Lernheft Web

HTML · CSS · JavaScript
einfach erklärt, Schritt für Schritt

Für den Unterricht Web & Programmierung

Dieses Heft begleitet dich beim Lernen. Es ist so aufgebaut, wie ihr im Unterricht vorgeht: Erst die **Struktur** der Seite mit HTML, danach die **Gestaltung** mit CSS und zuletzt die **Interaktion** mit JavaScript.

HTML	**CSS**	**JavaScript**
Das Skelett (Struktur)	Die Kleidung (Aussehen)	Die Muskeln (Verhalten)

Inhalt

- **Teil 1:** HTML (Grundgerüst, Text, Listen, Links, Bilder, Tabellen, Formulare, Struktur)
- **Teil 2:** CSS (Selektoren, Farben, Box-Modell, Layout)
- **Teil 3:** JavaScript (Variablen, Funktionen, Ereignisse, DOM)
- **Teil 4:** Dein Lernplan mit eigener HTML-Datei, typische Fehler, Übungsaufgaben

> **Tipp:** Schreibe beim Lernen **jeden Befehl selbst ab** und öffne die Datei im Browser. Nur Tippen und Ausprobieren bringt Routine. Dafür gibt es die fertige Beispieldatei **lernseite.html** und das Übersichtsbild **html_uebersicht.png**.

Teil 1: HTML

1.1 Was ist HTML?

HTML heißt **HyperText Markup Language**. Es ist keine Programmiersprache, sondern eine **Auszeichnungssprache**: Du sagst dem Browser mit **Tags**, was ein Inhalt ist (Überschrift, Absatz, Bild ...). Der Browser zeigt das dann an ("rendert" es). Eine HTML-Datei ist eine **reine Textdatei** mit der Endung `.html`. Speichere sie im Editor immer mit **UTF-8**.

1.2 Aufbau eines Tags

```
<p class="info">Das ist ein Absatz.</p>
 |  |__________|  |_______________|  |
 |   Attribut       Inhalt          Ende-Tag
 Start-Tag
```

- Die meisten Elemente haben ein **Start-Tag** `<p>` und ein **Ende-Tag** `</p>` (mit Schrägstrich).
- **Attribute** stehen im Start-Tag und haben die Form `name="wert"`.
- Einige Tags haben **keinen Inhalt** und kein Ende-Tag, zum Beispiel `<br>`, `<hr>`, `<img>`, `<input>`.
- Tags können **verschachtelt** sein. Immer in umgekehrter Reihenfolge schließen: `<p><b>Text</b></p>`.

1.3 Das Grundgerüst jeder Webseite

grundgeruest.html

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Meine erste Seite</title>
</head>
<body>
  <h1>Hallo Welt</h1>
  <p>Mein erster Absatz.</p>
</body>
</html>
```

Baustein	Aufgabe
`<!DOCTYPE html>`	Sagt dem Browser: Das ist modernes HTML5.
`<html lang="de">`	Wurzel der Seite. **lang** nennt die Sprache (de = Deutsch).
`<head>`	Unsichtbare Zusatzinfos: Titel, Zeichensatz, CSS-Einbindung.
`<meta charset="UTF-8">`	Zeichensatz. Ohne ihn erscheint "Ã¤" statt "ä" (siehe Einführung).
`<title>`	Text im Browser-Tab.
`<body>`	Alles, was auf der Seite **sichtbar** ist.

1.4 Texte

Tag	Bedeutung
`<h1> bis <h6>`	Überschriften. **h1** ist die wichtigste (nur einmal pro Seite), **h6** die kleinste.
`<p>`	Absatz. Der Browser setzt davor und danach Abstand.
` `	Zeilenumbruch innerhalb eines Absatzes.
`<hr>`	Waagerechte Trennlinie.
`<strong>`	Wichtiger Text, wird **fett** dargestellt.
`<em>`	Betonter Text, wird *kursiv* dargestellt.
`<mark>`	Hervorgehobener Text (gelb markiert).
`<small>`	Kleingedrucktes.
`<sub> / <sup>`	Tiefgestellt (H_2O) und hochgestellt (x^2).
`<pre>`	Text so anzeigen, wie er getippt wurde (Leerzeichen und Umbrüche bleiben).
`<code>`	Kennzeichnet Programmcode.
`<blockquote>`	Längeres Zitat.

> **Merke:** HTML ignoriert mehrfache Leerzeichen und Zeilenumbrüche im Quelltext. Zehn Leerzeichen sehen aus wie eins. Für Umbrüche brauchst du `<br>` oder einen neuen `<p>`.

1.5 Listen

Ungeordnete Liste (Punkte) links, geordnete Liste (Zahlen) rechts

```
<ul>                    <ol>
  <li>Apfel</li>         <li>Erst das</li>
  <li>Birne</li>         <li>Dann das</li>
</ul>                   </ol>
```

Tag	Bedeutung
`<ul>`	Unordered List: Liste mit Aufzählungspunkten.
`<ol>`	Ordered List: nummerierte Liste.
`<li>`	List Item: ein Eintrag. Steht **immer** in **ul** oder **ol**.
`<dl> <dt> <dd>`	Beschreibungsliste: Begriff (dt) und Erklärung (dd).

1.6 Links und Bilder

```
<a href="https://www.bundeswehr.de" target="_blank">Zur Seite</a>
<a href="seite2.html">Interner Link</a>
<a href="#unten">Springe zum Anker</a>
<a href="mailto:name@beispiel.de">E-Mail schreiben</a>

<img src="html_uebersicht.png" alt="Übersicht der HTML-Tags" width="400">
```

Tag	Erklärung
`<a href="...">`	Link (Anchor). Das Ziel steht in **href**. Mit **target="_blank"** öffnet er sich in einem neuen Tab.

Tag	Erklärung
`<img src="...">`	Bild einbinden. **src** = Dateiname/Pfad, **alt** = Ersatztext (für Blinde und bei Ladefehlern), **width** = Breite in Pixeln.

Wichtig: Pfade: Liegt das Bild im selben Ordner wie die HTML-Datei, reicht der Dateiname. Liegt es im Unterordner `bilder`, schreibst du `bilder/foto.png`. Ein Ordner höher ist `../foto.png`.

1.7 Tabellen

```
<table border="1">
  <tr>
    <th>Name</th> <th>Fach</th>
  </tr>
  <tr>
    <td>Anna</td> <td>HTML</td>
  </tr>
</table>
```

Tag	Bedeutung
`<table>`	Die ganze Tabelle.
`<tr>`	Table Row: eine Zeile.
`<th>`	Table Header: Kopfzelle (fett, zentriert).
`<td>`	Table Data: normale Zelle.
`<thead>` `<tbody>` `<tfoot>`	Kopf, Körper und Fuß der Tabelle (gliedert sie).
`colspan` / `rowspan`	Attribute: Zelle über mehrere Spalten oder Zeilen verbinden.

1.8 Formulare

```
<form action="/senden" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" placeholder="Dein Name">

  <label for="mail">E-Mail:</label>
  <input type="email" id="mail" name="mail" required>

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

  <select name="fach">
    <option>HTML</option>
    <option>CSS</option>
  </select>

  <textarea name="text" rows="4"></textarea>
  <button type="submit">Absenden</button>
</form>
```

Element	Erklärung
`<form>`	Umschließt das Formular. **action** = Ziel, **method** = get oder post.
`<label>`	Beschriftung. Mit **for** wird sie über die **id** mit dem Feld verbunden.

Element	Erklärung
`<input type="...">`	Eingabefeld. Typen: text, password, email, number, date, checkbox, radio, file, submit.
`<select> <option>`	Auswahlliste (Dropdown).
`<textarea>`	Mehrzeiliges Textfeld.
`<button>`	Schaltfläche.
`placeholder / required`	Hinweistext im Feld / Pflichtfeld.

1.9 Struktur und Gliederung der Seite

Früher wurde alles in `<div>` verpackt. Heute nutzt man **semantische Tags**, die ihren Zweck im Namen tragen. Das hilft Suchmaschinen und Screenreadern.

Tag	Bedeutung
`<header>`	Kopfbereich der Seite (Logo, Titel).
`<nav>`	Navigation, also die Menüleiste mit Links.
`<main>`	Hauptinhalt der Seite (nur einmal).
`<section>`	Thematischer Abschnitt.
`<article>`	Eigenständiger Beitrag.
`<aside>`	Nebeninhalt, zum Beispiel eine Seitenleiste.
`<footer>`	Fußbereich (Impressum, Kontakt).
`<div>`	Neutraler Block-Container ohne Bedeutung, zum Gruppieren.
`<span>`	Neutraler Inline-Container für einen Teil eines Textes.

Skizze einer typischen Seite

```
<body>
  <header> <h1>Meine Seite</h1> <nav>...</nav> </header>
  <main>
    <section> <h2>Thema 1</h2> <p>...</p> </section>
    <aside>Nebeninfo</aside>
  </main>
  <footer>&copy; 2026</footer>
</body>
```

1.10 Block- und Inline-Elemente

- **Block-Elemente** beginnen in einer neuen Zeile und nehmen die volle Breite ein: `h1`, `p`, `div`, `ul`, `table`.
- **Inline-Elemente** fließen im Text mit und sind nur so breit wie ihr Inhalt: `a`, `strong`, `em`, `span`, `img`.

1.11 Kommentare, Sonderzeichen, globale Attribute

```
<!-- Das ist ein Kommentar, der Browser zeigt ihn nicht an -->

&lt;  &gt;  &amp;  &copy;  &euro;   
```

Entity	Ergebnis
`< >`	Zeichen < und > (damit der Browser sie nicht als Tag liest).
`&`	Das &-Zeichen.
` `	Geschütztes Leerzeichen (kein Umbruch).
`© €`	Copyright- und Euro-Zeichen.

Attribut	Wirkung
`id="x"`	Eindeutiger Name eines Elements (nur **einmal** pro Seite).
`class="x"`	Gruppenname, den **mehrere** Elemente tragen dürfen. Wichtig für CSS.
`style="..."`	CSS direkt am Element (nur für Tests, besser eine eigene CSS-Datei).
`title="..."`	Tooltip beim Daraufzeigen mit der Maus.

Teil 2: CSS

2.1 Was ist CSS?

CSS heißt **Cascading Style Sheets**. Es bestimmt das **Aussehen**: Farben, Schriften, Abstände, Anordnung. HTML sagt *was* etwas ist, CSS sagt *wie* es aussieht.

2.2 Drei Wege, CSS einzubinden

```
<!-- 1. Extern (empfohlen): eigene Datei style.css -->
<head> <link rel="stylesheet" href="style.css"> </head>

<!-- 2. Intern: im head der Seite -->
<style> p { color: blue; } </style>

<!-- 3. Inline: direkt am Element (nur für Tests) -->
<p style="color: blue;">Blauer Text</p>
```

2.3 Aufbau einer CSS-Regel

```
selektor {
  eigenschaft: wert;
  eigenschaft: wert;
}

h1 {
  color: darkblue;       /* Textfarbe  */
  font-size: 32px;       /* Schriftgröße */
}
```

- Der **Selektor** wählt aus, welche Elemente gestaltet werden.
- In den **geschweiften Klammern** stehen Paare aus Eigenschaft und Wert. Jedes Paar endet mit einem **Semikolon**.
- Kommentare in CSS: `/* Text */`.

2.4 Selektoren

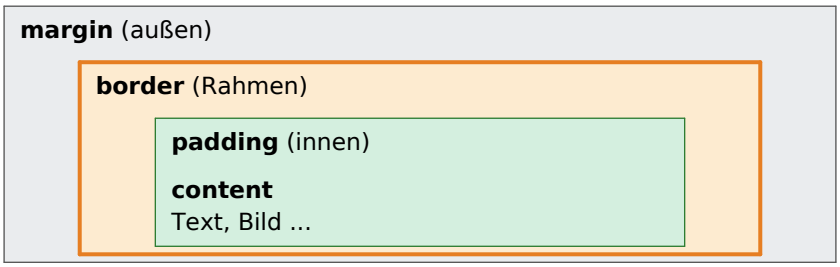
Selektor	Wählt aus
`p`	**Tag-Selektor**: alle Absätze.
`.info`	**Klassen-Selektor**: alle Elemente mit **class="info"**.
`#kopf`	**ID-Selektor**: das eine Element mit **id="kopf"**.
`nav a`	**Nachfahre**: alle Links innerhalb von nav.
`h1, h2`	**Gruppe**: h1 und h2 bekommen dieselbe Regel.
`a:hover`	**Pseudoklasse**: Link, wenn die Maus darüber ist.
`*`	Alle Elemente.

2.5 Die wichtigsten Eigenschaften

Eigenschaft	Wirkung
`color`	Textfarbe. Namen (red), Hex (#1f4e79) oder rgb(31,78,121).
`background-color`	Hintergrundfarbe.
`font-family`	Schriftart, zum Beispiel Arial, sans-serif.
`font-size`	Schriftgröße in px, em oder rem.
`font-weight`	Fettdruck: normal oder bold.
`text-align`	Ausrichtung: left, center, right, justify.
`text-decoration`	Unterstreichung: none oder underline.
`width / height`	Breite und Höhe.
`margin`	**Außenabstand** um das Element.
`padding`	**Innenabstand** zwischen Inhalt und Rand.
`border`	Rahmen, zum Beispiel 2px solid black.
`border-radius`	Abgerundete Ecken.
`display`	Darstellungsart: block, inline, flex, none (versteckt).

2.6 Das Box-Modell

Jedes HTML-Element ist für CSS eine **Box** aus vier Schichten, von innen nach außen:




```
div {
  width: 300px;
  padding: 20px;                /* innen  */
  border: 2px solid black;      /* Rahmen */
  margin: 10px auto;            /* außen; auto = mittig */
  box-sizing: border-box;       /* Breite inkl. padding und border */
}
```

2.7 Layout mit Flexbox

Flexbox ordnet Elemente in einer Reihe oder Spalte an. Man setzt `display: flex` auf den **Elternteil**.

```
.menue {
  display: flex;
  justify-content: space-between;  /* Abstand in der Reihe */
  align-items: center;             /* senkrecht mittig */
  gap: 16px;                       /* Abstand zwischen Kindern */
}
```

Mini-Beispiel für den Anfang

```
/* style.css, passend zu lernseite.html */
body   { font-family: Arial, sans-serif; background: #f5f5f5; margin: 0; }
header { background: #1f4e79; color: white; padding: 20px; }
nav a  { color: white; margin-right: 12px; text-decoration: none; }
nav a:hover { text-decoration: underline; }
main   { max-width: 800px; margin: 20px auto; padding: 0 16px; }
```

Merke: CSS-Regeln, die später stehen, überschreiben frühere. Genauere Selektoren gewinnen gegen allgemeinere (ID vor Klasse vor Tag). Das ist die "Kaskade" aus dem Namen.

Teil 3: JavaScript

3.1 Was ist JavaScript?

JavaScript ist eine **Programmiersprache**, die direkt im Browser läuft. Sie macht Seiten **interaktiv**: auf Klicks reagieren, Texte ändern, Eingaben prüfen, rechnen. Der Browser hat dafür eine eigene **JavaScript-Engine**. (Nicht verwechseln mit Java, das ist eine andere Sprache.)

3.2 Einbinden

```
<!-- Extern (empfohlen), am Ende des body -->
<script src="script.js"></script>

<!-- Intern -->
<script>
  console.log("Hallo aus JavaScript");
</script>
```

Tipp: Mit `console.log(...)` schreibst du Ausgaben in die **Browser-Konsole**. Öffnen mit **F12**, Reiter "Konsole". Hier siehst du auch Fehlermeldungen. Dein wichtigstes Werkzeug zum Fehlersuchen.

3.3 Variablen und Datentypen

```
let alter = 25;            // Zahl, veränderbar
const name = "Anna";       // Text (String), nicht veränderbar
let aktiv = true;          // Wahrheitswert (true oder false)
let liste = [1, 2, 3];     // Array (Liste)
alter = alter + 1;         // 26
```

- `let` für Werte, die sich ändern, `const` für feste Werte. Das alte `var` vermeidet man heute.
- Namen sind **case-sensitive**: `Name` und `name` sind zwei verschiedene Variablen.
- Ein Semikolon beendet eine Anweisung. Kommentare: `// eine Zeile` oder `/* mehrere Zeilen */`.

3.4 Operatoren

Operator	Bedeutung
`+ - * / %`	Rechnen. **%** = Rest der Division (7 % 2 ergibt 1).
`== === !==`	Vergleichen. **===** prüft Wert **und** Typ (empfohlen).
`< > <= >=`	Größer/kleiner.
`&& || !`	Und, Oder, Nicht.
`+`	Bei Texten: Zusammenfügen ("Hallo " + name).

3.5 Bedingungen und Schleifen

```
if (alter >= 18) {
  console.log("Volljährig");
} else {
  console.log("Minderjährig");
}

for (let i = 1; i <= 5; i++) {      // zählt von 1 bis 5
  console.log(i);
}

let n = 0;
while (n < 3) { n++; }              // solange die Bedingung stimmt
```

3.6 Funktionen

Eine Funktion ist ein **benannter Codeblock**, den man beliebig oft aufrufen kann.

```
function begruessen(person) {
  return "Hallo " + person + "!";
}

console.log(begruessen("Anna"));   // Hallo Anna!
```

3.7 DOM: Die Seite verändern

Der Browser baut aus dem HTML einen Baum, das **DOM** (Document Object Model). JavaScript kann Elemente darin **suchen** und **verändern**.

Befehl	Wirkung
`document.getElementById("x")`	Findet das Element mit der id **x**.
`document.querySelector(".klasse")`	Findet das erste Element, das zum CSS-Selektor passt.
`element.textContent = "..."`	Ändert den Text.
`element.style.color = "red"`	Ändert CSS direkt.
`element.classList.add("aktiv")`	Fügt eine Klasse hinzu (auch **remove** und **toggle**).

3.8 Ereignisse: auf Klicks reagieren

Dein erstes interaktives Beispiel

```
<button id="knopf">Klick mich</button>
<p id="ausgabe">Noch nichts passiert.</p>

<script>
  const knopf = document.getElementById("knopf");
  const ausgabe = document.getElementById("ausgabe");

  knopf.addEventListener("click", function () {
    ausgabe.textContent = "Du hast geklickt!";
  });
</script>
```

Ereignis	Wann?
`click`	Mausklick.
`input`	Eingabe in ein Feld.
`submit`	Formular wird abgeschickt.
`mouseover`	Maus fährt über das Element.
`keydown`	Taste gedrückt.

Typischer Fehler: Das Skript muss **nach** den Elementen stehen (am Ende des body), sonst findet `getElementById` sie noch nicht und liefert `null`.

Teil 4: Dein Lernplan

4.1 Deine eigene Lerndatei

Dein Plan ist gut: Du schreibst den Unterrichtsstoff in eine **eigene HTML-Datei**. So übst du HTML und hast gleichzeitig dein Nachschlagewerk. Empfohlener Ordneraufbau:

```
lernprojekt/
  lernseite.html        <- deine Lerndatei (HTML)
  html_uebersicht.png   <- das Übersichtsbild
  style.css             <- später: CSS
  script.js             <- später: JavaScript
```

4.2 Vorgehen in kleinen Schritten

- **Schritt 1:** Grundgerüst kopieren, Datei als **lernseite.html** in UTF-8 speichern, im Browser öffnen.
- **Schritt 2:** Pro Unterrichtsthema ein **section** mit `<h2>` anlegen und die Inhalte in eigenen Worten schreiben.
- **Schritt 3:** Neue Tags immer **sofort ausprobieren**, Seite im Browser mit **F5** neu laden.
- **Schritt 4:** Oben ein **Inhaltsverzeichnis** mit internen Links (`<a href="#abschnitt">`) bauen.
- **Schritt 5:** Sobald CSS dran ist, Datei `style.css` anlegen und per `<link>` einbinden.
- **Schritt 6:** Bei JavaScript Beispiele wie den Klick-Knopf einbauen.

4.3 Typische Anfängerfehler

Fehler	Ursache und Lösung
`Ende-Tag vergessen`	Layout "läuft aus dem Ruder". Jedes geöffnete Tag prüfen.
`Falsch verschachtelt`	Richtig: ** Text **, falsch: Text .
`Anführungszeichen fehlen`	Attributwerte immer in "..." setzen.
`Umlaute kaputt`	Datei nicht als UTF-8 gespeichert oder **meta charset** fehlt.
`Bild erscheint nicht`	Pfad oder Dateiname falsch (Groß-/Kleinschreibung, Endung).
`Änderung nicht sichtbar`	Datei nicht gespeichert oder Browser nicht neu geladen.
`CSS wirkt nicht`	Tippfehler im Selektor, Semikolon fehlt oder link-Pfad falsch.
`JS tut nichts`	**F12** öffnen und in der Konsole den Fehler lesen.

4.4 Übungsaufgaben

Aufgabe A (HTML, Einstieg): Baue eine Seite "Über mich" mit Überschrift, zwei Absätzen, einer Liste deiner Hobbys und einem Bild.

Aufgabe B (HTML, Tabelle): Erstelle deinen Stundenplan als Tabelle mit Kopfzeile und mindestens 3 Tagen.

Aufgabe C (HTML, Formular): Ein Kontaktformular mit Name, E-Mail, Auswahlliste und Textfeld.

Aufgabe D (Struktur): Gliedere die Seite mit header, nav, main, section und footer.

Aufgabe E (CSS): Gib der Seite Hintergrundfarbe, eine Schriftart und einen farbigen Header.

Aufgabe F (JavaScript): Baue einen Knopf, der beim Klick einen Text ändert, und danach einen, der eine Zahl hochzählt.

> **Hilfe:** Wenn dich eine Aufgabe nicht weiterbringt, schick mir deinen Code. Ich zeige dir, wo der Fehler liegt, und erkläre es Schritt für Schritt.