justify-content: center` (waagerecht), `align-items: center` (senkrecht), `flex-direction: column` (untereinander).

Handy & Tablet

@media

```
/* gilt nur bis 600px Breite */
@media (max-width: 600px) {
  nav ul {
    flex-direction: column;
  }
}
```

Braucht im HTML-head die viewport-Zeile aus dem Grundgerüst.

Typische Fehler

- `;` oder `}` vergessen: Ab da wirkt nichts mehr.
- Im CSS `.hinweis` mit Punkt, im HTML `class="hinweis"` ohne.
- Pfad im `<link>` falsch: liegt `style.css` im Ordner `css`?
- Alte Version im Browser: neu laden mit Strg+F5 bzw. Cmd+Shift+R.
- Nachsehen, welche Regel wirkt: F12 bzw. Rechtsklick → Untersuchen.

Ordner einer Website

```
website/
├── index.html
├── team.html
├── kontakt.html
├── css/
│   └── style.css
└── bilder/
    └── logo.png
```

Von `index.html` aus: `css/style.css`, `bilder/logo.png`. Alles klein, ohne Leerzeichen.

Seitenlayout

Flexbox

```
<div class="inhalt">
  <main>…</main>
  <aside>…</aside>
</div>
```

```
.inhalt {
  display: flex;
  gap: 2rem;
}
main  { flex: 3; }   /* 3/4 der Breite */
aside { flex: 1; }   /* 1/4 der Breite */
```


Nachschlagen: developer.mozilla.org/de/docs/Web/CSS

Spickzettel HTML & CSS · Webseiten erstellen

Aufbau · Felder · Attribute · Auswahl · CSS · Hilfe

Holger Sebastião · www.sebastiao.org
Version 1.1 · 2026-10-25

Aufbau eines Formulars — form

```
<form action="senden.php" method="post">
  <label for="vorname">Vorname</label>
  <input type="text" id="vorname"
         name="vorname" required>

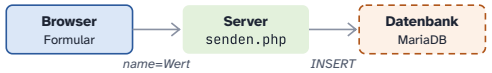
  <button type="submit">Senden</button>
</form>
```

`action`	wohin die Daten gehen (Datei auf dem Server)
`method`	`get`: Daten stehen in der Adresszeile · `post`: unsichtbar, für Passwörter und viele Daten
`name`	unter diesem Namen kommt der Wert beim Server an
`id`	verbindet das Feld mit seinem `label`

label for = id: Klick auf den Text setzt den Cursor ins Feld. Vorleseprogramme lesen den Text vor.

Ohne `name` wird das Feld **nicht** mitgeschickt.

Was passiert beim Senden?



HTML zeigt nur das Formular. Speichern kann erst ein Programm auf dem Server, z. B. in PHP mit SQL.

Knöpfe — button

```
<button type="submit">Senden</button>
<button type="reset">Leeren</button>
<button type="button">Nur Knopf</button>
```

`submit` schickt ab, `reset` leert alle Felder, `button` tut nichts (für JavaScript). Immer `type` angeben.

Ohne Server testen

```
<form action="danke.html" method="get">
```

Eine leere Seite `danke.html` anlegen. Nach dem Senden steht in der Adresszeile:
`danke.html?vorname=Ali`
So siehst du, was das Formular schickt.

Eingabefelder — input

TYPE	FÜR
`text`	kurzer Text, eine Zeile
`email`	E-Mail, prüft auf @
`password`	zeigt nur Punkte
`number`	Zahl, mit `min` `max` `step`
`date`	Datum mit Kalender
`tel`	Telefonnummer
`url`	Internetadresse, prüft auf https://
`color` · `range`	Farbwähler · Schieberegler
`file`	Datei hochladen
`checkbox`	Häkchen, mehrere möglich
`radio`	genau eins aus einer Gruppe
`submit`	Senden-Knopf (besser `<button>`)

```
<label for="alter">Alter</label>
<input type="number" id="alter"
       name="alter" min="14" max="99">

<label for="mail">E-Mail</label>
<input type="email" id="mail" name="mail"
       placeholder="name@schule.de">
```

Wichtige Attribute

`required`	Pflichtfeld, sonst kein Senden
`placeholder`	grauer Beispieltext im leeren Feld
`value`	Wert, der geschickt wird (oder Vorgabe)
`maxlength="20"`	höchstens 20 Zeichen
`min` · `max`	kleinste und größte Zahl bzw. Datum
`checked`	Häkchen oder Punkt schon gesetzt
`disabled`	ausgegraut, wird nicht geschickt
`readonly`	nur lesen, wird geschickt
`autofocus`	Cursor steht beim Laden im Feld

placeholder ersetzt kein label: Der Text verschwindet beim Tippen.

Auswählen — radio · checkbox

```
<fieldset>
  <legend>Klasse</legend>
  <input type="radio" id="k11"
         name="klasse" value="11" checked>
  <label for="k11">11</label>
  <input type="radio" id="k12"
         name="klasse" value="12">
  <label for="k12">12</label>
</fieldset>

<input type="checkbox" id="agb"
       name="agb" required>
<label for="agb">Ich stimme zu</label>
```

radio: gleicher `name` = eine Gruppe, nur eins wählbar. Jedes braucht ein eigenes `value`.
`fieldset` rahmt zusammengehörige Felder ein, `legend` ist die Überschrift.

Liste und langer Text

```
<label for="fach">Fach</label>
<select id="fach" name="fach">
  <option value="">Bitte wählen</option>
  <option value="inf">Informatik</option>
  <option value="ma">Mathe</option>
</select>

<label for="text">Nachricht</label>
<textarea id="text" name="text"
          rows="4"></textarea>
```

`textarea` hat ein End-Tag und kein `value`: Vorgabetext steht zwischen den Tags.

Was kommt beim Server an?

FELD	GESCHICKT WIRD
Textfeld	`vorname=Ali`
radio	`klasse=12` (das gewählte `value`)
checkbox an	`agb=on` (ohne `value`)
checkbox aus	gar nichts
select	`fach=inf` (`value`, nicht der Text)

Bei `get` stehen die Paare mit `&` in der Adresszeile:
`?vorname=Ali&klasse=12`

Formulare gestalten — CSS

```
label {
  display: block;      /* eigene Zeile */
  margin-top: 0.8rem;
}
input, select, textarea {
  width: 100%;
  padding: 0.4rem;
  border: 1px solid #8a92a3;
  border-radius: 4px;
  box-sizing: border-box;
}
input:focus, textarea:focus {
  outline: 2px solid #1f5fae; /* aktiv */
}
input:invalid {
  border-color: #b3261e; /* Fehler */
}
button {
  margin-top: 1rem;
  padding: 0.5rem 1.5rem;
}
```

Bei radio und checkbox `width: auto` lassen und `label` mit `display: inline` daneben setzen.

```
input[type="radio"],
input[type="checkbox"] { width: auto; }
```

label ohne for

```
<label>
  <input type="checkbox" name="news">
  Newsletter bestellen
</label>
```

Steht das Feld **im** `label`, braucht es kein `for` und keine `id`. Praktisch bei checkbox und radio.

Typische Fehler

- `name` fehlt: Wert kommt nicht an.
- `for` und `id` unterschiedlich geschrieben.
- Radio-Knöpfe mit verschiedenem `name`: alle gleichzeitig wählbar.
- `<input>` hat kein End-Tag, `<textarea>` und `<select>` schon.
- Knopf außerhalb von `<form>`: sendet nichts.
- `required` ersetzt keine Prüfung auf dem Server.