

# HTML Basics

## Inhalt

- [Was ist HTML?](#)
- [Aufbau einer HTML-Datei](#)
- [IDs & Klassen](#)
- [Block- & Inline-Elemente](#)
- [Neue Elemente](#)
- [Tabellen](#)

## Setup

Für Übungen nutzen wir die folgende **CodeSandbox** als Startpunkt:

<https://codesandbox.io/s/rrcjw>

Die Übungen bauen immer aufeinander auf. Aber keine Angst! Für den Fall, dass bei einer Übung etwas nicht klappen sollte, gibts bei jeder Übung einen Link zur CodeSandbox mit dem aktuellen Stand.

## Was ist HTML?

- «Hyper Text Markup Language»
- Auszeichnungssprache, keine Programmiersprache
- Zweck: Beschreiben von Inhalt
- Eigenheit: Presentation Semantics

## Eigenheiten einer HTML-Datei

- HTML benötigt keinen Server
- Dateien müssen mit `.html` enden
- Aufrufbar in jedem Webbrowser
- `index.html` ist der Root / die Startseite einer Website
  - <http://www.something.com>  
→ <http://www.something.com/index.html>  
Die `index.html` -Datei wird automatisch angezeigt
  - <http://www.something.com/about.html>  
→ <http://www.something.com/about.html>  
Hier verweisen wir explizit auf eine Seite
  - *Dieses Standardverhalten kann mit div. Servereinstellungen überschrieben werden.*

## Syntax

Offizielle Spezifikation: <https://www.w3.org/TR/2011/WD-html-markup-20110405/syntax.html>

Elemente starten mit einem `<` gefolgt vom **Tagnamen**. Anschliessend kommen *optionale* **Attribute** und deren Values. Das Ende des Starttags ist mit einem `>` gekennzeichnet.

Danach kommt der **Inhalt** des Elementes.

Abschliessend kommt der Endtag, dieser startet wieder mit einem `<` gefolgt von einem `/` und dem **Tagnamen**, danach wird dieser geschlossen mit einem `>`

```
<!-- Generelle Syntax -->
<tagname attribute="value">Inhalt</tagname>
```

```
<!-- Attribut ohne Wert -->
<tagname attribute>Inhalt</tagname>

<!-- Ohne Attribute -->
<tagname>Inhalt</tagname>
```

## Beispiele

```
<h1 id="my-id">Hallo Welt!</h1>
<p class="some-fany-class">Dies ist ein Paragraph</p>
```

## Void Elemente

Es gibt auch sogenannte void elements. Diese haben keinen Inhalt und der Endtag ist daher implizit und muss nicht angegeben werden.

```
<!-- Generelle Syntax -->
<tagname attribute="value">

<!-- Attribut ohne Wert -->
<tagname attribute>

<!-- Ohne Attribute -->
<tagname>

<!-- Mit zusätzlichem / -->
<tagname />
```

## Beispiele

```
<!-- Line Break <br> -->
<p>In diesem Satz gibt es<br>einen line break</p>

<!-- Image -->

```

Liste von void-elements: <https://html.spec.whatwg.org/#void-elements>

### Das Wichtigste in Kürze

- Elemente haben normalerweise einen Starttag, einen Inhalt und einen Endtag
- Die Ausnahme sind sogenannte void-elements. Diese haben keinen Inhalt und daher auch keinen expliziten Endtag
- Elemente können optional einen oder mehrere Attribute haben

## Aufbau einer HTML-Datei

HTML-Dateien haben einen gewissen Grundaufbau, damit der Browser genau weiss, was er machen muss.

```

<!DOCTYPE html>
<html>
  <head>
    <title>Test</title>
  </head>
  <body>
    <h1 class="fancy-title">Hello World</h1>
    
    <p>
      Lorem ipsum dolor sit amet, consetetur.
    </p>
  </body>
</html>

```

- Der **Doctype** sagt dem Browser, welchen HTML-Standard dieser benutzen soll, um das Dokument anzuzeigen.
- `<html>` ist das Root-Element des Dokuments
- Im `<head>` stehen **Metainformationen**. Diese werden **nicht angezeigt**, sondern sind nur Informationen für Browser und Roboter, die diese auslesen.
- Im `<body>` steht der Inhalt der Page. Elemente innerhalb des Bodys werden dem User angezeigt.

### Infos zum `<html>`

Die Sprache der Website sollte auf dem `<html>` -Element angegeben werden, damit Roboter und Screenreader wissen, in welcher Sprache die Page angezeigt wird.

### Beispiel

```

<!DOCTYPE html>
<html lang="en">
  ...
</html>

```

### Hilfreiche Links

- [Documentation auf MDN](#)

### Infos zum `<head>`

- Das `<title>` -Element ist das Einzige, welches required ist
- Der Head beinhaltet vor allem Metainformationen
- Darin befinden sich z.B. Verlinkungen von CSS-Dateien, JavaScript-Dateien oder auch Informationen für Roboter (Google, Facebook, Twitter usw.)

► Beispiel head von galaxus.ch

Folgendes sollte im `<head>` als **minimum** drin sein (abgesehen vom mandatory `<title>` ):

```

<head>
  <!-- Stellt sicher, dass das richtige Charset verwendet wird -->
  <meta charset="utf-8" />
</head>

```

## Hilfreiche Links

- [Documentation auf MDN](#)
- [Practical Guide](#)

### Das Wichtigste in Kürze

- Ein HTML-Dokument beinhaltet einen Doctype, das `<html>` -Element und darin den `<head>` und den `<body>`
- Im `<head>` ist das `<title>` -Element required, ansonsten stehen im Head vor allem Metainformationen
- Im `<body>` ist der Inhalt der Page angegeben, welche der User sehen kann

## Practice

Öffne diese [CodeSandbox](#) als Startpunkt.

- ☐ Füge ein HTML-Grundgerüst ein
- ☐ Wir bilden ein kleine Todo App. Du kannst einen passenden `<title>` einfügen

Zeit: ~ 5 min

**Solution:** <https://codesandbox.io/s/eemf9>

## IDs & Klassen

IDs und Klassen können auf jedes HTML-Element gesetzt werden. Sie dienen zur Identifikation und dem Styling von Elementen.

### IDs

- Können auf jedes Element mit dem `id` -Attribut gesetzt werden, sind aber überall optional
- Werden zur Identifikation von Elementen benutzt (z.B. um ein bestimmtes Element im JavaScript anzusteuern, oder um es mit CSS zu stylen)
- Dürfen jeweils nur einmal in einen HTML-Dokument vorkommen

### Beispiel

```
<div id="container">
  <h1 id="page-title">Headline</h1>
  ...
</div>
```

### Klassen

- Können auf jedem Element mit dem `class` -Attribut gesetzt werden, sind aber überall optional
- Klassen werden fürs Styling bevorzugt
- Es können mehrere Klassen auf ein Element angewandt werden, diese sind mit einem Leerzeichen zu trennen

### Beispiel

```
<div class="container">
  <h1 class="headline-primary second-class">
```

```
    Headline
    <h1>
    ...
  </div>
```

## Block- & Inline-Elemente

HTML-Elemente unterscheiden sich grundsätzlich in zwei Typen: in Block- und Inline-Elemente.

### Block-Element

- Block-Elemente nehmen immer die volle Breite des Elternelementes ein
- Aufeinanderfolgende Block-Elemente werden daher per default untereinander dargestellt

#### Beispiel

```
<p style="border: 2px solid green">
  Lorem ipsum dolor sit amet
  consectetur adipisicing elit.
  Aut earum dolor perferendis
  ducimus vitae soluta.
</p>
<p style="border: 2px solid green">
  Lorem ipsum dolor sit amet
  consectetur adipisicing elit.
  Aut earum dolor perferendis
  ducimus vitae soluta.
</p>
```

#### Demo 🍷

- [Block-Elemente](#)

### Inline Element

- Inline-Elemente besetzen nur den Platz, der benötigt wird
- Aufeinanderfolgende Inline-Elemente werden daher nebeneinander dargestellt

```
<p>
  <a href="#" style="border: 2px solid blue">
    Link A
  </a>
  <a href="#" style="border: 2px solid blue">
    Link B
  </a>
</p>
```

#### Demo 🍷

- [Inline-Elemente](#)

### Weitere Infos

- Das Verhalten von Elementen kann über die display Property mit CSS überschrieben werden (mehr dazu später)
- Eine Übersicht von Block- & Inline-Elementen unter [https://www.w3schools.com/html/html\\_blocks.asp](https://www.w3schools.com/html/html_blocks.asp)

## Practice

Öffne diese [CodeSandbox](#) als Startpunkt.

- ☐ Erstelle als erstes eine zweite Seite `about.html`
- ☐ Füge auf der `index.html` einen Link ein auf `about.html`
- ☐ Füge nun auf der `about.html` einen Link ein, sodass man wieder zur Startseite kommt
- ☐ Füge bei beiden Seiten einen `<h1>` mit einem passenden Titel ein
- ☐ Schreibe auf der `about.html` einen kurzen Paragraphen über Dich selbst und füge ein Bild ein.  
Du kannst dafür das vorhandene `avatar.jpg` nutzen oder selbst ein Bild per Drag & Drop hochladen und dieses einbinden
- ☐ Füge auf beiden Seiten einen Fussbereich ein, wo Du dein © platzieren kannst
- ☐ Passe deine Navigation so an, dass auf jeder Seite alle Navigationslinks vorhanden sind. Zeige die Navigationspunkt als Liste `<ul>` an

Hilfestellung und Spezifikation der benutzten Tags unter [https://www.w3schools.com/html/html\\_blocks.asp](https://www.w3schools.com/html/html_blocks.asp)

Zeit: ~ 20 min

**Solution:** <https://codesandbox.io/s/2spun>

## Infos zu den benutzten Elementen

- ▶ Headings und Paragraphen
- ▶ Links
- ▶ Bilder
- ▶ Listen
- ▶ Divs

## "Neue" Elemente

HTML5 bietet viele neue Elemente, die den Browser vor allem mit semantischen Informationen füttern. Dies ist speziell gut für Suchmaschinen und um eine bessere Accessibility zu erzielen.

Zu den meistbenutzten Elementen gehören:

- `<header>`
- `<footer>`
- `<nav>`
- `<main>`
- `<section>`
- `<article>`

- `<aside>`

Beschreibungen der Elemente unter: [https://developer.mozilla.org/en-US/docs/Web/HTML/Element#Content\\_sectioning](https://developer.mozilla.org/en-US/docs/Web/HTML/Element#Content_sectioning)

## Practice 🔥

Öffne diese [CodeSandbox](#) als Startpunkt.

- ☐ Strukturiere die Seite mit den neuen HTML5-Elementen
  - ☐ Nutze mindestens 4 der oben gelisteten Elemente
  - ☐ Strukturiere Deine Page so, wie es für Dich am meisten Sinn ergibt

Tipp: Du kannst Dir auch den Quellcode einer beliebigen Page anschauen und dort die Struktur analysieren

Hilfestellung und Beschreibungen zu den Elementen unter: [https://developer.mozilla.org/en-US/docs/Web/HTML/Element#Content\\_sectioning](https://developer.mozilla.org/en-US/docs/Web/HTML/Element#Content_sectioning)

Zeit: ~ 5 min

**Solution:** <https://codesandbox.io/s/f01xs>

## Tabellen

Tabellen sind dazu da um strukturierte Daten abzubilden. Sie sollten nicht für das Layout verwendet werden.

Bei Tabellen gibt es folgende Elemente:

- `<table>` definiert die Tabelle.
- `<tr>` definiert eine Tabellenzeile.
- `<th>` definiert eine Header-Zelle (für die Überschriften der Spalten).
- `<td>` definiert eine normale Datenzeile.

### Tabelle mit einer Kopfzeile

```
<table>
  <tr>
    <th>Header 1</th>
    <th>Header 2</th>
    <th>Header 3</th>
  </tr>
  <tr>
    <td>Zeile 1, Spalte 1</td>
    <td>Zeile 1, Spalte 2</td>
    <td>Zeile 1, Spalte 3</td>
  </tr>
  <tr>
    <td>Zeile 2, Spalte 1</td>
    <td>Zeile 2, Spalte 2</td>
    <td>Zeile 2, Spalte 3</td>
  </tr>
</table>
```

## Demo 🍷

- [Kopfzeile](#)

### Tabelle mit einer Kopfspalte

```
<table>
  <tr>
    <th>Header 1</th>
    <td>Zeile 1, Spalte 2</td>
    <td>Zeile 1, Spalte 3</td>
  </tr>
  <tr>
    <th>Header 2</th>
    <td>Zeile 2, Spalte 2</td>
    <td>Zeile 2, Spalte 3</td>
  </tr>
  <tr>
    <th>Header 3</th>
    <td>Zeile 3, Spalte 2</td>
    <td>Zeile 3, Spalte 3</td>
  </tr>
</table>
```

## Demo 🍷

- [Kopfspalte](#)

### Tabelle mit einer Kopfzeile & einer Kopfspalte

```
<table>
  <tr>
    <th>Header 1.1</th>
    <th>Header 1.2</th>
    <th>Header 1.3</th>
  </tr>
  <tr>
    <th>Header 2.1</th>
    <td>Zeile 2, Spalte 2</td>
    <td>Zeile 2, Spalte 3</td>
  </tr>
  <tr>
    <th>Header 3.1</th>
    <td>Zeile 3, Spalte 2</td>
    <td>Zeile 3, Spalte 3</td>
  </tr>
</table>
```

## Demo 🍷

- [Kopfzeile & Kopfspalte](#)

### Gruppierung von Zellen



Wir können unsere Zeller der Tabelle auch gruppieren. Dafür gibt es folgende Attribute:

- `colspan` : Anzahl der Spalten, die eine Zelle einnehmen soll
- `rowspan` : Anzahl der Zeilen, die eine Zelle einnehmen soll

### Beispiel

```
<table>
  <tr>
    <td colspan="2">Zeile 1, Spalte 1 & 2</td>
    <td>Zeile 1, Spalte 3</td>
  </tr>
  <tr>
    <td>Zeile 2, Spalte 1</td>
    <td rowspan="2">Zeile 2 & 3, Spalte 2</td>
    <td>Zeile 2, Spalte 3</td>
  </tr>
  <tr>
    <td>Zeile 3, Spalte 1</td>
    <td>Zeile 3, Spalte 3</td>
  </tr>
</table>
```

### Demo 🍷

- [Gruppierung von Zellen](#)