

# HTML(5)

*Eine Einführung in die grundlegenden Konzepte von HTML*

**Dozent:** Prof. Dr. Michael Eichberg  
**Kontakt:** [michael.eichberg@dhbw.de](mailto:michael.eichberg@dhbw.de), Raum 149B  
**Version:** 2.1.1

---

**Folien:** <https://delors.github.io/web-html/folien.de.rst.html>  
<https://delors.github.io/web-html/folien.de.rst.html.pdf>  
**Fehler melden:** <https://github.com/Delors/delors.github.io/issues>



# 1. Einführung

# HTML(5)

**Gegenstand:**

- Sprache zur Beschreibung der Darstellung von Inhalten (Markup Language), zwischen denen „navigiert“ werden kann (Hypertext).  
Ursprünglich war HTML ein Akronym für *HyperText Markup Language*; heutzutage wird HTML als Eigennamen verwendet.
- Auszeichnungssprache abgeleitet aus SGML (Standard Generalized Markup Language).

**Verwendung:**

- Webseiten
- Progressive Web-Apps
- Desktop Apps (z. B. mit Electron)

## Ein einfaches HTML(5) Dokument

```
1 <!DOCTYPE html>
2 <html lang="de">
3   <head><title>Eine Webseite</title></head>
4   <body>
5     <h1>Informationen</h1>
6     <p><!-- Ein Kommentar.. -->
7       Ein einfacher link auf:
8       <a href="http://www.michael-eichberg.de">
9         Michael Eichberg's Homepage
10      </a>.
11    </p>
12  </body>
13 </html>
```

---

Im Folgenden werden wir uns mit den grundlegenden Konzepten von HTML beschäftigen und die wichtigsten Bestandteile von HTML-Dokumenten kennenlernen. Das Ziel ist es, ein Verständnis für HTML zu entwickeln, dass für die fundierte Entwicklung von Webseiten erforderlich sind.

---

# HTML<sup>[1]</sup> - Historie

**WHATWG:** Web Hypertext Application Technology Working Group

---

<sup>[1]</sup> Im Folgenden bezeichnet HTML die HTML(5) Spezifikation (Living Standard).

# HTML Universum

Spezifikationen (siehe [2]), die im Rahmen der Entwicklung von Webseiten/Webanwendungen relevant sind:


- **CSS**: Cascading Style Sheets
- **JavaScript**: ECMAScript (Fetch API, XMLHttpRequest, ...)
- **DOM**: Document Object Model
- **Encoding**: Unicode
- **HTTP**: Hypertext Transfer Protocol
- **SVG**: Scalable Vector Graphics
- **MathML**: Mathematical Markup Language
- **WebAssembly**: Low-Level Bytecode
- ...

---

[2] <https://spec.whatwg.org>

# HTML vs. XML Syntax vs. DOM

Die *HTML Spezifikation* definiert eine abstrakte Sprache zur Beschreibung von Dokumenten. XML und HTML sind konkrete Syntaxbeschreibungen dieser abstrakten Sprache.

- HTML ist eine Beschreibungssprache für entsprechende Dokumente.
- XML ist eine allg. Beschreibungssprache, die auch für HTML verwendet wurde. (MIME Type: `application/xhtml+xml`)  
(Heutzutage wird XML zur Beschreibung von HTML Dokumenten nicht mehr verwendet.)
- Das DOM ( *Document Object Model*) ist die In-Memory Darstellung eines Dokuments.  
Das DOM ist ein API, um HTML Dokumente zu manipulieren.

## Hinweis

Das DOM, die HTML-Syntax und die XML-Syntax können nicht alle denselben Inhalt darstellen.

---

## Beispiele für Unterschiede

- Namespaces werden nicht von der HTML-Syntax unterstützt, aber sowohl vom DOM als auch der XML-Syntax unterstützt.
- `noscript` wird nur in HTML Dokumenten unterstützt.
- Kommentare, die `—>` enthalten, werden nur vom DOM unterstützt; in einem HTML-Dokument beendet dies den Kommentar.

# HTML in a Nutshell

HTML-Dokumente bestehen aus einem Baum von Elementen und Text.

## HTML Dokument

```

1  <!DOCTYPE html>
2  <html lang="de">
3  <head><title>Eine Webseite</title></head>
4  <body>
5    <h1>Informationen</h1>
6    <p><!-- Ein Kommentar.. -->
7      Ein einfacher link auf
8      <a href="
9        http://www.michael-eichberg.de
10       ">
11        Michael Eichberg's Homepage
12      </a>.
13    </p>
14  </body>
15  </html>

```

## DOM

```

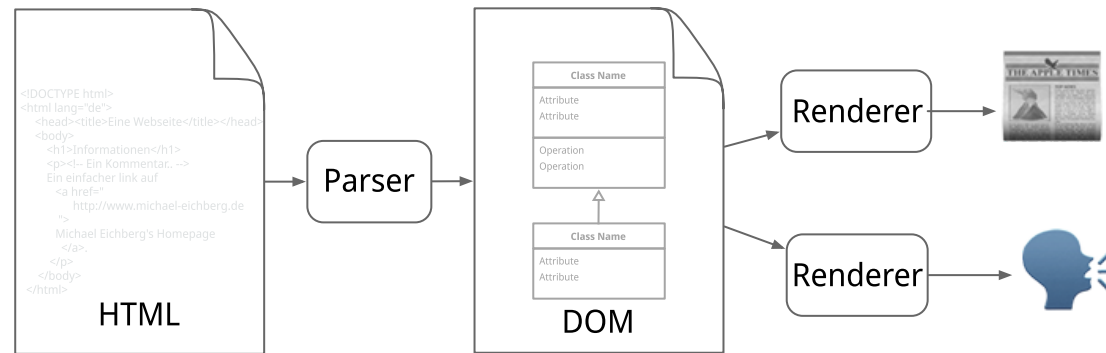
1  | DOCTYPE: html
2  | html lang="de"
3  |   | head
4  |   |   | title
5  |   |   |   | #text: Eine Webseite
6  |   |   |   | #text: 
7  |   |   |   | body
8  |   |   |   |   | #text: 
9  |   |   |   |   | h1
10 |   |   |   |   | ...

```

Mehrere HTML Dokumente bilden ggf. auf den selben DOM ab. Zum Beispiel werden die Tags als solches gar nicht abgebildet und wenn im HTML Code ein optionales (schließendes) Tag fehlt, dann ist dies im DOM nicht mehr ersichtlich.



# HTML - Verarbeitung



# Übung

## 1.1. Vertraut machen mit dem DOM

1. Öffnen Sie eine Webseite (z. B. <https://delors.github.io/web-html/folien.de.rst.html>)
2. Öffnen Sie die Entwicklerconsole Ihres Browsers
3. Lassen Sie sich den Sourcecode anzeigen
4. Vergleichen Sie den DOM mit dem Sourcecode

## 2. Aufbau von HTML Dokumenten

# HTML Dokumente

Die Dokumente müssen aus den folgenden Teilen in der angegebenen Reihenfolge bestehen:

- Optional ein einzelnes U+FEFF BYTE ORDER MARK (BOM) Zeichen.
- Eine beliebige Anzahl von Kommentaren und ASCII-Whitespace.
- Eine DOCTYPE Deklaration: `<!DOCTYPE html>`.
- Eine beliebige Anzahl von Kommentaren und ASCII-Whitespace.
- Das **Dokumentelement** in Form eines `html`-Elements<sup>[3]</sup>.
- Eine beliebige Anzahl von Kommentaren und ASCII-Leerzeichen.

—HTML Spezifikation

[3] HTML ist nicht case-sensitive, d. h. die Tags `html` und `HTML` sind gleichwertig. Wir verwenden jedoch immer die Kleinschreibung.

## Allgemeiner Aufbau von HTML Elementen

Start Tag	Inhalt des Elements	End Tag
<code>&lt;b&gt;</code>	Sehr Wichtig...	<code>&lt;/b&gt;</code>
Element		

### Warnung

Die Spezifikation verlangt nicht in allen Fällen ein Start und Endtag. Es ist jedoch eine gute Praxis, diese immer zu verwenden, wenn ein Endtag möglich ist.

Im Fall von Elementen ohne Endtag (z. B. `<br>`<sup>[4]</sup>) darf auch keines hinzugefügt werden!

---

### Beispiel

```
1 <!DOCTYPE HTML><head>
2   <title>Hello</title>
3 </head>
4 <body>
5   <p>Welcome to this example.</p>
6 </body>
7 </html>
```

Ist ein gültiges Dokument. Es ist jedoch **keine** gute Praxis (hier wurde das *Start Tag* des `html` Elements weggelassen).

[4] `<br>` steht für Line Break Opportunity.

# Typen von HTML Elementen

Sechs Typen von HTML-Elementen werden unterschieden:

**Void elements:**      `area, base, br, col, embed, hr, img, input, link, meta, source, track, wbr`

**Raw text elements:** `script, style`

**Escapable raw text elements:**  
`textarea, title`

**Das *template* Element:**  
`template`


**Foreign elements:** Elemente aus dem MathML- und SVG-Namensraum.

**Normal elements:** **Alle weiteren HTML Elemente** sind *normale Elemente*.

# Attribute in HTML

Attribute liefern Informationen über das Element.

Start Tag
<code>&lt;a class="obsolete" href="#top" &gt;...</code>
Attribute

- Attribute kommen nur beim Start Tag vor.
- Attribute (in HTML) können, müssen aber kein Wert haben (Boolsche Attribute).
- Attributwerte sollten in Anführungszeichen (  *quoted*) (entweder: " oder ' ) stehen, müssen aber nicht.  
Werte ohne Anführungszeichen dürfen keine Leerzeichen oder Anführungszeichen enthalten.  
Welche Anführungszeichen verwendet werden, ist egal. Es ist jedoch eine gute Praxis, immer die gleichen Anführungszeichen zu verwenden.[5]
- Konkrete Attributwerte aus der HTML Spezifikation sind case-insensitive; andere Werte sind es nicht.

---

Im Allgemeinen sollten Attributwerte klein geschrieben werden. Selektoren in CSS und JavaScript sind case-sensitive.

Z. B. ist `<input type="text">` und `<input type="TEXT">` gleichwertig, aber `<div id="text">` und `<div id="Text">` nicht!

---

[5] Sollte man innerhalb eines Attributwertes Anführungszeichen verwenden wollen, dann sind HTML Entities notwendig: `&quot;` für " und `&#39;` für '.

# HTML Grundgerüst


```
1 <!DOCTYPE HTML">  
2 <html lang="de">  
3   <head>  
4     ... Meta-Daten, Scripte, Stylesheets, etc. ...  
5   </head>  
6  
7   <body>  
8     ... das Dokument ...  
9   </body>  
10 </html>
```



## HTML `<head>` Element - „obligatorische“ Elemente


Im Head sollten immer die folgenden Informationen deklariert werden:

- Titel:** Der Titel des Dokuments mit Hilfe des `<title>...</title>` elements
- Zeichensatz:** Der verwendete Zeichensatz mit Hilfe des passenden meta elements: `<meta charset="utf-8">`
- Viewport:** Konfiguration des Viewports<sup>[6]</sup> (insbesondere für mobile Geräte relevant): `<meta name="viewport" content="width=device-width, initial-scale=1.0">`

Insbesondere Mobilgeräte haben oft entweder eine geringere Auflösung als Desktop-Computer oder verwenden HiDPI Screens. Beides führt dazu, dass die Webseiten nicht wie gewünscht aussehen. In diesem Fall verwenden die Browser für die Webseiten einen virtuellen Viewport mit (z. B.) 960px und skalieren dann die Seite (z. B.) auf 390px herunter. Wenn dieses Verhalten nicht gewünscht ist — z. B. weil die Seite  *Responsive* ist oder von vornherein auf mobile Endgeräte ausgerichtet ist — dann ist auf jeden Fall eine *Viewport* Konfiguration notwendig.

Device	Viewport Size (width x height)	Device Resolution (width x height)
iPhone 12	390 x 844	1170 x 2532
iPhone 12 Mini	360 x 780	1080 x 2340
iPhone 12 Pro	390 x 844	1170 x 2532
iPhone 12 Pro Max	428 x 926	1248 x 2778

Siehe: <https://experienceleague.adobe.com/en/docs/target/using/experiences/vec/mobile-viewports.html?lang=de> für weitere Details.

[6] Der *Viewport* ( *Ansichtsbereich*) des Browsers ist der Bereich des Fensters, in dem der Webinhalt zu sehen ist.

## HTML `<head>` Element - weitere Elemente

Im Head können weitere Informationen und Pragmas deklariert werden bzw. sollten dort deklariert werden, wenn sie benötigt werden:

**Skripte:** `<script [src="script.js" [defer|async]]></script>`

**CSS:** Mittels `<link rel="stylesheet" href="style.css">` oder `<style>...</style>`

**Favorite Icon:** `<link rel="icon" type="image/png" href="/img/icon.png" />`

**Pragmas:** `http-equiv` = *HTML equivalent*; d. h. die Informationen könnte auch im HTTP Header stehen.

- `<meta charset="utf-8">` (alt)

- `<meta http-equiv="Content-Security-Policy" content="default-src https:">`

Äquivalente HTTP Header Definition:

Content-Security-Policy: default-src https:

**Benannte Meta-Daten:**

`<meta name="author" content="Michael Eichberg">`

---

### Content Security Policies

Pragmas sind Anweisungen, die die Steuerung/Verarbeitung beeinflussen.

# Semantisches vs. „generisches“ HTML

## Semantisches HTML

- Verwendung von HTML Elementen, die die Bedeutung des Inhalts klar machen.
- Bessere Zugänglichkeit
- Bessere Suchmaschinen-Optimierung

### Beispiel Elemente

`<header>`, `<footer>`, `<nav>`, `<article>`, `<section>`, `<aside>`, `<main>`, `<figure>`,  
`<address>`, `<em>`, `<s>`, ...

## Semantik freies HTML

- Verwendung von `<div>` und `<span>` Elementen, um den Inhalt zu strukturieren.
- Keine klare Bedeutung des Inhalts.

## Semantisches vs. „generisches“ HTML - Beispiel

```

1 <div>
2   <span>Zwei Wörter</span>
3   <div>
4     <a>Ein Wort</a><a>Ein Wort</a>
5   </div>
6 </div>
7 <div>
8   <div>
9     <div>Viele Wörter</div>
10   </div>
11   <div>
12     <div>Erste Worte</div>
13     <div>DaDaDa</div>
14     <div>BlaBlaBla</div>
15   </div>
16 </div>
17 <div><span>Alle Worte</span></div>

```

```

1 <header>
2   <h1>Zwei Wörter</h1>
3   <nav>
4     <a>Ein Wort</a><a>Ein Wort</a>
5   </nav>
6 </header>
7 <main>
8   <header>
9     <h1>Viele Wörter</h1>
10  </header>
11  <section>
12    <h2>Erste Worte</h2>
13    <p>DaDaDa</p>
14    <p>BlaBlaBla</p>
15  </section>
16 </main>
17 <footer><p>Alle Worte</p></footer>

```

Semantische Informationen im DOM zu haben, ist insbesondere für die Barrierefreiheit notwendig.

Alternativ zur Verwendung von semantischen Elementen können auch generische Attribute mit dem `role` Attribute versehen werden, um die Bedeutung des Elements zu spezifizieren: `<div role="navigation">...</div>`

- 01 Verwenden Sie HTML zur Strukturierung von Inhalten, und nicht, um das Aussehen der Inhalte zu definieren.
- 02 Das Aussehen ist Sache von CSS.

# Strukturierung von Dokumenten

- header, footer, nav, article, section, aside, main, figure, address, ...  
In Hinblick auf die konkrete Semantik eines Elements gibt es Unterschiede wo und wie oft diese verwendet werden.  
Ein footer Element innerhalb eines article Elements hat eine andere Bedeutung als ein footer Element auf oberster Ebene.  
Ein main Element sollte nur einmal pro Dokument verwendet werden.
- Überschriften: h1, h2, h3, h4, h5, h6  
Überschriften sollten in der richtigen Reihenfolge verwendet werden.
- Überschriften gruppiert mit zugehörigem Inhalt: hgroup.

---

Das hgroup-Element stellt eine Überschrift und den zugehörigen Inhalt dar. Dient dazu eine Überschrift mit einem oder mehreren p-Elementen zu gruppieren. Zum Beispiel für eine Unterüberschrift oder einen alternativen Titel.

---

# Attribute

**Boolsche Attribute:** sind wahr, wenn diese angegeben sind und falsch andernfalls.

Z. B. `<input id="the-checkbox" type="checkbox" checked>`.

**Aufgezählte Attribute** ( *enumerated values*):

definieren eine begrenzte Anzahl von gültigen Werten sowie einen Default, der verwendet wird, wenn kein Wert angegeben ist, aber das Attribut verwendet wird.

**Globale Attribute:** können für jedes Element verwendet werden; sind aber nicht immer sinnvoll.<sup>[7]</sup>

Globale HTML Attribute sind Z. B. `id`, `class`, `data-*`, `autofocus`, `role`, `lang`, `style`, `popover`, `tabindex`.

Veraltet

Event Handler Attribute: `onclick`, `onclose`, ...

Boolsche Attribute sollten in JavaScript durch hinzufügen bzw. löschen gesetzt werden (und nicht durch die Manipulation des Wertes eines Attributs).

```
1 | const checkbox = document.getElementById("the-checkbox");
2 | checkbox.removeAttribute("checked");
3 | checkbox.setAttribute("checked");
```

Der Wert eines Attributs kann über mehrere Zeilen gehen solange diese keine Anführungszeichen enthalten. Zeilenumbrüche und Einrückungen (mit Tabulatoren (→)) werden dabei automatisch gefiltert.

Zum Beispiel kann der `content`-Wert des `meta`-Elements wie folgt geschrieben werden:

```
<meta name="author" content="
→ Michael Eichberg
→ Professor
">
```

Dies ist äquivalent zu:

```
<meta name="author" content="Michael Eichberg Professor">
```

<sup>[7]</sup> Globale Attribute

## Ausgewählte globale Attribute

id:	<ul style="list-style-type: none"><li>■ verwendet, um ein Element eindeutig zu identifizieren (Welches man in CSS oder JavaScript per Selektor referenzieren kann.)</li><li>■ als Ziel von Hyperlinks (<code>&lt;a href="#id"&gt;</code>)</li><li>■ im Rahmen der Unterstützung von Barrierefreiheit</li><li>■ der Wert ist case-sensitive</li></ul> <p>Best Practice: Kleinbuchstaben und Bindestriche verwenden (Unterstriche sind erlaubt aber im Zusammenhang mit CSS nicht optimal).</p>
class:	<ul style="list-style-type: none"><li>■ das <code>class</code>-Attribut ermöglicht es Elemente mit CSS und JavaScript anzusprechen</li><li>■ dient keinem anderen Zweck in HTML</li><li>■ wird sehr häufig von Frameworks und Bibliotheken verwendet</li></ul>
style:	Das <code>style</code> -Attribut ermöglicht die (ad-hoc) Anwendung von Inline-Styles auf das entsprechende Element (nicht empfohlen).
data-*:	Das <code>data-*</code> -Attribut ermöglicht es, benutzerdefinierte Daten an das Element zu binden, die von JavaScript verwendet werden können. <code>*</code> kann ein beliebiger Name sein, aber nicht <code>xml</code> oder <code>:</code> enthalten.



# HTML - logische Gruppierung von Text

## Paragrafen:

`<p>Inhalt</p>`

## Zitate:

`<blockquote>` und `<q>` (für kurze Zitate innerhalb eines Absatzes)

Das Inline-Zitat-Element `<q>` fügt der Sprache entsprechende Anführungszeichen hinzu.

### Beispiel


„Ein Zitat“ (deutsch)

`<q lang="de">Ein Zitat</q>`

„A quote“ (englisch)

`<q lang="en">A quote</q>`

## Betonung:

`<em>` ( *emphasized*) und `<strong>`

## Randbemerkungen:

`<small>` - für Randbemerkungen und Kleingedrucktes (d. h. `small` steht nicht für unwichtige(re)n Text oder die Schriftgröße)

## Veraltet bzw. nicht mehr korrekt:

`<s>`

## Zitierung:

`<cite>` - für den **Titel** eines Werkes oder einer Publikation

## Definitionen:

`<dfn [title="der definierte Begriff"]>` - für die Definition eines Begriffs

## Abkürzungen:

`<abbr title="HyperText Markup Language">HTML</abbr>` - für Abkürzungen

## Zeitangaben:

`<time datetime="2021-10-01">1. Oktober 2021</time>` - für Zeitangaben

`datetime` dient der Zeitangabe in einem maschinenlesbaren Format (z. B. für Suchmaschinen)

## Code:

`<code>` - für Code; für das Darstellen von Code-Beispielen wird `code` häufig mit `<pre>` kombiniert; die Sprache des Codes wird dann über ein `class` Attribut spezifiziert (z. B. `<pre><code class="language-java">...</code></pre>`)

## Variablen:

`<var>` - für Variablen in mathematischen oder Programmierkontexten

## (Tastatur-)Eingaben:

`<kbd>` - für Tastatureingaben oder andere Benutzereingaben

Drücken Sie `<kbd>cmd</kbd>` + `<kbd>c</kbd>` zum Kopieren.

## Hoch-/Tiefstellung:

`<sup>` und `<sub>` - für Hoch- und Tiefstellung, die nicht typographisch Zwecken dient, sondern inhaltlichen Zwecken.

H`<sub>2</sub>`O steht für Wasser.

**Text mit abweichender Bedeutung:**

`<i>` - Text, der von normaler Prosa abweicht wie z. B. eine taxonomische Bezeichnung, ein technischer Begriff, ...

Brot besteht aus `<i>`Mehl`</i>`.

**Text mit erhöhter Aufmerksamkeit:**

`<b>` - Text, der erhöhte Aufmerksamkeit erfordert, aber nicht unbedingt betont werden muss; z. B. Schlüsselwörter in einem Artikel.

`<p>`Das `<b>`Wetter`</b>` ist heute schön.`</p>`

**Text mit erhöhter Bedeutung:**

`<mark>` - Text, der hervorgehoben werden soll, z. B. Suchergebnisse.

---

Es gibt weitere Elemente, die für spezielle Anwendungsfälle verwendet werden können. Siehe **WHATWG**.

---


# HTML - physische Auszeichnung von Text

## Vorformatierter text:

`<pre> . . . </pre>` - für Text, der so angezeigt werden soll, wie er geschrieben wurde.

**Zeilenumbrüche:** `<br>` - für Zeilenumbrüche, die inhärenter Teil der Daten sind wie zum Beispiel bei Adressen. D. h. sollte nicht innerhalb von Text verwendet werden!

## Optionale Zeilenumbrüche:

`<wbr>` ( *word break opportunity*) - ein optionaler Zeilenumbruch

Beispiel:

```
<p> Er schrie:  
<q lang="de">Lasst<wbr>Mich<wbr>In<wbr>Ruhe!</q>  
</p>
```

## HTML - `<span>` und `<div>`

- `<span>` und `<div>` sind generische Container-Elemente, die verwendet werden, um Text oder andere Elemente zu gruppieren.
- `<span>` ist ein Inline-Element
- `<div>` ist ein Block-Element
- beide werden häufig verwendet, um CSS-Klassen zuzuweisen, um den Inhalt zu gruppieren oder um den Inhalt zu manipulieren.

## HTML - `<data>`

- Das `<data>`-Element ermöglicht es, maschinenlesbare Daten an ein Element zu binden, die dann von JavaScript verwendet werden können.
- Die maschinenlesbaren Daten werden im `value` Attribut gespeichert.

```
<data value="8">Acht</data>
```

# HTML - Links

- Hyperlinks werden mit dem `<a>` Element erstellt.
- Der `href`-Attribut enthält die Adresse des Ziels (innerhalb des gleichen Dokuments, auf einer anderen Webseite, per E-Mail, ...)

```
1 <a href="https://www.dhbw-mannheim.de">DHBW Mannheim</a>
2 <a href="#teachers">Unsere Lehrenden</a>
3 <a href="mailto:michael.eichberg@dhbw.de">Email: Michael Eichberg</a>
```

## 1. Externer Link

## 2. Interner Link ( *link fragment identifier*) auf ein Element mit der ID `teachers`

## 3. E-Mail Link - kann ergänzt werden durch `subject` und `body` Parameter innerhalb des `href` Attributs.

- Das `target`-Attribut ermöglicht die Definition des Browsing-Kontextes für die Link-Navigation (und die Formularübermittlung).

```
1 <a target="_blank" href="https://www.dhbw-mannheim.de">DHBW Mannheim</a>
2 <a target="_self" href="https://www.dhbw-mannheim.de">DHBW Mannheim</a>
3 <a target="_top" href="https://www.dhbw-mannheim.de">DHBW Mannheim</a>
4 <a target="_parent" href="https://www.dhbw-mannheim.de">DHBW Mannheim</a>
5 <a target="dhbw" href="https://www.dhbw-mannheim.de">DHBW Mannheim</a>
```

## 1. Öffnet den Link in einem neuen Fenster oder Tab

## 2. Öffnet den Link im gleichen Browsing-Kontext

## 3. Öffnet den Link im obersten Browsing-Kontext

## 4. Öffnet den Link im übergeordneten Browsing-Kontext

## 5. Öffnet den Link im Browsing-Kontext mit dem Namen `dhbw` (Beispiel: **DHBW Mannheim**)

`_self`, `_top` und `_parent` sind relativ zum aktuellen Browsing-Kontext und unterscheiden sich nur, wenn die Seite in einem Frame oder einem `<iframe>` angezeigt wird.

- Das `rel`-Attribut legt die Art des Links fest und definiert die Beziehung zwischen dem aktuellen Dokument und der Ressource, auf die der Hyperlink verweist. (Z. B. `rel="license"`, `rel="author"` oder `rel="noopener"`; siehe **MDN rel attribute**)

---

Durch die Zuweisung zu einem Browsing-Kontext kann verhindert werden, dass die selbe Seite X mal geöffnet wird, wenn ein Nutzer auf den Link klickt.

# HTML - Lists

Drei Arten von Listen werden unterstützt, die beliebig verschachtelt werden können:

## ■ Definitionslisten: `<dl>`

```
<dl>
  <dt>Erster Begriff</dt>
  <dd>Erklärung des 1. Begriffs</dd>
  <dt>Zweiter Begriff</dt>
  <dd>Erklärung des 2. Begriffs</dd>
</dl>
```

### **Erster Begriff**

Erklärung des 1. Begriffs

### **Zweiter Begriff**

Erklärung des 2. Begriffs

## ■ geordnete Listen: `<ol [reversed] [start=<NO>]>`

```
<ol start="0">
  <li>Erster Punkt</li>
  <li>Zweiter Punkt</li>
  <li value="10">Dritter Punkt</li>
</ol>
```

0. Erster Punkt  
1. Zweiter Punkt  
10. Dritter Punkt

## ■ ungeordnete Listen: `<ul>`

```
<ul>
  <li>Erster Punkt</li>
  <li>Zweiter Punkt</li>
</ul>
```

■ Erster Punkt  
■ Zweiter Punkt

## HTML - Navigation

- Das `<nav>` Element wird verwendet, um Navigationslinks zu gruppieren.
- Insbesondere für Screenreader und die Suchmaschine relevant.

### Beispiel

```
1 <nav id="ld-menu">
2   <button type="button"
3     id="ld-slides-button"
4     aria-label="show slides"></button>
5   <button type="button"
6     id="ld-slides-with-nr-button"
7     aria-label="show slides with numbers"></button>
8   <button type="button"
9     id="ld-help-button"
10    aria-label="show help"></button>
11 </nav>
```



# HTML - Tabellen

Verwendet für die Darstellung von tabellarischen Daten mit Zeilen und Spalten.

## Achtung!

Die Verwendung von `<table>` sollte sich nach dem Inhalt richten!

Tabellen sollten nicht zum Layout von Webseiten verwendet werden.

Aufbau von Tabellen:

```

1 <table>
2   <caption>Logische Operation</caption>
3   <thead>
4     <tr><th>not xor</th><th>1</th><th>0</th></tr>
5   </thead>
6   <tbody>
7     <tr><th>1</th><td>1</td><td>0</td></tr>
8     <tr><th>0</th><td>0</td><td>1</td></tr>
9   </tbody>
10  <tfoot></tfoot>
11 </table>

```

Logische  
Operation

not xor	1	0
1	1	0
0	0	1

- Zellen, die über mehrere Spalten oder Zeilen gehen können mit Hilfe von `colspan` und `rowspan` Attributen definiert werden.
- Spalten und Zeilen können mit Hilfe von `<col>` und `<colgroup>` Elementen definiert werden.

# HTML - Images

- Bilder werden mit dem `<img>` Element eingebunden.

```

```

- Das `src`-Attribut enthält die Adresse des Bildes.
- Das `alt`-Attribut enthält eine Beschreibung des Bildes, die angezeigt wird, wenn das Bild nicht geladen werden kann.
- Das `width` und `height`-Attribut können und sollten verwendet werden, um die Größe des Bildes festzulegen.
- Lazy loading ist durch die Verwendung des `loading` Attributs möglich (`loading="lazy"`).
- Folgende Bildformate werden breit unterstützt: `jpg`, `png`, `gif`, `svg` und `webp`.
- Responsive Images werden über das `srcset` Attribut unterstützt:

```
1 
```

Weitere **Responsive Features** werden mittels CSS ermöglicht. Um zum Beispiel zu verhindern, dass ein Bild größer als eine Textzeile wird, kann folgendes CSS definiert werden:

```
img {  
  max-inline-size: 100%;  
  block-size: auto;  
}
```

# HTML - Formulare

Formulare werden mit dem `<form>` Element erstellt.

- `action` enthält die Adresse, an die die Formulardaten gesendet werden.
- `method` definiert die Methode, die zum Senden der Daten verwendet wird (GET oder POST).
- `name` setzt den Namen des Formulars.
- `target` enthält den Namen des Browsing-Kontexts, in dem die Antwort angezeigt wird.
- `autocomplete` ermöglicht das automatische Ausfüllen von Formularen.
- `novalidate` verhindert die Validierung der Formulardaten durch den Browser.
- `accept-charset` definiert die Zeichencodierung, die zum Senden der Formulardaten verwendet wird.

Formularelemente werden mit dem `<input>` Element erstellt.

- `type` definiert den Typ des Formularelements.
- `name` definiert den Namen des Formularelements.
- `value` definiert den Wert des Formularelements.
- `placeholder` definiert den Platzhaltertext des Formularelements.
- `required` definiert, ob das Formularelement erforderlich ist.
- `disabled` definiert, ob das Formularelement deaktiviert ist.
- `autofocus` definiert, ob das Formularelement den Fokus erhält.

## Beispiel

```
<form method="GET"
      name="Folienauswahl">
  <label for="slide">Folie:</label>
  <select name="id-slide-no" id="slide">
    <option value="8">Elemente</option>
    <option value="10">Attribute</option>
    <option value="29">Formulare</option>
  </select>
  <input type="submit" value="Submit">
</form>
```

Folie: Elemente ▼

Submit

---

Für weitere Informationen bzgl. Formulare siehe [MDN Web Docs](#) oder [Web.dev](#).

# HTML - Zusammenfassungen und Details

Anzeigen von optionalen Details mit Hilfe des `<details>` Elements.

```
1 <details [open]>
2   <summary>Abstract</summary>
3   <p>Password guessing ...</p>
4 </details>
```

Geschlossen - Details werden erst nach einem Klick angezeigt:

► Abstract

Offen - Details werden direkt angezeigt:

▼ Abstract

Password guessing ...

## HTML - Dialoge

Dialoge werden mit Hilfe des `<dialog>` Elements erstellt. Dialoge sind spezielle Fenster, die den Fokus auf sich ziehen und die Interaktion mit dem Rest der Seite unterbrechen - falls diese modal sind.

Beispiel[8]:

Open Dialog

```
1 <dialog>
2   <h1>Dialog</h1>
3   <p>Dialog Inhalt</p>
4   <button formmethod="dialog">
5     OK
6   </button>
7   <button autofocus>Abbrechen</button>
8 </dialog>
```

[8] JavaScript Code zum Öffnen des Dialogs wird hier nicht gezeigt.

# HTML Entities

Ausgewählte Zeichen können (in manchen Kontexten) nur durch HTML Entities dargestellt werden:

- < durch &lt; oder &#60; (<)
- > durch: &gt; oder &#62; (>)
- & durch: &amp; oder &#38; (&)
- " durch: &quot; oder &#34; (")
- \_ durch: &nbsp; ( )

## Liste benannter Zeichen

Die numerischen Werte sind **Unicode Code Points** (d.h. #60 ist der Unicode Code Point von <).

# Eingebettet Webseiten

Das `<iframe>` Element ermöglicht das Einbetten von Webseiten in Webseiten:

```
1 <iframe src="https://www.dhbw-mannheim.de"
2   width="600"
3   height="400">
4   iframes are not supported</iframe>
5
6 <iframe srcdoc="
7   <h1>HTML</h1>
8   <p>HTML is a markup language.</p>"
9   width="600"
10  height="400">
11  iframes are not supported
12 </iframe>
```

## HTML

HTML is a markup language.

# HTML Erweiterbarkeit

- Hinzufügen von Meta-daten (`<meta name="" content="">`)
- class Attribute
- „Custom Elements“ (z. B. `<my-element>`)
- Autoren können APIs mit Hilfe des JavaScript-Prototyping-Mechanismus erweitern



## Veraltetes - aber noch unterstütztes - HTML

- keine `border` Attribute auf `<img>` Elementen
- keine `charset` Attribute auf `<script>` Elementen (UTF-8 ist gefordert)
- keine `language` Attribute auf `<script>` Elementen (JavaScript ist der Standard)
- kein `type` Attribute auf `<style>` Elementen (`text/css` ist der Standard)

## HTML - „nicht mehr unterstützt - April Stand 2024“

### Nicht mehr unterstützte Elemente (Auswahl):

- `<big>`
- `<blink>`
- `<center>`
- `<font>`
- `<marquee>`
- `<nobr>`
- `<tt>`
- `<menuitem>`
- ...

### Nicht mehr unterstützte Attribute (Auswahl):

- `align` bei `<h1>` bis `<h6>` Elementen
- `bgcolor` bei `<body>` Elementen
- `charset` bei `<a>` und `<link>` Elementen
- `name` bei `<img>`, `<option>`, ... Elementen

## Barrierefreiheit ist verpflichtend

*[...] Webseiten, Onlineshops, Apps, Onlinebuchungs- und Ticketdienste, Fahrkartenautomaten, Selbstbedienungs- und Zahlungsterminals, Telefon- und Messenger-Dienstleistungen sowie E-Book-Reader ab Juni 2025 barrierefrei sein müssen. Denn am 28. Juni 2025 tritt das Barrierefreiheitsstärkungsgesetz (BFSG)[9] in Kraft. [...]*

*[...] Diese [die Barrierefreie-Informationstechnik-Verordnung (BITV 2.0)] regelt, wie Behörden und Ministerien Webseiten gestalten müssen. In Paragraph 3 Absatz 2 ist festgelegt, dass diese dann als barrierefrei anzusehen sind, wenn sie Normen erfüllen, die im Amtsblatt der EU genannt werden – beispielsweise die EN 301 549 oder die Web Content Accessibility Guidelines 2.1 (WCAG) des World Wide Web Consortiums.*

*[...] Verstöße gegen das Barrierefreiheitsstärkungsgesetz [...] muss mit einem Bußgeld bis zu 100.000 Euro rechnen. [...]*

*—Sept. 2024; **Golem.de - Deutsche Webseiten sind versetzungsgefährdet***

---

[9] <https://bfsg-gesetz.de/>

## Web Content Accessibility Guidelines 2.1 (WCAG)

Einige Anforderungen<sup>[10]</sup>:

### „Harte“ Kriterien

- die Schrift auf einer Website wenigstens 16 Pixel groß und
- Zeilen nicht mehr als 80 Zeichen lang sein sollen.
- Der Abstand dazwischen soll das 1,5-Fache ihrer Höhe betragen.
- Um einen ausreichenden Kontrast zu erzeugen, muss die Schrift wenigstens 4,5-mal dunkler sein als der Hintergrund.
- Klickflächen schließlich müssen wenigstens 44 x 44 Pixel groß sein.

### „Weiche“ Kriterien

- Starke Animationen sind auf barrierefreien Seiten tabu.
- das Layout der Seite soll außerdem einfach, logisch und auf jeder Unterseite einheitlich aufgebaut sein.

---

[10] Sept. 2024; [Golem.de](#) - So setzen gute Webdesigner Barrierefreiheit um

## Referenzen

- [MDN Web Docs](#)
- [caniuse.com: Unterstützung von HTML, CSS etc. Features](#)
- [HTML \(Living Standard\) \(aka HTML5\)](#)
- [HTML DOM](#)
- [Web Content Accessibility Guidelines](#)

# Übung

## 2.1. Eine einfache HTML Webseite

Erzeugen Sie ein HTML Dokument, das wie das Dokument auf der rechten Seite aussieht. Verwenden Sie kein CSS!

Nutzen Sie den [HTML Validator](#), um zu verifizieren, dass Ihr Dokument valide ist.

Achten Sie auf eine korrekte Strukturierung des Dokuments und verwenden Sie semantische Elemente, wo immer dies sinnvoll ist. Denken Sie auch grundlegend an die Barrierefreiheit.

### Ausbildung Ehrenamt

#### **Lebenslauf**

*Musterstraße 1*

*12345 Musterstadt*

[x.y@nirgendwo.de](mailto:x.y@nirgendwo.de)

#### **Ausbildung**

##### **Datum**

##### **Ort**

1990 ► Theodor Gymnasium

2000 [Duale Hochschule Baden-Württemberg Mannheim \(DHBW\)](#)

#### **Ehrenamtliche Tätigkeiten**

- DRK
- Messdiener

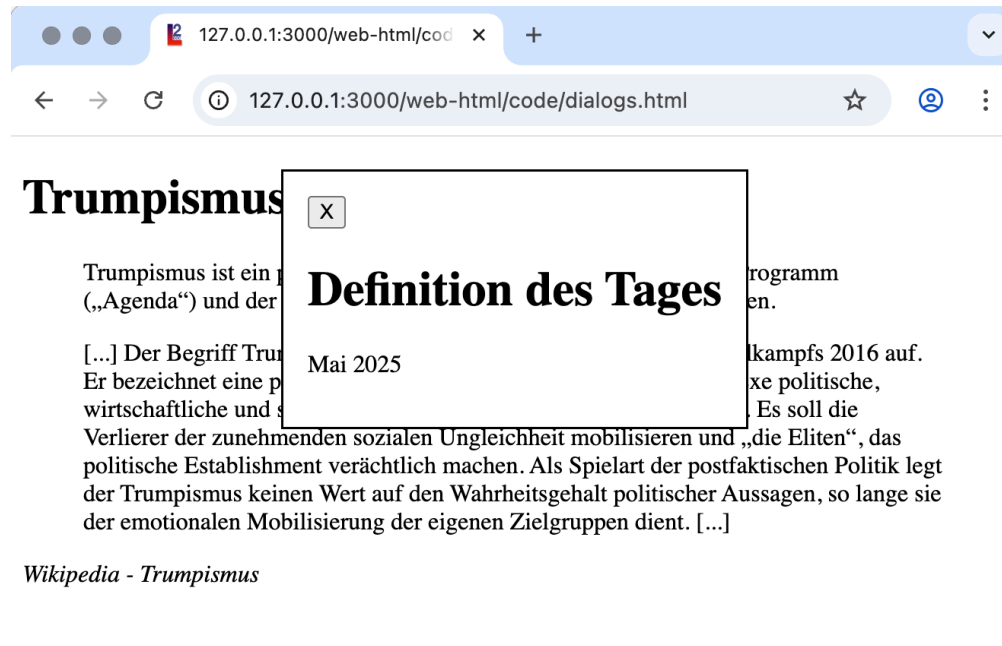
Seite 1 von 1 - Version: 2024

# Übung

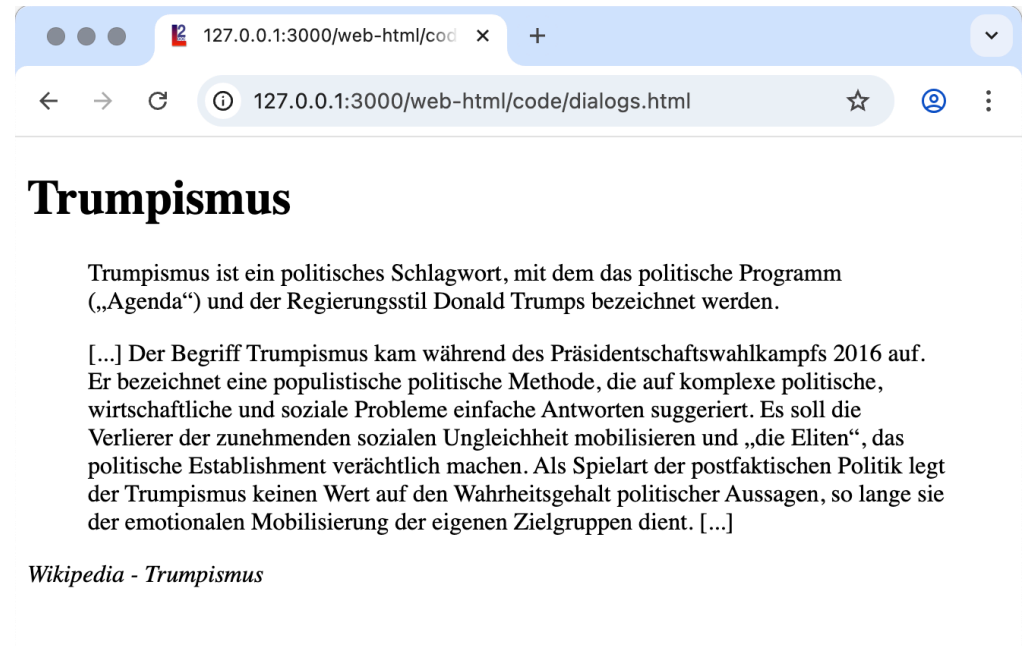
## 2.2. HTML Dialog

Erstellen Sie folgende Webseite (ohne Verwendung von CSS und/oder JavaScript):

### Nach dem Laden der Seite



### Nach dem Schließen des Dialogs



Verwenden Sie das `<dialog>` Element, um den Dialog zu erstellen. Damit der Dialog direkt beim Laden geöffnet ist, verwenden Sie das boolsche Attribut `open`. Für den Schließen-Button verwenden Sie ein `<form>` Element mit dem Attribut `method="dialog"` und ein Button-element (z. B. `<form method="dialog"><button>X</button></form>`).

Sie können das `closedby` Attribut verwenden (see [WhatWg closedby attribute](#)), um den Dialog zu schließen, wenn der Benutzer außerhalb des Dialogs klickt.

Die Definition von Trumpismus finden Sie hier: [Wikipedia - Trumpismus](#); Sie können auch jegliche andere Definition verwenden.