Tag 1 CSS: Selektoren

CSS-Selektoren sind das Fundament der Gestaltung mit CSS. Mit ihnen wählst du gezielt Elemente in deinem HTML aus, um sie zu stylen. CSS bietet dabei eine Vielzahl an Selektoren und Kombinationen, die du für spezifische Designs einsetzen kannst. In diesem Kapitel erweitern wir das Grundwissen und erklären die verschiedenen Selektortypen inklusive ihrer Notation und Anwendungsbeispiele. Dabei gehen wir tiefer auf die Bedeutung und Einsatzszenarien jedes Selektors ein.

Element-Selektor

Der Element-Selektor spricht alle HTML-Elemente eines bestimmten Typs an. Wenn du z. B. `p` als Selektor verwendest, wird die CSS-Regel auf **alle Absatz-Elemente** (`<p>`) auf der Seite angewendet.

```
p {
  color: blue;
}
```

Beispiel HTML:

```
<p>Ein erster Absatz.</p>
<p>Ein zweiter Absatz.</p>
```

Dieser Selektor ist nützlich, um Grundstile für Standard-Elemente festzulegen.

ID-Selektor (`#`)

Der ID-Selektor ermöglicht es, ein ganz bestimmtes Element gezielt zu stylen. Dabei wird das `id`-Attribut des Elements verwendet, das **einzigartig** innerhalb des Dokuments sein muss.

```
#header {
  background-color: lightgray;
}
```

Beispiel HTML:

```
<div id="header">Willkommen!</div>
```

Dieser Selektor ist ideal, wenn du **ein einzelnes, spezifisches Element** formatieren willst – z. B. eine Navigation, einen Footer oder einen Container.

> IDs sollten nur einmal pro Seite vorkommen, sonst kann es zu unerwartetem Verhalten kommen.

Klassen-Selektor (`.`)

Der Klassen-Selektor wendet Stile auf **alle Elemente an, die eine bestimmte Klasse besitzen**. Anders als IDs kann eine Klasse mehrfach im Dokument verwendet werden, was sie ideal für wiederverwendbare Komponenten macht.

```
.card {
  border: 1px solid #ccc;
  padding: 10px;
}
```

Beispiel HTML:

```
<div class="card">Beitrag 1</div>
<div class="card">Beitrag 2</div>
```

Mit Klassen lassen sich also **Gruppen von Elementen konsistent gestalten**.

Kombination: Element + Klasse

Wenn du nur bestimmte Elemente mit einer Klasse ansprechen willst, kombinierst du den Elementnamen mit dem Klassennamen.

```
p.note {
  font-style: italic;
}
```

Beispiel HTML:

```
<p class="note">Wichtiger Hinweis</p>
```

Diese Regel wirkt nur auf `<p>`-Elemente mit der Klasse `note`, nicht auf andere Elemente, die ebenfalls `note` als Klasse tragen.

Nachfahren-Selektor (Leerzeichen)

Ein Nachfahren-Selektor wird verwendet, um **alle Elemente innerhalb eines anderen Elements** zu stylen, egal wie tief sie geschachtelt sind.

```
#content p {
  color: darkgreen;
}
```

Beispiel HTML:

```
<div id="content">
  <p>Dies ist ein Absatz im Content-Bereich.</p>
</div>
```

Das ist hilfreich, wenn man z. B. bestimmte Stile nur in einem Container anwenden möchte, ohne andere Abschnitte zu beeinflussen.

Kind-Selektor (>)

Der Kind-Selektor spricht **nur die direkten Kinder** eines Elements an.

```
ul > li {
  list-style-type: square;
}
```

Beispiel HTML:

```
<ul>
  <li>Direktes Kind</li>
  <div>
    <li>Kein direktes Kind</li>
  </div>
</ul>
```

Im Unterschied zum Nachfahren-Selektor werden hier **nur die Elemente eine Ebene tiefer** gestylt. Das erhöht die Kontrolle und vermeidet ungewollte Seiteneffekte.

Adjacent Sibling Selector (+)

Dieser Selektor spricht **das unmittelbar folgende Geschwisterelement** eines bestimmten Elements an.

```
h2 + p {
  margin-top: 0;
}
```

Beispiel HTML:

```
<h2>Überschrift</h2>
<p>Direkt folgender Absatz</p>
```

Sehr nützlich, wenn du z. B. Abstand oder Formatierung nur für ein direkt folgendes Element anpassen willst.

General Sibling Selector (~)

Dieser Selektor spricht **alle nachfolgenden Geschwisterelemente** an, die denselben Elternelement haben.

```
h2 ~ p {
  color: teal;
}
```

Beispiel HTML:

```
<h2>Start</h2>
<p>Absatz 1</p>
<p>Absatz 2</p>
```

Praktisch, um z. B. bestimmte Folgen von Elementen auf Basis eines vorhergehenden zu beeinflussen.

Universal-Selektor (*)

Der Universal-Selektor wendet Stile auf **alle HTML-Elemente** an.

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

Das ist besonders nützlich für globale Layoutregeln oder CSS-Resets, um konsistente Ausgangspunkte zu schaffen.

Attribut-Selektor

Mit Attributselektoren kannst du Elemente ansprechen, die ein bestimmtes Attribut besitzen – optional auch mit bestimmten Werten.

```
a[target="_blank"] {
  color: red;
}
```

Beispiel HTML:

```
<a href="https://example.com" target="_blank">Externer Link</a>
```

Weitere Beispiele:

Selektor	Bedeutung
`[attr]`	Hat das Attribut überhaupt
`[attr="wert"]`	Attribut ist genau „wert“
`[attr^="wert"]`	Attribut beginnt mit „wert“ (Prefix)
`[attr$="wert"]`	Attribut endet mit „wert“ (Suffix)
`[attr*="wert"]`	Attribut enthält „wert“ irgendwo (Substring)

Dies erlaubt es, z. B. alle Checkboxen, Eingabefelder oder externe Links gezielt zu stylen.

Pseudo-Klassen

Pseudo-Klassen wählen Elemente auf Basis ihres **Zustands oder ihrer Position im DOM** aus.

```
a:hover {
  color: orange;
}

li:first-child {
  font-weight: bold;
}

input:focus {
  outline: 2px solid blue;
}
```

Diese Selektoren ermöglichen dynamische Stile, z. B. für Hover-Effekte, Formularzustände oder die gezielte Formatierung einzelner Listeneinträge.

Pseudo-Elemente

Pseudo-Elemente greifen auf virtuelle Teile eines Elements zu, die im HTML nicht direkt vorhanden sind.

```
p::first-line {
  text-transform: uppercase;
}

p::before {
  content: "Hinweis: ";
```

```
  font-weight: bold;
}
```

Das ist besonders hilfreich für typografische Stile oder um Inhalte rein visuell zu ergänzen, ohne das HTML zu ändern.

CSS Spezifität

Die **Spezifität** (engl. *specificity*) entscheidet, **welcher CSS-Selektor gewinnt**, wenn mehrere auf dasselbe Element zutreffen.

Gewichtung:

- **Inline-Styles** → `1000`
- **IDs (`#id`)** → `100`
- **Klassen, Attribute, Pseudoklassen** (`.klasse`, `[type="text"]`, `:hover`) → `10`
- **Elemente & Pseudoelemente** (`div`, `h1`, `::before`) → `1`

> Der Selektor `div .box #main` hat folgende Spezifität:
>
> - `div` = 1
> - `.box` = 10
> - `#main` = 100
>
> → **Gesamt: 111**

- Höhere Spezifität überschreibt niedrigere.
- Reihenfolge im CSS zählt nur bei **gleicher Spezifität**.

HTML zur Veranschaulichung:

```
<div id="main">
  <p class="text" style="color: green;">Hallo Welt!</p>
</div>
```

CSS-Regeln:

```
p { color: blue; }              /* Spezifität: 1 */
.text { color: red; }           /* Spezifität: 10 */
#main .text { color: orange; }  /* Spezifität: 110 */
```

→ **Ergebnis:** `orange`, weil `#main .text` die höchste Spezifität hat.

→ Wird `style="color: green"` verwendet, gewinnt **immer der Inline-Style**, weil **Spezifität 1000**.

Übungsaufgaben zu CSS-Selektoren

Aufgabe 1:

Erstelle eine Liste mit drei Einträgen. Wende mithilfe eines Kind-Selektors einen eigenen Stil auf die direkten `<li>`-Elemente an.

Aufgabe 2:

Erstelle zwei Absätze unter einer Überschrift. Verändere nur den ersten Absatz.

Aufgabe 3:

Füge einer `<a>`-Verlinkung das Attribut `target="_blank"` hinzu und style nur diesen Link mit einem Attributselektor rot.

Aufgabe 4:

Erstelle zwei Klassen `.info` und `.warnung` und weise sie verschiedenen Elementen zu. Style sie unterschiedlich und kombiniere jeweils mit einem Elementselektor (z. B. `p.warnung`).

Aufgabe 5:

Nutze einen Universal-Selektor, um auf der ganzen Seite eine Schriftart deiner Wahl zu setzen.

Aufgabe 6:

Verwende Pseudo-Klassen wie `:hover`, `:first-child` oder `:focus` in einem kleinen Formular oder einer Liste und beobachte die Auswirkungen.

Aufgabe 7:

Experimentiere mit `::before` und `::after`, um Inhalte vor oder nach einem Element anzuzeigen, z. B. "✔" vor einem Listenelement.