

WEB-TECHNOLOGIEN

C: *HTML*

01: EINFÜHRUNG UND GESCHICHTE

THEMEN DER VERANSTALTUNG



LERNZIEL

Die Grundlagen von HTML verstehen und anwenden können

HYPertext Markup Language (HTML)

- Textbasierte Auszeichnungssprache (*markup language*) zur Beschreibung von Hypertext-Dokumenten
- HTML-Dokumente werden im WWW von Browsern dargestellt
- Erinnerung: 1990 von Tim Berners-Lee entwickelt

HTML: GESCHICHTE



1990-1993

- 1990 entwickelt Tim Berners-Lee die erste Version von HTML, basierend auf der **Standard Generalized Markup Language (SGML)**
- 1993 erscheinen zwei Drafts bei der IETF: *HTML* und *HTML+*

HTML: GESCHICHTE

NP



1994

- Berners-Lee gründet das **World Wide Web Consortium** ([W3C ↗](#))
- Ziel: Definition gemeinsamer Web-Standards
- Weitere Beispiele von W3C-Spezifikationen: **Document Object Model** (DOM), **Extensible Markup Language (XML)**, **Scalable Vector Graphics (SVG)**

HTML: GESCHICHTE



1995

- Die IETF veröffentlicht HTML 2.0 ([RFC 1866 ↗](#))
- Die weitere Entwicklung findet jedoch unter Schirmherrschaft des W3C statt

HTML: GESCHICHTE



1997

- Januar: W3C veröffentlicht HTML 3.2
- Dezember: W3C veröffentlicht HTML 4.0

HTML: GESCHICHTE



1998

- W3C startet die Arbeit an XHTML
- Grundidee: HTML sollte auf Basis von XML neu spezifiziert werden
- Da XML strengeren Regeln unterliegt, sollte so z.B. eine vereinfachte Validierung ermöglicht werden

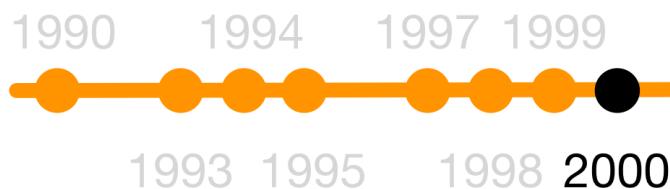
HTML: GESCHICHTE



1999

W3C veröffentlicht HTML 4.01

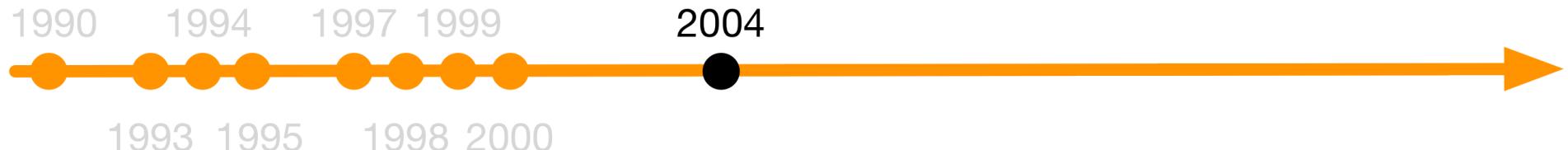
HTML: GESCHICHTE



2000

- W3C veröffentlicht XHTML 1.0
- Im Folgenden startet die Arbeit an XHTML 2.0 (inkompatibel zu HTML und XHTML 1.0)

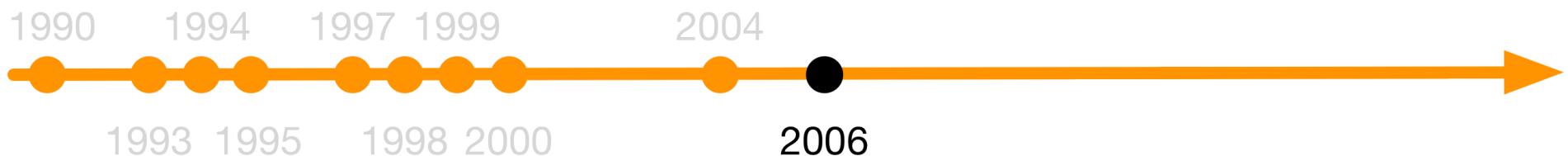
HTML: GESCHICHTE



2004

- Browser-Hersteller sind frustriert aufgrund des Stillstandes von HTML und der Langsamkeit der Prozesse im W3C
- Mozilla, Opera und Apple gründen die **Web Hypertext Application Technology Working Group** ([WHATWG](#))
- Ziel: Schnellere Weiterentwicklung von HTML und neuen Web-Standards unter dem Sammelbegriff **HTML5**

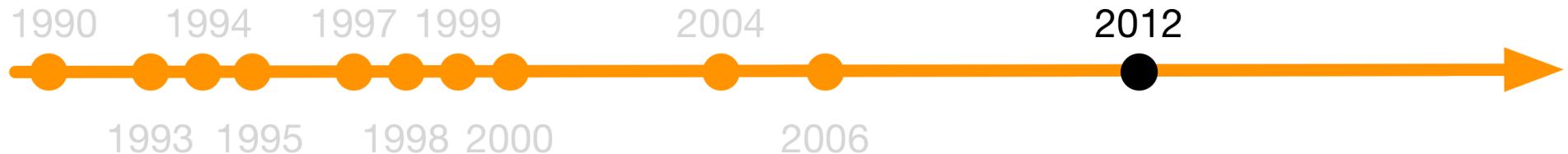
HTML: GESCHICHTE



2006

- W3C entschließt sich, WHATWG bei der Entwicklung von HTML5 zu unterstützen (im Rahmen einer Arbeitsgruppe)
- (2009 stellt das W3C sogar XHTML 2.0 zugunsten von HTML5 ein)

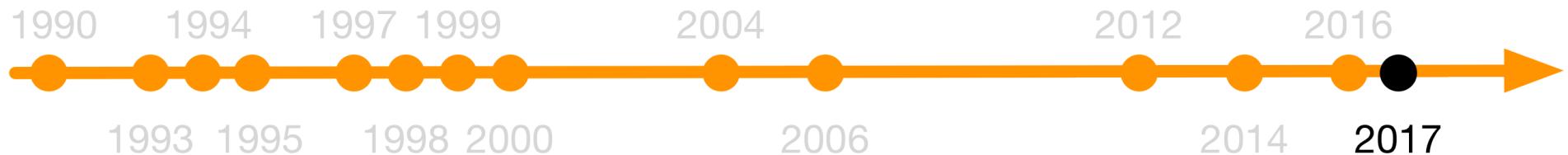
HTML: GESCHICHTE



2012

- W3C und WHATWG gehen wegen unterschiedlicher Ziele wieder "getrennte Wege"
- WHATWG führt HTML als [lebenden Standard ↗](#) (ohne Versionierung) weiter
- W3C möchte weiterhin fixe Standards mit festen Versionen veröffentlichen (gezieltes Übernehmen von Neuerungen aus der WHATWG-Spezifikation)

HTML: GESCHICHTE

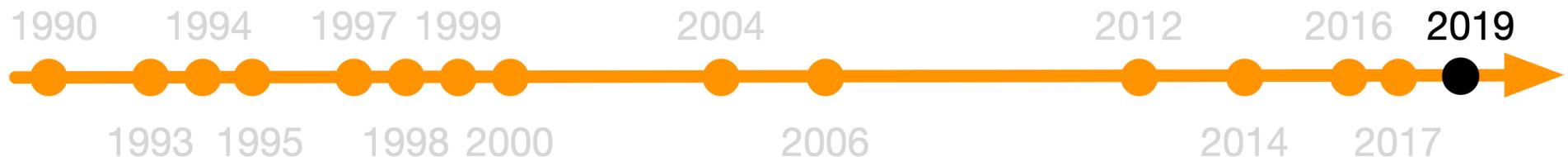


2014: W3C veröffentlicht HTML5

2016: W3C veröffentlicht [HTML 5.1 ↗](#)

2017: W3C veröffentlicht [HTML 5.2 ↗](#)

HTML: GESCHICHTE



2019

- W3C und WHATWG beschließen erneut eine Zusammenarbeit ↗
- HTML wird nur noch im lebenden Standard der WHATWG spezifiziert
- W3C wird keine weiteren HTML-Standards oder -Versionen veröffentlichen

KOMPATIBILITÄT UND PORTABILITÄT

Ist Feature X schon "offiziell"?

Unterstützt Browser Y Feature X?

Ab welcher Version unterstützt Browser Y Feature X?

Welche Features benötigt meine Anwendung?

Welche Browser muss meine Anwendung unterstützen?

! Hilfestellung: [Can I Use...↗](#)

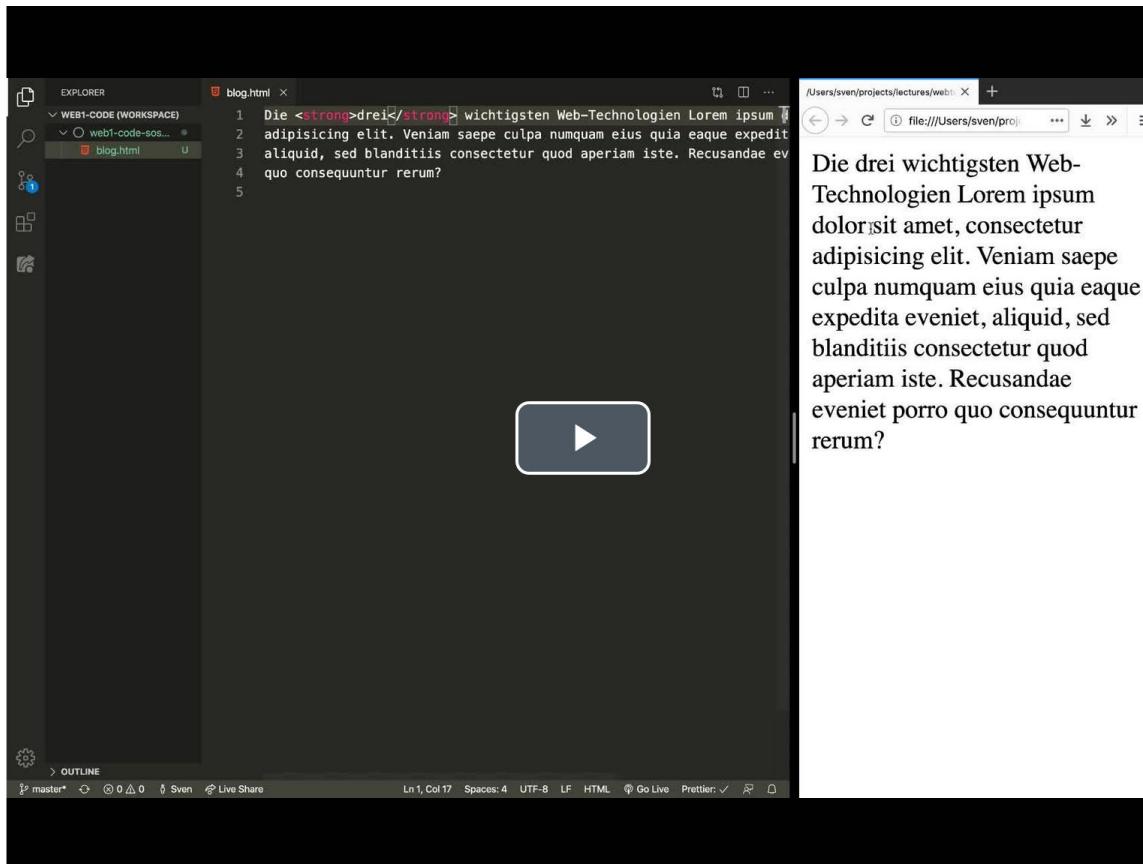
WEB-TECHNOLOGIEN

C: HTML

02: GRUNDELEMENTE

HTML-GRUNDELEMENTE

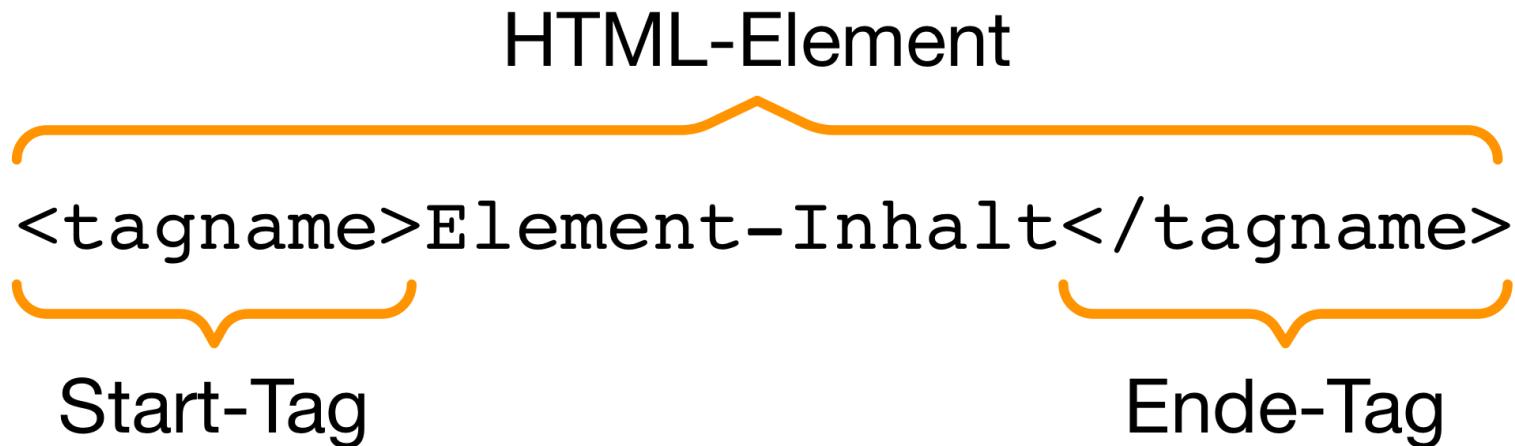
Video in ILIAS:



Quellcode zum Video: [GitLab](#) ↗

HTML: GRUNDELEMENTE

- Als Auszeichnungssprache dient HTML der semantischen (nicht visuellen!) Strukturierung und Anreicherung von Inhalten
- Die Auszeichnung erfolgt über **HTML-Elemente**



TAGS

- Dienen der Markierung von HTML-Elementen
- Treten bei den meisten HTML-Elementen paarweise auf (Start-Tag + Ende-Tag):

```
<p>Dies ist ein Absatz mit Text (p = "Paragraph")</p>
```

- Manche HTML-Elemente haben keinen Inhalt (*standalone tags*, *void tags*) und bestehen nur aus einem Tag:

```
<br> <!-- Ein Zeilenumbruch (br = "Break") -->
<br /> <!-- Ein Zeilenumbruch (XHTML-konform) -->
```

KOMMENTARE

- Kommentare in HTML starten mit `<!--` und enden mit `-->`

```
<!-- Ich bin ein Kommentar. -->
```

- Werden im Browser nicht dargestellt (sind aber dennoch im Quelltext einsehbar!)

VERSCHACHTELTE HTML-ELEMENTE

- HTML-Elemente können verschachtelt werden:

```
<p>Absatz mit <em>betontem Text</em> (em = "Emphasis")</p>
```

! Tags in umgekehrter Reihenfolge schließen!

- Durch die Verschachtelung entsteht eine hierarchische Struktur

TAG-NAMEN

- Tag-Namen können in HTML5 in Großbuchstaben (*uppercase tags*) oder Kleinbuchstaben (*lowercase tags*) geschrieben werden:

```
<p>GROß oder klein, ist doch <EM>egal</em></P>
```

- in XHTML sind Tag-Namen jedoch nur in Kleinbuchstaben erlaubt



Best Practice: Lowercase Tags verwenden!

ATTRIBUTE

- HTML-Elemente können mit **Attributen** versehen werden
- Attribute geben zusätzliche Informationen zu einem HTML-Element
- Attribute werden am Start-Tag (oder an *standalone tags*) notiert

ATTRIBUTE: SYNTAX

- Typischerweise als Paar aus Attributname und Wert angegeben (`attributname="wert"`):

```
<p title="Titel zum Absatz">  
    Inhalt von "title" wird als Tooltip des Absatzes angezeigt  
</p>
```

```
<!-- Standalone Tag zum Einfügen eines Bildes (img = "Image") -->  

```

- Manche Attribute kommen auch ohne Angabe eines Wertes aus:

```
<p hidden>Ein nicht sichtbarer Absatztext</p>
```

ATTRIBUTE: SYNTAX (2)

- Wie Tag-Namen können Attributnamen in Groß- oder Kleinbuchstaben geschrieben werden
- Attributwerte können unterschiedlich notiert werden:

```
<p title="Titel">Mit doppelten Anführungszeichen (empfohlen)</p>
```

```
<p title='Titel'>Mit einfachen Anführungszeichen</p>
```

```
<p title='Ein "guter" Titel'>Mischform</p>
```

```
<p title=Titel>  
    Wert ohne Anführungszeichen (nur möglich, wenn der Wert  
    keines der folgenden Zeichen enthält: Leerzeichen " ' = < > ` )  
</p>
```

ATTRIBUTE: SYNTAX (3)

! Best Practices:

- Attributnamen in Kleinbuchstaben schreiben
- Attributwerte immer in Anführungszeichen setzen

UNIVERSALATTRIBUTE

- Die meisten Attribute sind elementspezifisch
- Es gibt jedoch ein paar Attribute, die für die meisten HTML-Elemente verwendet werden können (**Universalattribute**)

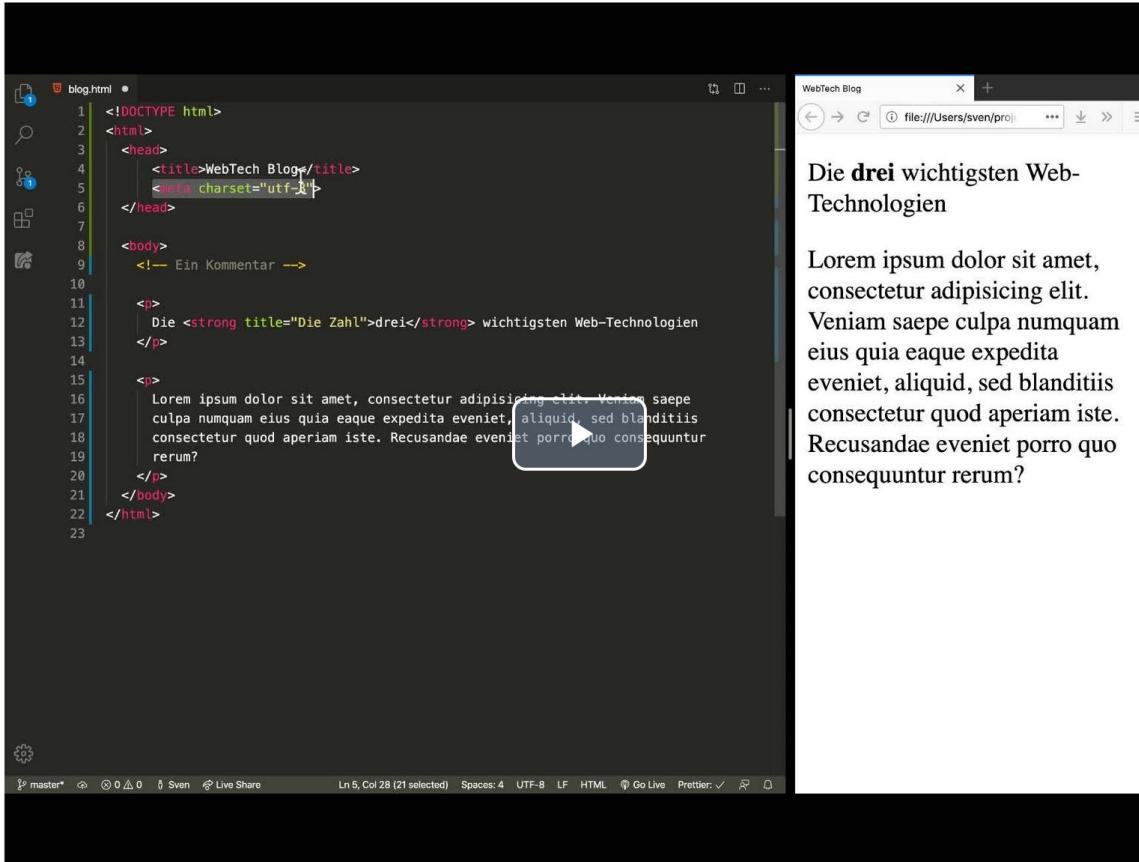
UNIVERSALATTRIBUTE: BEISPIELE

Attribut	Beschreibung
id	Eindeutige Identifizierung eines HTML-Elementes, Wert darf nur einmal im gesamten HTML-Dokument vorkommen
class, style	Attribute für die Formatierung mit CSS*
title	Kurzer Beschreibungstext für ein Element, wird häufig als Tooltip angezeigt
hidden	Markiert ein HTML-Element als nicht mehr relevant, es wird vom Browser nicht dargestellt

* Später mehr dazu!

HTML-GRUNDSTRUKTUR

Video in ILIAS:



The screenshot shows a code editor on the left and a web browser on the right. The code editor displays the file `blog.html` with the following content:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>WebTech Blog</title>
5     <meta charset="utf-8">
6   </head>
7
8   <body>
9     <!-- Ein Kommentar -->
10
11    <p>
12      | Die <strong title="Die Zahl">drei</strong> wichtigsten Web-Technologien
13    </p>
14
15    <p>
16      | Lorem ipsum dolor sit amet, consectetur adipisicing elit. Veniam saepe
17      | culpa numquam eius quia eaque expedita eveniet, aliquid sed blanditiis
18      | consectetur quod aperiam iste. Recusandae eveniet porro quo consequuntur
19      | rerum?
20    </p>
21  </body>
22</html>
```

The browser window shows the rendered HTML with the heading "WebTech Blog" and the text "Die drei wichtigsten Web-Technologien". Below it is a large block of placeholder text from the Latin epic poem *Genitivus*.

Quellcode zum Video: [GitLab](#) ↗

GRUNDSTRUKTUR EINES HTML-DOKUMENTS

```
1 <!DOCTYPE html>
2 <html>
3
4     <head>
5         <title>Titel meiner Web-Seite</title>
6         <meta charset="utf-8">
7     </head>
8
9     <body>
10        Mein sichtbarer Inhalt
11    </body>
12
13 </html>
```

Best Practice: Jedes HTML-Dokument sollte der oben stehenden Grundstruktur folgen

GRUNDSTRUKTUR EINES HTML-DOKUMENTS

```
1 <!DOCTYPE html> <!-- Definition des Dokumenttyps -->
2 <html>
3
4     <head>
5         <title>Titel meiner Web-Seite</title>
6         <meta charset="utf-8">
7     </head>
8
9     <body>
10        Mein sichtbarer Inhalt
11    </body>
12
13 </html>
```

DOKUMENTTYP

- Mittels `<!DOCTYPE>` wird der Dokumenttyp festgelegt
- Bei früheren (X)HTML-Versionen musste hier eine Document Type Definition ([DTD ↗](#)) angegeben werden
- HTML5 basiert nicht mehr auf SGML, daher wird hier keine DTD benötigt - es reicht `<!DOCTYPE html>`

BEISPIELE: DOKUMENTTYPEN VOR HTML5

NP

HTML 4.01 Strict

Erlaubte keine veralteten (*deprecated*) Elemente

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  
      "http://www.w3.org/TR/html4/strict.dtd">
```

HTML 4.01 Transitional

Erlaubte auch veraltete Elemente

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
      "http://www.w3.org/TR/html4/loose.dtd">
```

GRUNDSTRUKTUR EINES HTML-DOKUMENTS

```
1 <!DOCTYPE html>
2 <html> <!-- Wurzelement jedes HTML-Dokuments -->
3
4     <head>
5         <title>Titel meiner Web-Seite</title>
6         <meta charset="utf-8">
7     </head>
8
9     <body>
10        Mein sichtbarer Inhalt
11    </body>
12
13 </html>
```

GRUNDSTRUKTUR EINES HTML-DOKUMENTS

```
1 <!DOCTYPE html>
2 <html>
3     <!-- Kopfdaten des Dokuments, werden nicht dargestellt -->
4     <head>
5         <title>Titel meiner Web-Seite</title>
6         <meta charset="utf-8">
7     </head>
8
9     <body>
10        Mein sichtbarer Inhalt
11    </body>
12
13 </html>
```

GRUNDSTRUKTUR EINES HTML-DOKUMENTS

```
1 <!DOCTYPE html>
2 <html>
3
4     <head>
5         <title>Titel meiner Web-Seite</title>
6         <meta charset="utf-8">
7     </head>
8     <!-- Sichtbarer Bereich: Im Browser dargestellte Inhalte -->
9     <body>
10        Mein sichtbarer Inhalt
11    </body>
12
13 </html>
```

OPTIONALE TAGS

- Bestimmte Tags sind optional und dürfen daher weggelassen werden, z.B.:
 - Tags aus der Grundstruktur: <html>, </html>, <head>, </head>, <body>, </body>
 - Manche Ende-Tags, z.B. , </p>
- ! Best Practice: Optionale Tags *nicht weglassen* (Code-Lesbarkeit, Vermeidung von Fehlern)!

VALIDES HTML

- Browser sind oft sehr "gnädig" und stellen auch falsches HTML problemlos dar (z.B. falsch verschachtelte Tags)
- Tipp: Besser nicht auf den Browser verlassen → HTML-Code regelmäßig mit dem [W3C Markup Validation Service ↗](#) überprüfen

SICHTBARER BEREICH EINES HTML-DOKUMENTS

Elemente innerhalb des body-Elements:

- Elemente zur Textauszeichnung
- Elemente zur Textstrukturierung
- Sektionselemente
- Hyperlinks
- Tabellen
- Multimedia
- Formulare

SYNTAX VS. SEMANTIK

Syntax

Form und Struktur einer Sprache,

z.B. HTML: *Tags haben die Form <Tag>Inhalt</Tag>*

Semantik

Bedeutung von Wörtern, Sätzen, etc., einer Sprache

z.B. HTML: *Das body-Element enthält den sichtbaren Teil des HTML-Dokuments.*



HTML-Elemente geben zusätzliche **semantische Informationen!**

WEB-TECHNOLOGIEN

C: HTML

03: ELEMENTE ZUR TEXTAUSZEICHNUNG

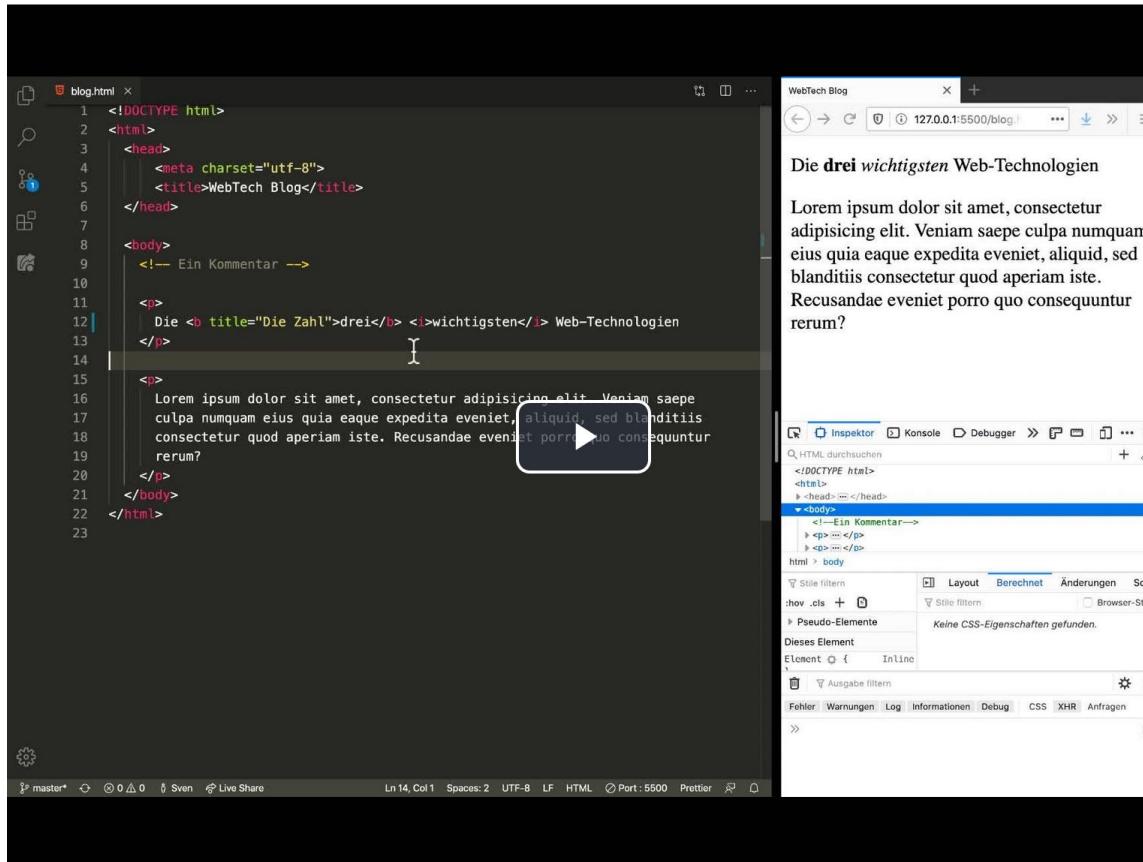
SICHTBARER BEREICH EINES HTML-DOKUMENTS

Elemente innerhalb des body-Elements:

- Elemente zur Textauszeichnung
- Elemente zur Textstrukturierung
- Sektionselemente
- Hyperlinks
- Tabellen
- Multimedia
- Formulare

ELEMENTE ZUR TEXTAUSZEICHNUNG

Video in ILIAS:



Quellcode zum Video: [GitLab](#) ↗

ELEMENTE ZUR TEXTAUSZEICHNUNG

- Reichern Buchstaben, Wörter oder Sätze im Fließtext um weitere Semantik an
- Auch *Inline-Elemente* genannt, da sie
 - keinen Zeilenumbruch bzw. neuen Absatz erzeugen
 - keinen zusätzlichen horizontalen Platz einnehmen
- Inline-Elemente dürfen nur Text und/oder andere Inline-Elemente enthalten

TEXTAUSZEICHNUNG: BEISPIELE*

* (links = HTML-Text, rechts = wie im Browser dargestellt)

Besondere Betonung (beim Sprechen):

Beim Sprechen
`besonders` betont

Beim Sprechen *besonders* betont.

Besondere Betonung (inhaltlich):

`Inhaltlich`
besonders wichtig!

Inhaltlich besonders wichtig!

TEXTAUSZEICHNUNG: BEISPIELE (2)

NP

Auszeichnen von Fachausdrücken, fremdsprachigen Wörtern, etc.:

```
HTML ist <i>très magnifique</i>
```

HTML ist *très magnifique*.

Auszeichnen von Schlüsselwörtern, Produktnamen, etc.:

```
HTML ist eine  
<b>Auszeichnungssprache</b>.
```

HTML ist eine
Auszeichnungssprache.

! Früher zum Markieren von kursivem (i = italic) oder fett gedrucktem Text (b = bold) verwendet, mit HTML5 dann "semantisch umdefiniert". Tipp: Eher sparsam einsetzen!

TEXTAUSZEICHNUNG: BEISPIELE (3)

NP

Markierter Text:

Dieser <mark>Text ist markiert</mark>.

Dieser Text ist markiert.

Kleingedrucktes:

<small>Ihre Daten gehören uns.</small>

Ihre Daten gehören uns.

Zitat oder wörtliche Rede (q = quotation):

Sie sagte es sei <q>nicht relevant</q>.

Sie sagte es sei „nicht relevant“.

TEXTAUSZEICHNUNG: BEISPIELE (4)

NP

Abkürzungen oder Akronyme:

```
<abbr title="Hypertext Markup  
Language">HTML</abbr> ist  
nicht schwer.
```

HTML ist nicht schwer.

! Der Inhalt von "title" erscheint im Browser als Tooltip

Referenz auf ein Werk (z.B. Quellenangabe):

```
<cite>Computer Networks</cite>  
von A. Tanenbaum
```

Computer Networks von A. Tanenbaum

TEXTAUSZEICHNUNG: BEISPIELE (5)

NP

Hoch- und tiefgestellter Text:

```
Entwickler lieben  
C<sub>8</sub>H<sub>10</sub>  
N<sub>4</sub>O<sub>2</sub>  
<sup>*</sup><br><small>  
<sup>*</sup> Koffein</small>
```

Entwickler lieben C₈H₁₀N₄O₂*
* Koffein

Datums- und Zeitangaben:

```
HTML ist <time  
datetime="1991-08-06">27  
Jahre</time> alt.
```

HTML ist 27 Jahre alt.

TEXTAUSZEICHNUNG: BEISPIELE (6)

NP

Quellcode:

```
<code>a="a=%c%s%c;  
print a%%(34,a,34)";  
print a%(34,a,34)</code>
```

```
a="a=%c%s%c;print a%%  
(34,a,34)";print a%(34,a,34)
```

Variablen:

```
<var>E</var>=  
<var>m</var><var>c</var>  
<sup>2</sup>
```

$$E = m c^2$$

TEXTAUSZEICHNUNG: `span`

- Das `span`-Element dient der Auszeichnung im Fließtext
- Angewendet, wenn die übrigen Elemente nicht passen
- Hauptsächlich für die Formatierung mittels CSS und Selektion mittels JavaScript verwendet*

* Später mehr dazu!

WEB-TECHNOLOGIEN

C: HTML

04: ELEMENTE ZUR TEXTSTRUKTURIERUNG

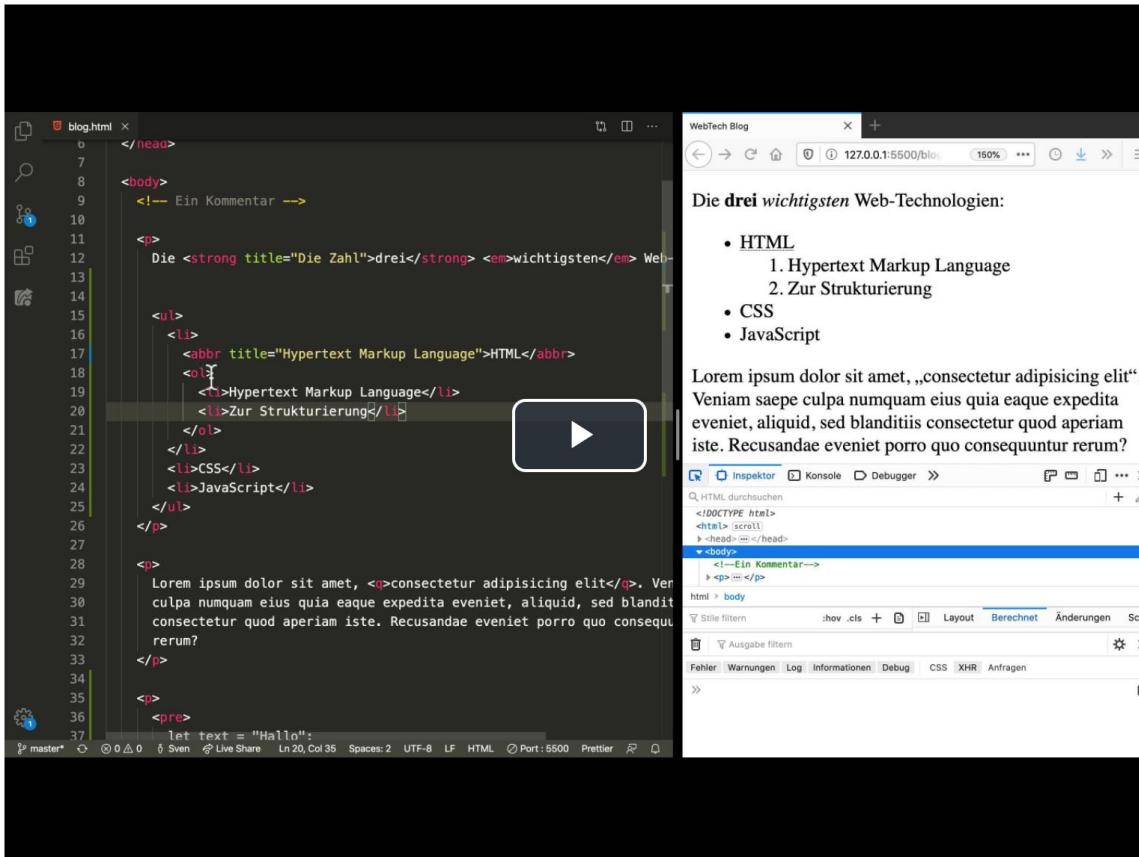
SICHTBARER BEREICH EINES HTML-DOKUMENTS

Elemente innerhalb des body-Elements:

- Elemente zur Textauszeichnung
- Elemente zur Textstrukturierung
- Sektionselemente
- Hyperlinks
- Tabellen
- Multimedia
- Formulare

ELEMENTE ZUR TEXTSTRUKTURIERUNG

Video in ILIAS:



Quellcode zum Video: [GitLab](#) ↗

ELEMENTE ZUR TEXTSTRUKTURIERUNG

- Dienen der Strukturierung bzw. Gruppierung von Textinhalten
- Auch *Block-Elemente* genannt, da sie
 - typischerweise vor und nach dem Element einen Zeilenumbruch erwirken
 - zumeist horizontal soviel Platz wie möglich einnehmen
- Block-Elemente können Text, Inline-Elemente und andere Block-Elemente enthalten*

* Inline-Elemente dürfen in der Regel jedoch keine Block-Elemente enthalten (Ausnahme: a)!

TEXTSTRUKTURIERUNG: BEISPIELE

Absatz (p = paragraph):

```
<p>Ein Absatz.</p>
<p>Und noch einer.</p>
```

Ein Absatz.

Und noch einer.

Thematische Trennung
(hr = horizontal rule):

```
<p>Ein Absatz.</p><hr>
<p>Und noch einer.</p>
```

Ein Absatz.

Und noch einer.

TEXTSTRUKTURIERUNG: BEISPIELE (2)

Manueller Zeilenumbruch (br = break, Inline-Element):

Dieser Satz ist
manuell umgebrochen.

Dieser Satz ist
manuell umgebrochen.

! Normalerweise entscheidet der Browser, wann umgebrochen wird.

Optionaler Zeilenumbruch im Wort (wbr = word break, Inline-Element):

Kraftfahrzeug<wbr>haftpflicht
<wbr>versicherungs<wbr>
vertreter

Kraftfahrzeughhaftpflichtversicherungs
vertreter

TEXTSTRUKTURIERUNG: BEISPIELE (3)

Mehrzeiliges Zitat:

```
<blockquote cite="Wau Holland">  
    Alles ist eins - außer  
    der Null.  
</blockquote>
```

Alles ist eins - außer der Null.

Mehrzeiliger Quellcode (pre = preformatted):

```
<pre>  
(function() {  
    document.querySelectorAll(".pdf-remove")  
        .forEach(function(elem) {  
            elem.remove();  
        });  
})();  
</pre>
```

```
(function() {  
    document.querySelectorAll(".pdf-remove")  
        .forEach(function(elem) {  
            elem.remove();  
        });  
})();
```

! Bei `pre` bleiben Umbrüche, Leerzeichen, etc. erhalten.

TEXTSTRUKTURIERUNG: BEISPIELE (4)

Ungeordnete Aufzählung
(ul = unordered list, li= list item):

```
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>
```

- HTML
- CSS
- JavaScript

Geordnete Aufzählung (ol = ordered list):

```
<ol>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ol>
```

1. HTML
2. CSS
3. JavaScript

TEXTSTRUKTURIERUNG: BEISPIELE (5)

Beschreibungsliste (dl = description list, dt = description term, dd = description):

```
<dl>
  <dt>HTML-Element "dl"</dt>
  <dd>
    Früher: "Definition List".
  </dd>
  <dt>HTML-Element "dt"</dt>
  <dt>HTML-Element "dd"</dt>
  <dd>
    Früher: "Definition Term",
    "Definition Description".
  </dd>
</dl>
```

HTML-Element "dl"

Früher: "Definition List".

HTML-Element "dt"

HTML-Element "dd"

Früher: "Definition Term",
"Definition Description".

TEXTSTRUKTURIERUNG: `div`

- Analog zum `span`-Element, allerdings als Block
- Hauptsächlich für die Formatierung mittels CSS und Selektion mittels JavaScript verwendet*

* Später mehr dazu!

WEB-TECHNOLOGIEN

C: HTML

05: SEKTIONSELEMENTE

SICHTBARER BEREICH EINES HTML-DOKUMENTS

Elemente innerhalb des body-Elements:

- Elemente zur Textauszeichnung
- Elemente zur Textstrukturierung
- Sektionselemente
- Hyperlinks
- Tabellen
- Multimedia
- Formulare

SEKTIONSELEMENTE

- Dienen der semantischen Strukturierung eines HTML-Dokuments, d.h. der Aufteilung in sinnvolle Bereiche (**Sektionen**)
- Das body-Element ist die Wurzel aller Sektionen

SEKTIONSELEMENTE: ÜBERSCHRIFTEN

Überschriften (h = heading):

```
<h1>Überschrift 1</h1>
<h2>Überschrift 2</h2>
<h3>Überschrift 3</h3>
<h4>Überschrift 4</h4>
<h5>Überschrift 5</h5>
<h6>Überschrift 6</h6>
```

Überschrift 1
Überschrift 2
Überschrift 3
Überschrift 4
Überschrift 5
Überschrift 6

- Bis zu sechs Überschriftenebenen möglich
- Tags (Ebene absteigend): h1, h2, h3, h4, h5, h6

SEKTIONEN: TYPISCHER AUFBAU EINER WEBSEITE

Studieninteressierte
Studierende
Promovierende
Presse
Forschung und Transfer
IDiAL
Hochschule
Jobs & Karriere



Hochschulrat
Ministerin überrieb Ernennungskunden

Tag der offenen Tür
Besuchen Sie uns am 5. Mai!

Spring School
Studierende bauen kulturelle Brücken

TV-Projekt
KOMM AN! Der Migrations-Talk auf nrwision

Neubau
Lernen im Lichthof

TalentKolleg Ruhr
Neue App „Mathematik Vorkurs“ gestartet

Forschungsfrühstück
Einladung ins Rathaus

Studieninteressierte
Freie Plätze in Master-Studiengängen

IT-Mittelstandsstipendium
Förderung für 35 neue Stipendiat*innen

Dortmunder Wirtschaftsprisie
FH-Alumnus als Unternehmer erfolgreich

News
Weitere aktuelle Beiträge



Eingeloggt als Sven Jörges.

**TAG DER
OFFENEN TÜR
5. MAI 2018**

BildungsOffensive

Erstsemester

FHDO hilft

Offene Fachhochschule

Studienplatzportal

FH DORTMUND

Über uns
Campus Emil-Figge-Straße
Campus Max-Ophüls-Platz
Campus Sonnenstraße
Jobs & Karriere

BILDUNGSANGEBOTE

Bachelor Studium
Master Studium
Duales Studium
Berufsbegleitendes Studium

SERVICE

Studienfinanzierung
Studien- und Prüfungsordnung
IT-Dienste
Redaktion
Hochschule intern

KONTAKT

Personen
Presse
VIA* Feedback
Ersthelfer
Impressum & Datenschutz

SOCIAL MEDIA



**Fachhochschule
Dortmund**
University of Applied Sciences and Arts

we focus on students

Kopfbereich

Suche
Schnellzugriff

G Sprache auswählen | ▾

Navigationsmenü

- Startseite
- Architektur
- Design
- Elektrotechnik
- Informatik
- Wirtschaftsingenieurwissenschaften
- Wirtschaft
- Informationstechnik

Studieninteressierte

Studierende

Promovierende

Presse

Forschung und Transfer

IDiAL

Hochschule

Jobs & Karriere

Hauptinhalt

Hochschulrat
Ministerin überreichte Ernennungsurkunden
Tag der offenen Tür
Besuchen Sie uns am 5. Mai
Spring School
Studierende bauen kulturelle Brücken
TV-Projekt
KOMM AN! Der Migrations-Talk auf nrwision
Neubau
Lernen im Lichthof
TalentKolleg Ruhr
Neue App „Mathematik Vorkurs“ gestartet
Frühstück im Rathaus
Einladung ins Rathaus
Studieninteressierte
Freie Plätze in Master-Studiengängen
IT-Mittelstandsstipendium
Förderung für 35 neue Stipendiat*innen
Dortmunder Wirtschaftspris
FH-Alumnus als Unternehmer erfolgreich
News
Weitere aktuelle Beiträge

Eingeloggt als Sven Jörges.

**TAG DER
OFFENEN TÜR
5. MAI 2018**

BildungsOffensive
Erstsemester
FHDO hilft
„Zusatz-
informationen“
Studienplatzportal

FH DORTMUND

Über uns
Campus Emil-Figge-Straße
Campus Max-Ophüls-Platz
Campus Sonnenstraße
Jobs & Karriere

BILDUNGSANGEBOTE

Bachelor Studium
Master Studium
Duales Studium
Berufsbegleitendes Studium

SERVICE

Studentenfinanzierung
Studentenwerk
IT-Dienste
Redaktion
Hochschule intern

KONTAKT

Personen
Presse
VIA* Feedback
Ersthelfer
Impressum & Datenschutz

Fußbereich

SOCIAL MEDIA

SEITENSTRUKTURIERUNG VOR HTML5

NP

FRAMESETS

- Jeder Bereich der Webseite ist eine eigene Unterseite (ein **Frame**), d.h. ein separates HTML-Dokument
- Ein zusätzliches HTML-Dokument definiert, wie die Unterseiten zusammengesetzt werden (**das Frameset**)



Beispiel: [API-Dokumentation zu Java 8](#) (→ Schauen Sie sich den Quellcode der Webseite an!)

SEITENSTRUKTURIERUNG VOR HTML5

FRAMESETS (2)

Nachteile (u.A.):

- Suchmaschinen verlinkten direkt auf Unterseiten
 - Lesezeichen funktionieren nicht wie erwartet
 - Ungeeignet für kleinere Bildschirme
- !** Im HTML5-Standard nicht mehr enthalten

SEITENSTRUKTURIERUNG VOR HTML5

HTML-TABELLEN*

* Später mehr dazu!

- Die ganze Webseite ist eine große Tabelle mit unsichtbarem Rand
- Die einzelnen Bereiche sind Zellen der Tabelle



Missbrauch von HTML-Tabellen - nicht empfohlen!

SEITENSTRUKTURIERUNG VOR HTML5

div-ELEMENTE UND CSS

```
<!DOCTYPE html>
<html>
<body>
  <div id="container">
    <div id="kopfbereich">...</div>
    <div id="navigation-kopf">...</div>
    <div id="navigation-seite">...</div>
    <div id="inhalt">...</div>
    <div id="zusatzinfos">...</div>
    <div id="fussbereich">...</div>
  </div>
</body>
</html>
```

- Bereiche werden mit div-Elementen ausgezeichnet und mit CSS formatiert
- + Gute Trennung von Struktur und Darstellung

SEITENSTRUKTURIERUNG VOR HTML5

div-ELEMENTE UND CSS (2)

```
<!DOCTYPE html>
<html>
<body>
  <div id="container">
    <div id="kopfbereich">...</div>
    <div id="navigation-kopf">...</div>
    <div id="navigation-seite">...</div>
    <div id="inhalt">...</div>
    <div id="zusatzinfos">...</div>
    <div id="fussbereich">...</div>
  </div>
</body>
</html>
```

- Jedoch: IDs der div-Elemente sind beliebig (div ist *semantisch neutral*)
- Für z.B. Browser und Screenreader nicht sinnvoll unterscheidbar
(Barrierefreiheit!)

SEKTIONSELEMENTE AB HTML5

- HTML5 führte neue Elemente zur *semantischen Strukturierung* ein
- Die Semantik der Bereiche wird so auch maschinell erfassbar, Vorteile sind z.B.
 - + Bessere Aufbereitung der Inhalte in Screenreadern
 - + Bessere Ergebnisse in Suchmaschinen
 - + Zusätzliche Browser-Funktionen wie z.B. der Lesemodus

SEKTIONSELEMENTE AB HTML5

Element	Semantik
header	Kopfbereich
nav	Navigationsbereich
main	Bereich mit dem Hauptinhalt <small>(streng genommen lediglich ein Gruppierungselement)</small>
section	(Unter-)Abschnitt eines Dokuments (z.B. Kapitel, Unterkapitel)
article	Inhaltlich in sich geschlossener Block (z.B. Blog-Artikel, Nachrichtenmeldung)
aside	Zusätzliche Informationen
footer	Fußbereich

Fachhochschule Dortmund
University of Applied Sciences and Arts

we focus on students

Kopfbereich

Suche
Schnellzugriff

G Sprache auswählen | ▾

Navigationsmenü

- Startseite
- Architektur
- Design
- Elektrotechnik
- Informatik
- Wirtschaftsingenieurwissenschaften
- Wirtschaft
- Informationstechnik

Studieninteressierte

Studierende

Promovierende

Presse

Forschung und Transfer

IDiAL

Hochschule

Jobs & Karriere

Hauptinhalt

Hochschulrat
Ministerin überrieb Ernennungsurkunden
Tag der offenen Tür
Besuchen Sie uns am 5. Mai
Spring School
Studierende bauen kulturelle Brücken
TV-Projekt
KOMM AN! Der Migrations-Talk auf nrwision
Neubau
Lernen im Lichthof
TalentKolleg Ruhr
Neue App „Mathematik Vorkurs“ gestartet
Frühstück
Einladung ins Rathaus
Studieninteressierte
Freie Plätze in Master-Studiengängen
IT-Mittelstandsstipendium
Förderung für 35 neue Stipendiat*innen
Dortmunder Wirtschaftspris
FH-Alumnus als Unternehmer erfolgreich
News
Weitere aktuelle Beiträge

Eingeloggt als Sven Jörges.

TAG DER OFFENEN TÜR
5. MAI 2018

BildungsOffensive

Erschreiber
FHDO hilft

„Zusatz-informationen“

Studienplatzportal

FH DORTMUND

Über uns
Campus Emil-Figge-Straße
Campus Max-Ophüls-Platz
Campus Sonnenstraße
Jobs & Karriere

BILDUNGSANGEBOTE

Bachelor Studium
Master Studium
Duales Studium
Berufsbegleitendes Studium

SERVICE

Studentenfinanzierung
Studentenwerk
IT-Dienste
Redaktion
Hochschule intern

KONTAKT

Personen
Presse
VIA* Feedback
Ersthelfer
Impressum & Datenschutz

Fußbereich

SOCIAL MEDIA

**Fachhochschule
Dortmund**
University of Applied Sciences and Arts

we focus on students

<header>

Suche
Schnellzugriff

G Sprache auswählen | ▾

<nav>

Startseite Architektur Design Elektrotechnik Informatik Maschinenbau Sozialwissenschaften Wirtschaft Informationstechnik

Studieninteressierte
Studierende
Promovierende
Presse
Forschung und Transfer
IDiAL
Hochschule
Jobs & Karriere

Hochschulrat
Ministerin überrieb Ernennungskunden
Tag der offenen Tür
Besuchen Sie uns am 5. Mai
Spring School
Studierende bauen kulturelle Brücken
TV-Projekt
KOMM AN! Der Migrations-Talk auf nrwision
Neubau
Lernen im Lichthof
TalentKolleg Ruhr
Neue App „Mathematik Vorkurs“ gestartet
Forschungsfrühstück
Einladung ins Rathaus
Studieninteressierte
Freie Plätze in Master-Studiengängen
IT-Mittelstandsstipendium
Förderung für 35 neue Stipendiat*innen
Dortmunder Wirtschaftspris
FH-Alumnus als Unternehmer erfolgreich
News
Weitere aktuelle Beiträge

Eingeloggt als Sven Jörges.

**TAG DER
OFFENEN TÜR
5. MAI 2018**

BildungsOffensive
Erstsemester
<aside>
Offene Fachhochschule
Studienplatzportal

<nav>

<main>

Deutschland STIPENDIUM
EUA
CODE OF CONDUCT
Dortmund seit 2001
HRK-Audit
Bildungsangebote

<footer>

FH DORTMUND
Über uns
Campus Emil-Figge-Straße
Campus Max-Ophüls-Platz
Campus Sonnenstraße
Jobs & Karriere

BILDUNGSANGEBOTE
Bachelor Studium
Master Studium
Duales Studium
Berufsbegleitendes Studium

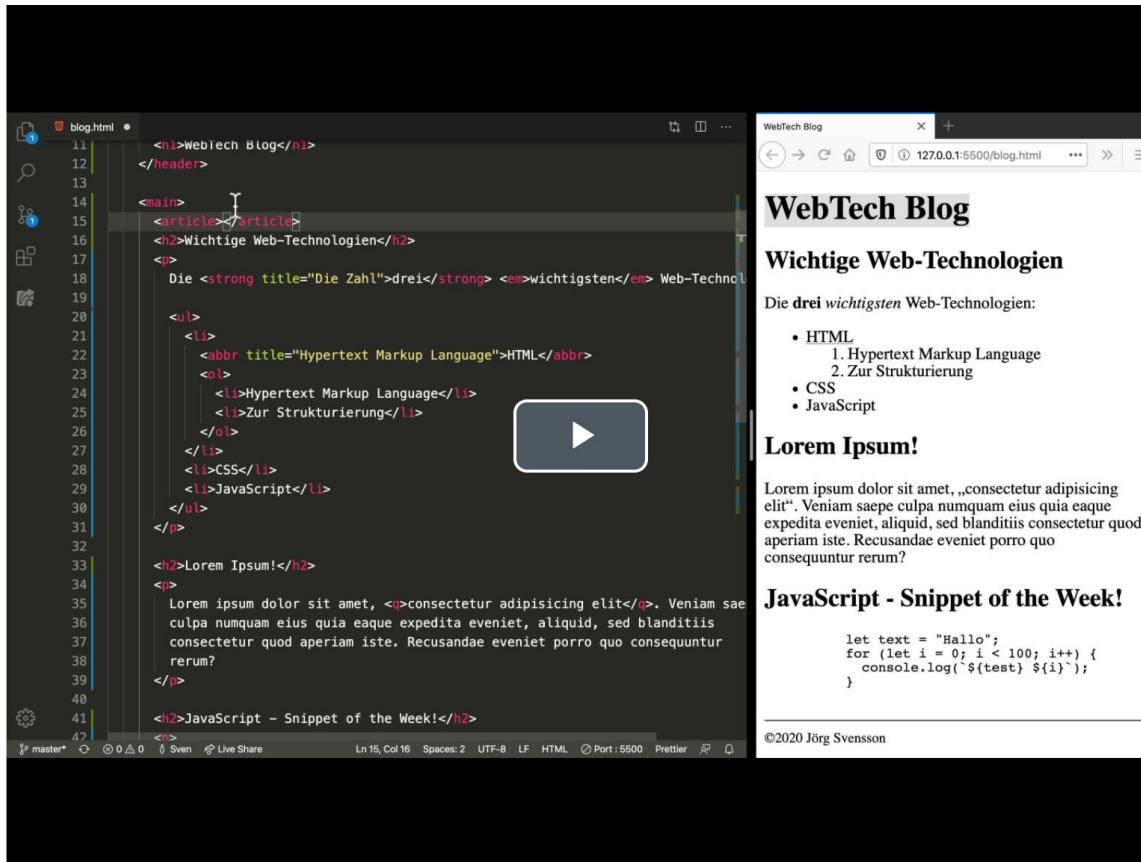
SERVICE
Studienfinanzierung
Studentenwerk NordWest
IT-Dienste
Redaktion
Hochschule intern

KONTAKT
Personen
Presse
VIA* Feedback
Ersthelfer
Impressum & Datenschutz

SOCIAL MEDIA

SEKTIONSELEMENTE UND SCREENREADER

Video in ILIAS:



Quellcode zum Video: [GitLab ↗](#)

WEB-TECHNOLOGIEN

C: HTML

06: HYPERLINKS

SICHTBARER BEREICH EINES HTML-DOKUMENTS

Elemente innerhalb des body-Elements:

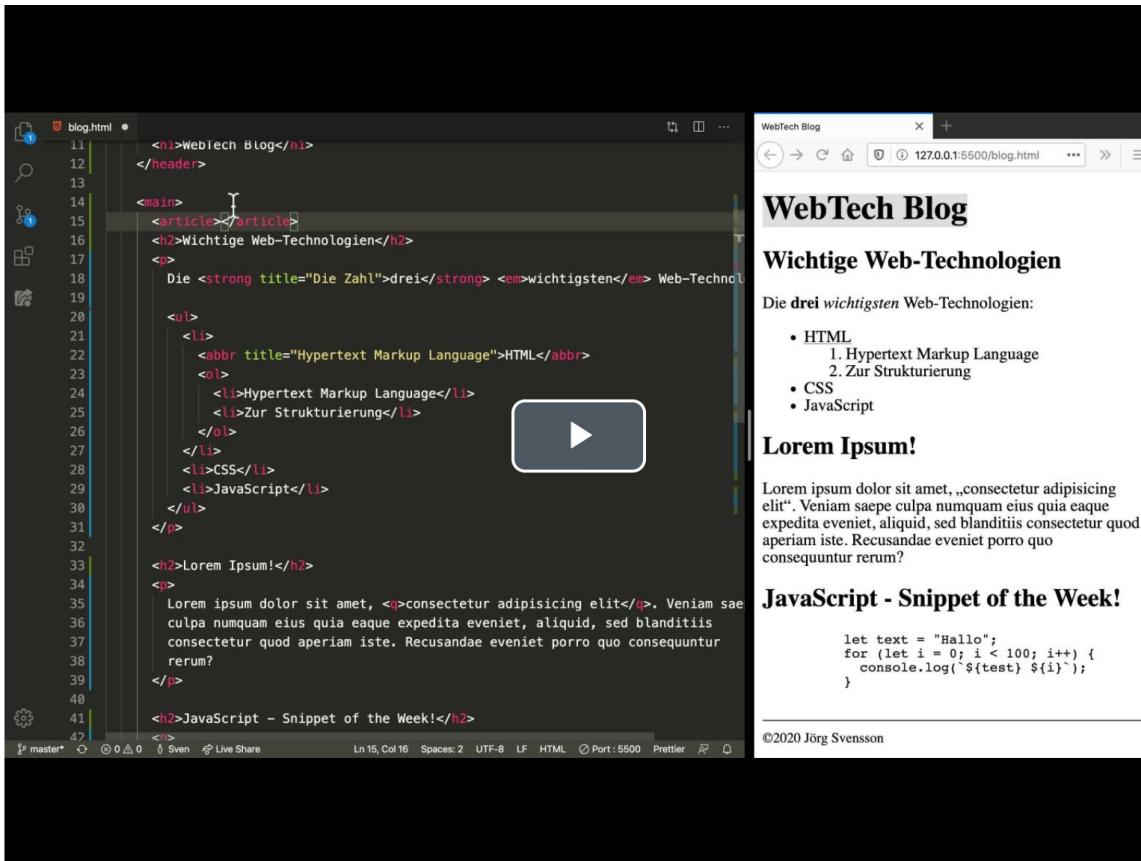
- Elemente zur Textauszeichnung
- Elemente zur Textstrukturierung
- Sektionselemente
- Hyperlinks
- Tabellen
- Multimedia
- Formulare

Hyperlinks

- Erinnerung an Themenblock B (Web-Grundlagen): „*Hypertext ist Text, welcher **Links** zu anderen Texten enthält*“
- **Hyperlinks** erlauben es, Verweise auf beliebige Inhalte zu setzen (z.B. andere HTML-Dokumente, Bilder, Videos)

VIDEO: HYPERLINKS

Video in ILIAS:



The image shows a split-screen view. On the left is a code editor displaying the file `blog.html`. The code contains HTML and some CSS-like styling. A cursor is visible over the `<article>` tag. On the right is a web browser window titled "WebTech Blog" showing the rendered content of the `blog.html` file. The browser window includes a navigation bar and a status bar at the bottom.

`blog.html` content (approximate):

```
11  <h1>WebTech Blog</h1>
12  </header>
13
14  <main>
15  <article>
16  <h2>Wichtige Web-Technologien</h2>
17  <p>
18  Die <strong title="Die Zahl">drei</strong> <em>wichtigsten</em> Web-Technologien
19
20  <ul>
21  <li>
22  <abbr title="Hypertext Markup Language">HTML</abbr>
23  <ol>
24  <li>Hypertext Markup Language</li>
25  <li>Zur Strukturierung</li>
26  </ol>
27  </li>
28  <li>CSS</li>
29  <li>JavaScript</li>
30  </ul>
31  </p>
32
33  <h2>Lorem Ipsum!</h2>
34  <p>
35  Lorem ipsum dolor sit amet, <q>consectetur adipisicing elit</q>. Veniam saepe
36  culpa numquam eius quia eaque expedita eveniet, aliquid, sed blanditiis
37  consectetur quod aperiam iste. Recusandae eveniet porro quo consequuntur
38  rerum?
39  </p>
40
41  <h2>JavaScript - Snippet of the Week!</h2>
42  <pre>
```

WebTech Blog

Wichtige Web-Technologien

Die **drei** wichtigsten Web-Technologien:

- HTML
 - 1. Hypertext Markup Language
 - 2. Zur Strukturierung
- CSS
- JavaScript

Lorem Ipsum!

Lorem ipsum dolor sit amet, „consectetur adipisicing elit“. Veniam saepe culpa numquam eius quia eaque expedita eveniet, aliquid, sed blanditiis consectetur quod aperiam iste. Recusandae eveniet porro quo consequuntur rerum?

JavaScript - Snippet of the Week!

```
let text = "Hallo";
for (let i = 0; i < 100; i++) {
  console.log(`${text} ${i}`);
}
```

©2020 Jörg Svensson

Quellcode zum Video: [GitLab](#) ↗

Hyperlinks in HTML

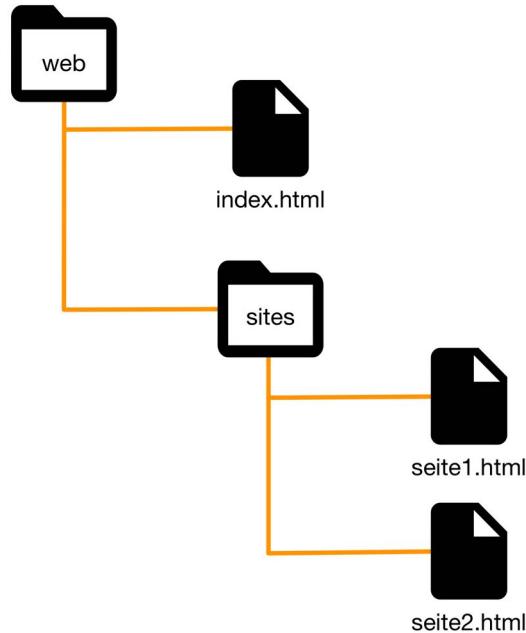
Erstellt mit dem Element a (für anchor):

```
<a href="https://www.fh-dortmund.de">  
    Webseite der FH  
</a>
```

Webseite der FH

- Über das Attribut `href` (= **hypertext reference**) wird das Ziel des Hyperlinks festgelegt (als URL)
- Der Inhalt des Elements gibt den Verweistext bzw. Inhalt (z.B. ein Bild) des Hyperlinks an

ABSOLUTE UND RELATIVE VERWEISE

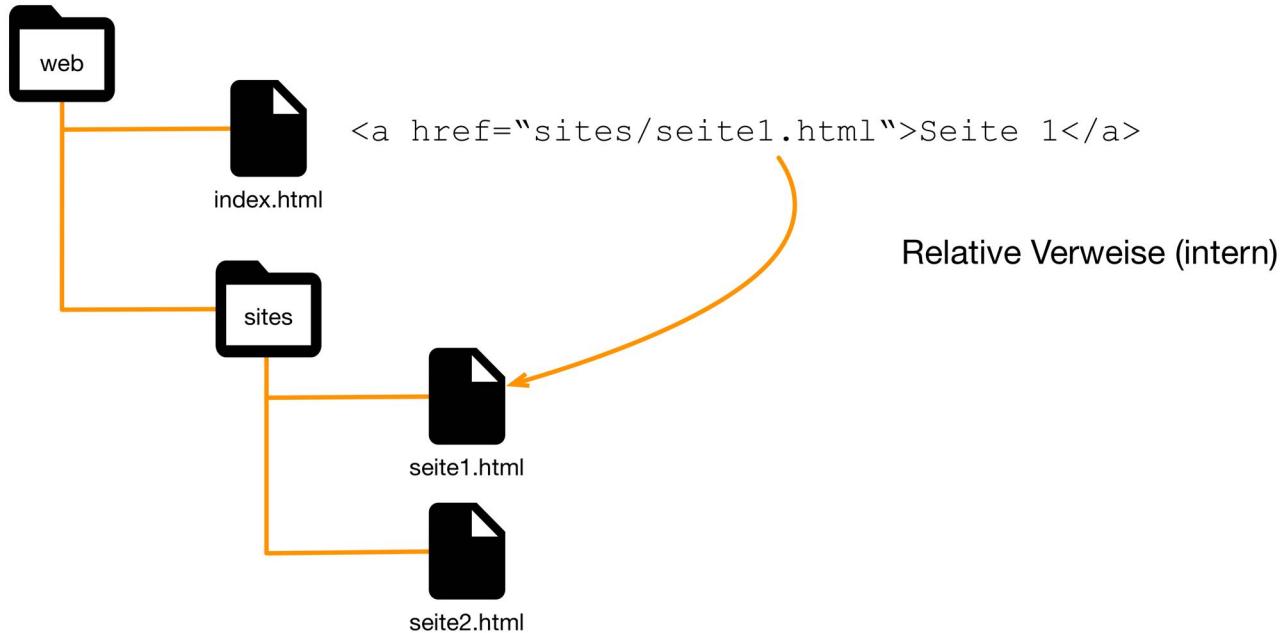


ABSOLUTE UND RELATIVE VERWEISE

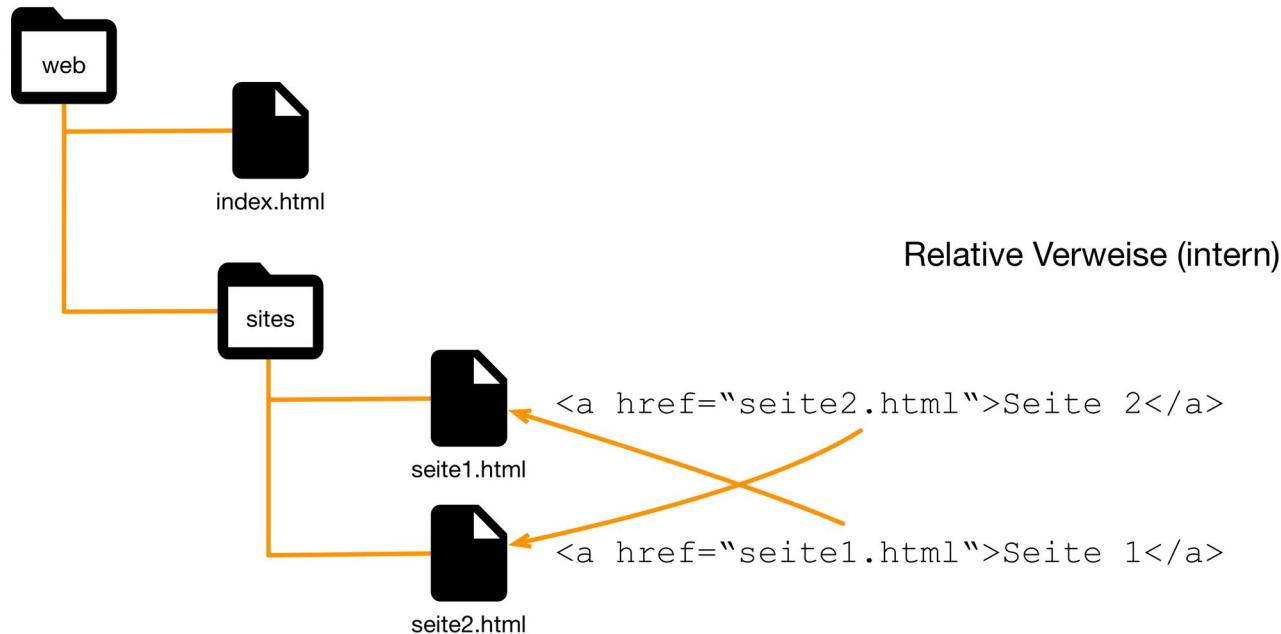
http://www.beispiel.de



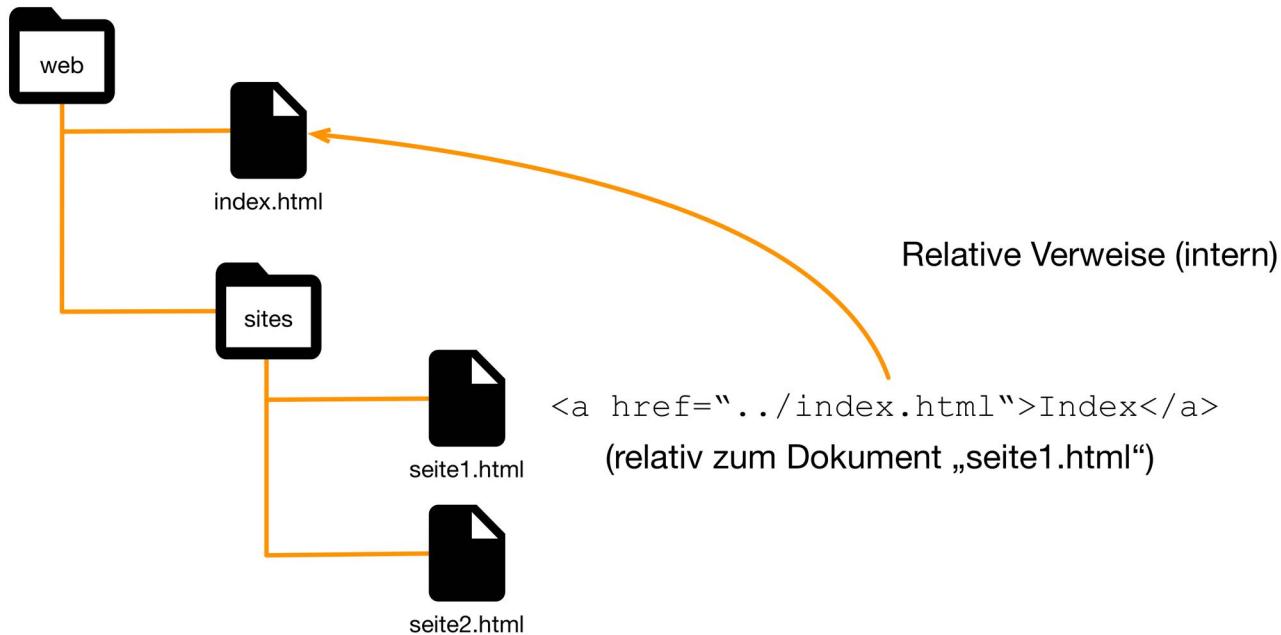
ABSOLUTE UND RELATIVE VERWEISE



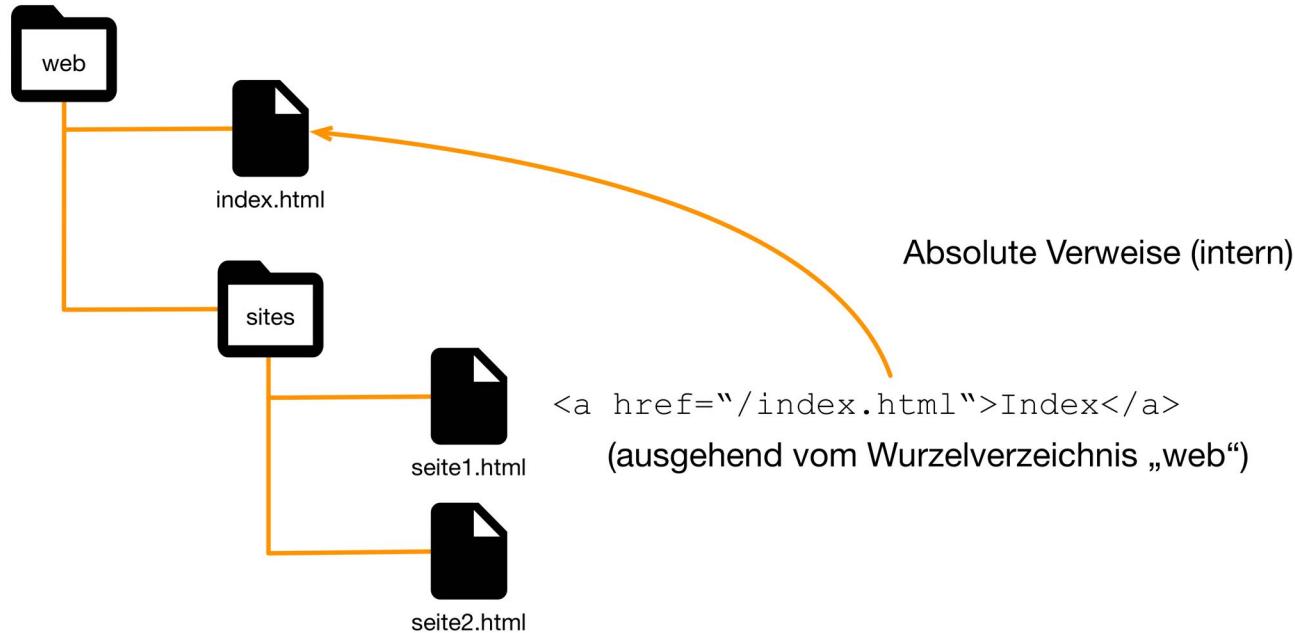
ABSOLUTE UND RELATIVE VERWEISE



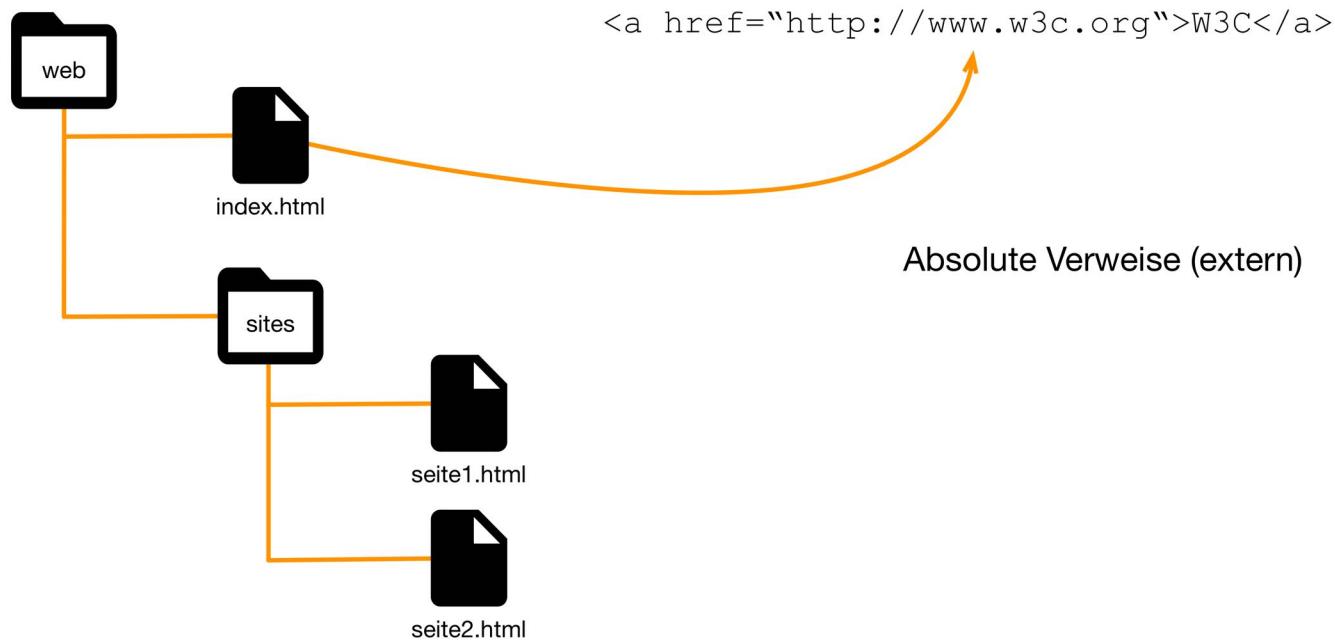
ABSOLUTE UND RELATIVE VERWEISE



ABSOLUTE UND RELATIVE VERWEISE



ABSOLUTE UND RELATIVE VERWEISE



VERWEISE INNERHALB EINES DOKUMENTS

- Z.B. zum Navigieren innerhalb langer Inhalte
- Benötigt:
 1. HTML-Element an der zu verlinkenden Stelle mit **Anker** der Form `id="ankernname"` versehen
 2. Hyperlink definieren:
- Der Anker wird über den *Fragment*-Teil der URL angesprungen:

```
<a href="#ankernname">...</a>
```

```
https://www.beispiel.de/seite.html#ankernname
```

VERWEISE INNERHALB EINES DOKUMENTS (2)

Beispiel:

```
...
<nav>
  <a href="#einleitung">Einleitung</a> |
  <a href="#analyse">Analyse</a>
</nav>
<section id="einleitung">
  <h1>Einleitung</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing...</p>
</section>
<section id="analyse">
  <h1>Analyse</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing...</p>
</section>
...
...
```

HYPERLINKS: WEITERE BEISPIELE

E-Mail-Links:

```
<a href="mailto:sven.joerges@fh-dortmund.de">Mail senden</a>
```

Download-Links und Inhaltstypen:

```
<!-- PDF bei Klick auf den Link herunterladen -->
<a href="dokument.pdf" download>PDF herunterladen</a>
```

```
<!-- Hinweis auf den Inhaltstyp geben -->
<a href="dokument.doc" type="application/msword">
    Word-Dokument
</a>
```

WEB-TECHNOLOGIEN

C: *HTML*

07: TABELLEN

SICHTBARER BEREICH EINES HTML-DOKUMENTS

Elemente innerhalb des body-Elements:

- Elemente zur Textauszeichnung
- Elemente zur Textstrukturierung
- Sektionselemente
- Hyperlinks
- Tabellen
- Multimedia
- Formulare

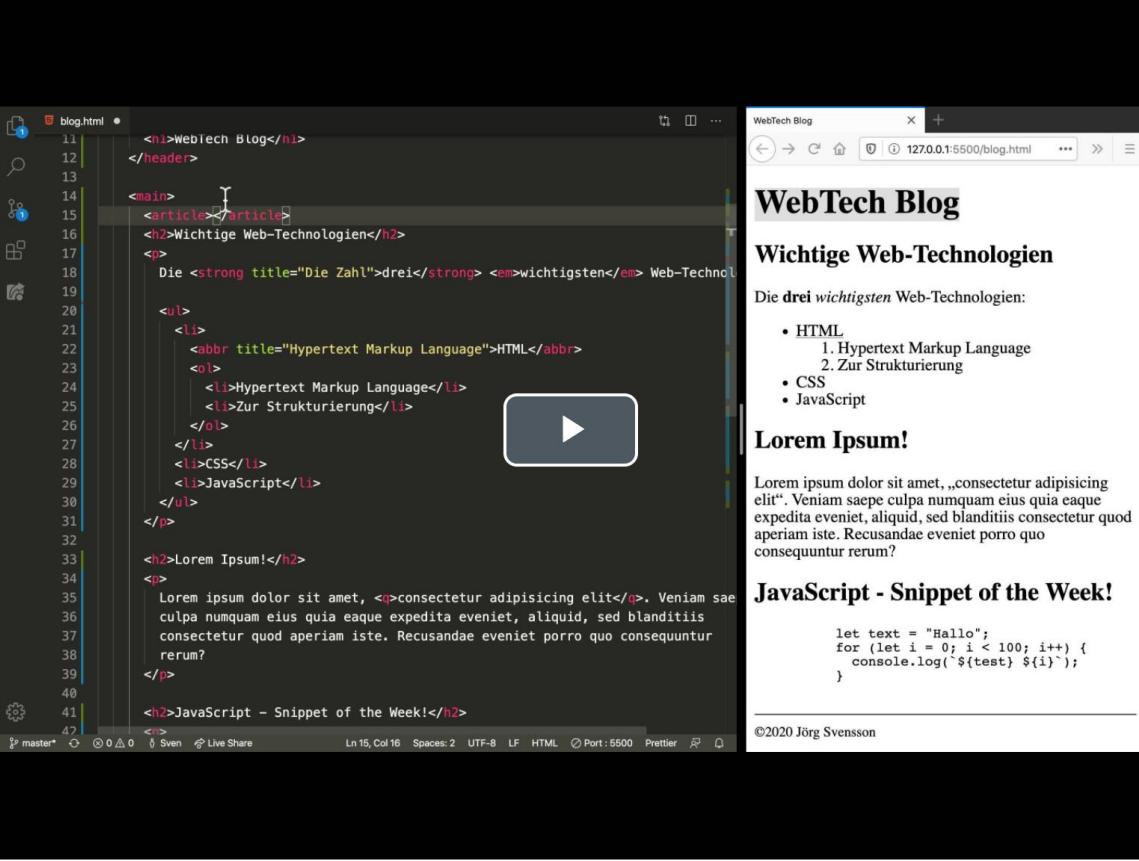
TABELLEN

- HTML-Elemente zur tabellarischen Strukturierung von Daten
- ! Keine optische Formatierung - diese erfolgt mit CSS*
- Attribute zur Formatierung wurden mit HTML5 daher entfernt

* Später mehr dazu!

VIDEO: TABELLEN

Video in ILIAS:



The image shows a split-screen view. On the left is a code editor displaying the file `blog.html`. The code contains HTML and some CSS-like styling. A play button icon is overlaid on the code editor's interface. On the right is a web browser window titled "WebTech Blog" showing the rendered content of the `blog.html` file. The browser window includes a header bar with icons and a status bar at the bottom.

`blog.html`

```
11  <h1>WebTech Blog</h1>
12  </header>
13
14  <main>
15  <article><h2>Wichtige Web-Technologien</h2>
16  <p>
17  Die <strong title="Die Zahl">drei</strong> <em>wichtigsten</em> Web-Technologien
18
19  <ul>
20  <li>
21  <abbr title="Hypertext Markup Language">HTML</abbr>
22  <ol>
23  <li>Hypertext Markup Language</li>
24  <li>Zur Strukturierung</li>
25  </ol>
26  </li>
27  <li>CSS</li>
28  <li>JavaScript</li>
29  </ul>
30  </p>
31
32  <h2>Lorem Ipsum!</h2>
33  <p>
34  Lorem ipsum dolor sit amet, „consectetur adipisicing elit“.
35  Veniam saepe culpa numquam eius quia eaque
36  expedita eveniet, aliquid, sed blanditiis consectetur quod
37  aperiam iste. Recusandae eveniet porro quo consequuntur
38  rerum?
39  </p>
40
41  <h2>JavaScript - Snippet of the Week!</h2>
42  </div>
```

WebTech Blog

Wichtige Web-Technologien

Die **drei** wichtigsten Web-Technologien:

- HTML
 - 1. Hypertext Markup Language
 - 2. Zur Strukturierung
- CSS
- JavaScript

LOREM IPSUM!

LOREM IPSUM DOLOR SIT AMET, „CONSECTETUR ADIPISCING ELIT“. VENIAM SAEPE CULPA NUMQUAM EIUS QUIA EAQUE EXPEDITA EVENIET, ALIQUID, SED BLANDITIIS CONSEQUENTUR QUOD APERIAM ISTE. RECUSANDAE EVENIET PORRO QUO CONSEQUUNTUR RERUM?

JavaScript - Snippet of the Week!

```
let text = "Hallo";
for (let i = 0; i < 100; i++) {
  console.log(`${text} ${i}`);
}
```

©2020 Jörg Svensson

Quellcode zum Video: [GitLab](#) ↗

TABELLEN: BASISELEMENTE

Element	Bedeutung
---------	-----------

table	Tabelle
-------	---------

tr	Tabellenzeile (table row)
----	------------------------------------

td	Tabellenzelle (table data)
----	-------------------------------------

th	Tabellenzelle für Kopfdaten (table header)
----	---

TABELLEN: BEISPIEL

```
<table>
  <tr>
    <th>Land</th>
    <th>Einwohner (Mio.)</th>
  </tr>
  <tr>
    <td>China</td>
    <td>1.390,85</td>
  </tr>
  <tr>
    <td>Indien</td>
    <td>1.316,9</td>
  </tr>
</table>
```

Land	Einwohner (Mio.)
China	1.390,85
Indien	1.316,9

TABELLEN: ZELLEN ZUSAMMENFASSEN

- Spalten bzw. Zeilen können über die Attribute `colspan` bzw. `rowspan` zusammengefasst werden
- Der Zahlenwert des Attributes gibt an über wie viele Spalten bzw. Zeilen sich die Zelle erstreckt

ZELLEN ZUSAMMENFASSEN: BEISPIEL

```
<table>
  <tr>
    <th>Stunde</th>
    <th>Montag</th>
    <th>Dienstag</th>
  </tr>
  <tr>
    <td>1</td>
    <td rowspan="2">WEB1</td>
    <td></td>
  </tr>
  <tr>
    <td>2</td>
    <td>PK1</td>
  </tr>
  <tr>
    <td>3</td>
    <td colspan="2">WEB2</td>
  </tr>
</table>
```

Stunde	Montag	Dienstag
1		
2	WEB1	PK1
3		WEB2

TABELLEN: WEITERE STRUKTURIERUNG

- Tabellen können mit weiteren semantischen Bereichen versehen werden
- z.B. erweiterte Formatierung mit CSS (fixer Tabellenkopf, scrollbarer Tabelleninhalt)

Element	Bedeutung
thead	Kopfbereich
tbody	Inhaltsbereich
tfoot	Fußbereich

WEITERE STRUKTURIERUNG: BEISPIEL

```
<table>
  <thead>
    <tr>
      <th>Land</th>
      <th>Einwohner (Mio.)</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>China</td>
      <td>1.390,85</td>
    </tr>
    <tr>
      <td>Indien</td>
      <td>1.316,9</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <th>Gesamt</th>
      <th>2.707,75</th>
    </tr>
  </tfoot>
</table>
```

Land	Einwohner (Mio.)
China	1.390,85
Indien	1.316,9
Gesamt	2.707,75

WEB-TECHNOLOGIEN

C: HTML

08: MULTIMEDIA

SICHTBARER BEREICH EINES HTML-DOKUMENTS

Elemente innerhalb des body-Elements:

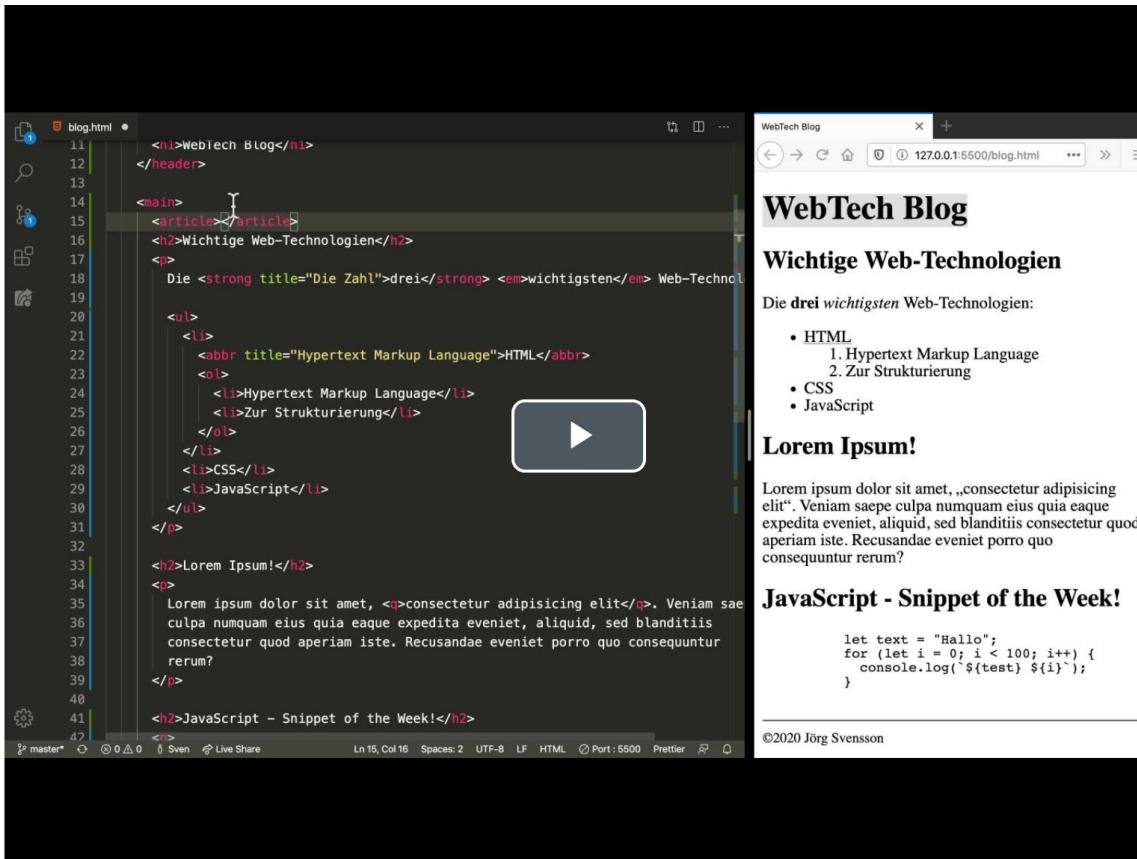
- Elemente zur Textauszeichnung
- Elemente zur Textstrukturierung
- Sektionselemente
- Hyperlinks
- Tabellen
- Multimedia
- Formulare

MULTIMEDIA

- Erinnerung an Themenblock B (Web-Grundlagen): „*Hypermedia ist Hypertext, welcher kein Text sein muss, sondern z.B. auch Videos, Audioinhalte oder Grafiken enthalten kann*“
- HTML unterstützt die Einbettung/Referenzierung verschiedenster Multimedia-Inhalte über entsprechende Elemente

VIDEO: Bilder

Video in ILIAS:



The image shows a split-screen view. On the left is a code editor displaying the file `blog.html`. The code contains HTML and some placeholder text (Lorem Ipsum). A play button icon is overlaid on the code editor area. On the right is a web browser window titled "WebTech Blog" showing the rendered content of the `blog.html` file. The browser displays a header, a section titled "Wichtige Web-Technologien", a list of technologies, a "Lorem Ipsum!" section, and a "JavaScript - Snippet of the Week!" section with a small code snippet.

```
<h1>WebTech Blog</h1>
</header>

<main>
  <article>
    <h2>Wichtige Web-Technologien</h2>
    <p>Die <strong title="Die Zahl">drei</strong> <em>wichtigsten</em> Web-Technologien sind: <br/>
      <ul>
        <li><abbr title="Hypertext Markup Language">HTML</abbr></li>
        <li>Hypertext Markup Language</li>
        <li>Zur Strukturierung</li>
      </ul>
    </p>
    <h2>Lorem Ipsum!</h2>
    <p>Lorem ipsum dolor sit amet, „consectetur adipisicing elit“. Veniam saepe culpa numquam eius quia eaque expedita eveniet, aliquid, sed blanditiis consectetur quod aperiam iste. Recusandae eveniet porro quo consequuntur rerum?</p>
    <h2>JavaScript - Snippet of the Week!</h2>
  </article>
</main>
```

WebTech Blog

Wichtige Web-Technologien

Die **drei wichtigsten** Web-Technologien:

- HTML
 - 1. Hypertext Markup Language
 - 2. Zur Strukturierung
- CSS
- JavaScript

LOREM IPSUM!

LOREM IPSUM DOLOR SIT AMET, „CONSECTETUR ADIPSICING ELIT“. VENIAM SAEPE CULPA NUMQUAM EIUS QUIA EAQUE EXPEDITA EVENIET, ALIQUID, SED BLANDITIIS CONSEQUENTUR QUOD APERIAM ISTE. RECUSANDAE EVENIET PORRO QUO CONSEQUUNTUR RERUM?

JavaScript - Snippet of the Week!

```
let text = "Hallo";
for (let i = 0; i < 100; i++) {
  console.log(`${text} ${i}`);
}
```

©2020 Jörg Svensson

Quellcode zum Video: [GitLab](#)

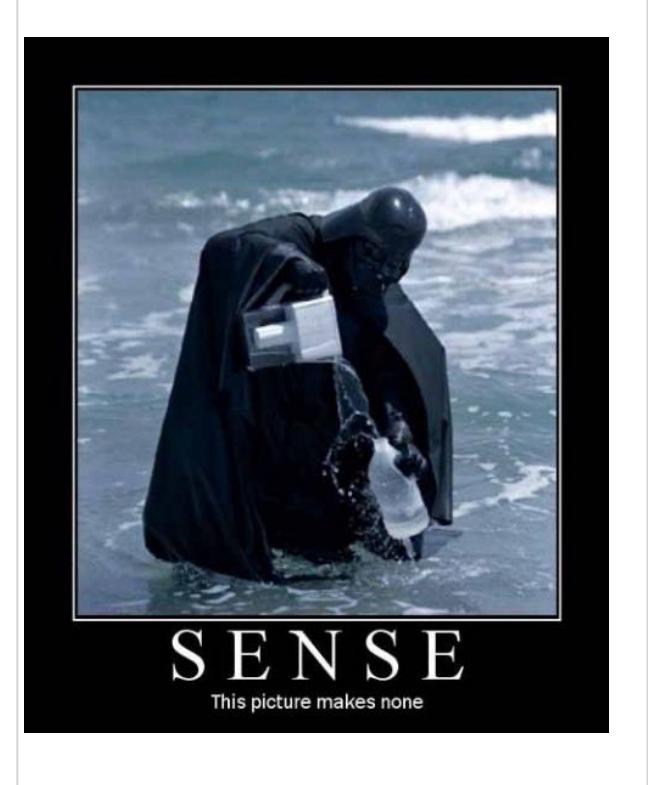
BILDER IN HTML

Eingebunden mit dem Element `img` (für `image`, Inline-Element):

```

```

- Attribut `src` definiert die URL zum Bild
- Attribut `alt` gibt einen Text an, der alternativ angezeigt wird, falls das Bild nicht angezeigt werden kann → besonders wichtig für Barrierefreiheit)
- unterstützte Formate z.B. WebP, PNG, GIF, JPEG



BILDER (2)

Höhe (`height`) und Breite (`width`) des Bildes angeben:

```
<!--Standardeinheit: Pixel-->  

```



! Achtung: Browser skaliert das Bild - die Bildgröße bleibt jedoch gleich!

BILDER (3)

! Empfehlung: Größe des Bildes immer angeben

→ Vorteil: Der Browser kann schon ausreichend Platz reservieren, bevor das Bild geladen ist (kein "Flackereffekt")

i Hinweis: Zur Unterstützung verschiedener Displaygrößen reichen absolute height- und width-Angaben nicht aus → *Responsive Bilder**

* Später mehr dazu!



Ohne Größenangabe



Mit Größenangabe

SEMANTISCHE AUSZEICHNUNG VON ABBILDUNGEN

Elemente `figure` und `figcaption` (Block-Elemente):

```
<figure>
  
  <figcaption>
    Abbildung: Vader ohne Sinn
  </figcaption>
</figure>
```



SENSE

This picture makes none

Abbildung: Vader ohne Sinn

SEMANTISCHE AUSZEICHNUNG VON ABBILDUNGEN (2)

- `figcaption` dient der Beschriftung (muss erstes oder letztes Element in `figure` sein)
- Nicht nur für Bilder anwendbar, auch z.B. für Tabellen, Videos, Code-Listings etc.

VIDEO

- Das `video`-Element ermöglicht das Einbinden von Video-Dateien
- Browser können Videos ohne Plugins/Erweiterungen abspielen
- Allerdings unterstützen die Browser teilweise unterschiedliche Video-Formate



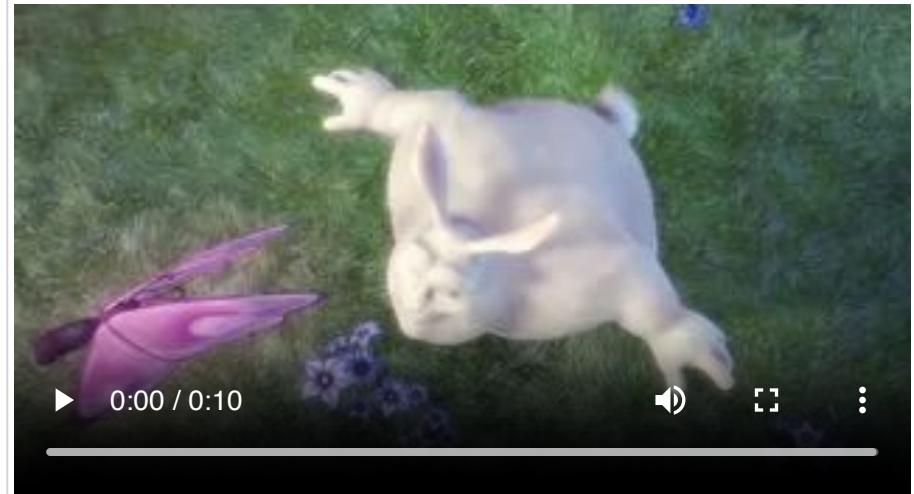
Unterstützung in Browzern → [Can I use video?](#) ↗

VIDEO: UNTERSTÜTZTE FORMATE (AUSZUG)

Format	Video-Codec	Audio-Codec	Medientyp
MP4	H264	AAC oder MP3	video/mp4
Ogg	Theora	Vorbis	video/ogg
WebM	VP8 oder VP9	Vorbis oder Opus	video/webm

VIDEO: BEISPIEL

```
<video src="assets/video-bbb.mp4"  
       width="500" controls>  
  Ihr Browser unterstützt das  
  video-Element nicht.  
</video>
```



- Attribut `src` definiert die URL zum Video
- Attribut `width` definiert die Breite des Videos (analog `img`)
- Attribut `controls` blendet Steuerelemente ein (browserspezifische Optik)
- Inhaltstext wird angezeigt, wenn das Video nicht dargestellt werden kann

VIDEO: BEISPIEL (2)

```
<video width="400" controls>
  <source src="assets/video-bbb.mp4" type="video/mp4">
  <source src="assets/video-bbb.ogv" type="video/ogg">
  Ihr Browser unterstützt das video-Element nicht.
</video>
```

- Um Probleme mit nicht unterstützten Formaten zu vermeiden:
Verschiedene Formate mittels `source` angeben
- Attribute: `src` zur Angabe der URL zur Datei, `type` für den Medientyp
- Browser wählt die erste Videodatei, deren Format unterstützt wird

AUDIO

- Das `audio`-Element ermöglicht das Einbinden von Audio-Dateien
- Verwendung vergleichbar zum `video`-Element
- Auch hier unterstützen die Browser unterschiedliche Formate



Unterstützung in Browzern → [Can I use audio?](#) ↗

AUDIO: UNTERSTÜTZTE FORMATE (AUSZUG)

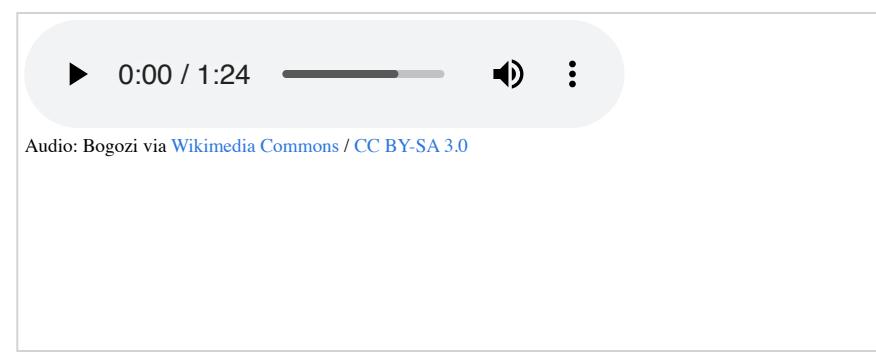
NP

Format	Audio-Codec	Medientyp
MP3		audio/mpeg
Vorbis		
Ogg	Opus	audio/ogg
FLAC		
AAC		audio/aac
WAV	PCM	audio/wav

AUDIO: BEISPIELE

Einbinden einer einzelnen Audiodatei:

```
<audio src="assets/tetris.ogg"
       controls>
  Ihr Browser unterstützt
  das audio-Element nicht.
</audio>
```



Anbieten verschiedener Formate: NP

```
<audio controls>
  <source src="assets/tetris.ogg" type="audio/ogg">
  <source src="assets/tetris.mp3" type="audio/mpeg">
  Ihr Browser unterstützt das audio-Element nicht.
</audio>
```

VIDEO UND AUDIO: ATTRIBUTES (WEITERE BEISPIELE)

Attribut Beschreibung

autoplay Video/Audio wird automatisch gestartet, sobald es geladen ist

loop Video/Audio in Endlosschleife spielen

muted Video/Audio ohne Ton abspielen

preload Einstellung, wie die Datei geladen wird. Mögliche Werte:

- *none*: Video/Audio wird nicht vorab geladen
- *metadata*: Nur Metadaten laden (z.B. Dauer)
- *auto*: Video/Audio darf vorab geladen werden (Browser entscheidet)

VEKTORGRAFIKEN MIT SVG

- SVG ([Scalable Vector Graphics ↗](#)) ist ein XML-basiertes Format zur Beschreibung von Vektorgrafiken
- + Vorteil: Vektorgrafiken sind auflösungsunabhängig (d.h. ohne Qualitätsverlust skalierbar)
- Einbindung in HTML erfolgt entweder als Referenz auf eine SVG-Datei oder durch direktes Einbetten

SVG: EINBINDEN PER REFERENZ

Beispiel:

```
<!-- Grafik (halbe Größe) -->  

```

```
<!-- Grafik (normale Größe) -->  

```

```
<!-- Grafik (doppelte Größe) -->  

```



SVG: EINBINDEN PER REFERENZ (2)

NP

- Einbindung erfolgt wie bei anderen Bilddateien über das `img`-Element
- Erstellung/Bearbeitung von SVG-Dateien über
 - beliebigen Texteditor (Klartext-Format!)
 - SVG-Editoren (z.B. [Inkscape](#) oder [SVG-edit](#))

SVG: DIREKTES EINBETTEN IN HTML

NP

Einbetten über das `svg`-Element:

```
<svg width="200" height="100">
  <ellipse fill="#0026ff"
    stroke="#000000"
    stroke-width="5"
    cx="98.742059" cy="51.710829"
    id="svg_1" rx="82.03125"
    ry="37.94643"/>
  <text fill="#ffffff"
    stroke-width="0"
    x="36.278429" y="67.989039"
    id="svg_2" font-size="55"
    font-family="Monospace"
    stroke="#000000">WEB1</text>
</svg>
```

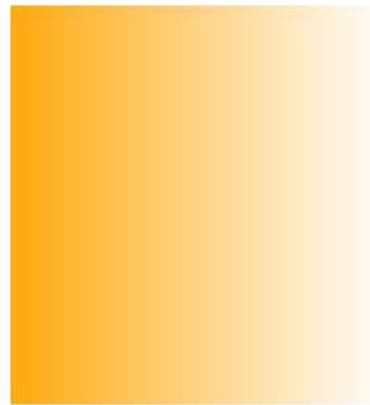


RASTERGRAFIKEN MIT CANVAS

- canvas-Element definiert rechteckigen Bereich ("Leinwand")
- Innerhalb dieses Bereiches kann mittels JavaScript gezeichnet werden
- Das Zeichnen erfolgt dabei pixelgenau (Raster)
- Verschiedene Modi erlauben neben zweidimensionalen Grafiken (Canvas 2D) auch z.B. hardwarebeschleunigte 3D-Grafiken ([WebGL ↗](#))

CANVAS 2D: BEISPIEL

```
<canvas id="meinCanvas" width="350" height="250">  
    Ihr Browser kennt das canvas-Element nicht!  
</canvas>  
  
<script type="text/javascript">  
    // Canvas-Element selektieren  
    const c = document.getElementById("meinCanvas");  
    // Kontext zum 2D-Zeichnen besorgen  
    const ctx = c.getContext("2d");  
  
    // Farbverlauf erstellen  
    const verlauf =  
        ctx.createLinearGradient(0,0,200,0);  
    verlauf.addColorStop(0,"orange");  
    verlauf.addColorStop(1,"white");  
  
    // Rechteck zeichnen und füllen  
    ctx.fillStyle=verlauf;  
    ctx.fillRect(10,10,300,200);  
</script>
```



Später mehr zu JavaScript!

SVG VS. CANVAS

SVG

Vektorgrafiken
(auflösungsunabhängig)

Teile der Grafik können (im DOM)
manipuliert werden, z.B. per CSS
oder JavaScript

Komplexe Animationen sind
langsam (DOM-Änderungen)

Canvas

Rastergrafiken
(auflösungsabhängig)

Bei Änderungen muss das
gesamte Bild neu gezeichnet
werden

Schnelle Animationen
möglich (z.B. bei Spielen)

WEB-TECHNOLOGIEN

C: *HTML*

09: FORMULARE

SICHTBARER BEREICH EINES HTML-DOKUMENTS

Elemente innerhalb des body-Elements:

- Elemente zur Textauszeichnung
- Elemente zur Textstrukturierung
- Sektionselemente
- Hyperlinks
- Tabellen
- Multimedia
- Formulare

FORMULARE

- Ermöglichen Interaktivität, insbesondere die Eingabe von Daten im Browser
- Beispiele: Account anlegen, Bestellung ausfüllen, Kommentar posten, Suchfunktion verwenden, Datei hochladen etc.

FORMULARE (2)

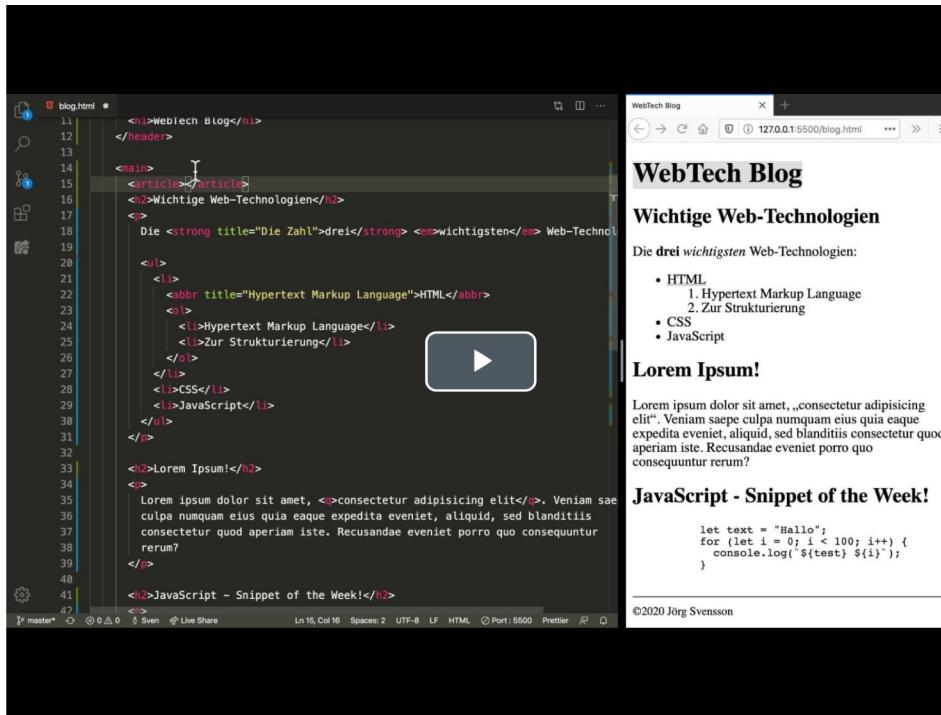
Verschiedene Verfahren zur Verarbeitung der eingegebenen Daten:

- Die Daten werden über den Anfrage-Antwort-Zyklus an den Server geschickt und dort verarbeitet
- Die Daten werden über AJAX asynchron an den Server geschickt und dort verarbeitet
- Die Daten werden clientseitig mittels JavaScript verarbeitet

! Im Folgenden: Betrachtung des klassischen Anfrage-Antwort-Zyklus

VIDEO: FORMULARE

Video in ILIAS:



Quellcode zum Video: [GitLab](#) ↗

DEFINITION EINES FORMULARES

Formularbereich mit `form`-Element festlegen:

```
<form action="kontakt.php" method="post">  
...  
</form>
```

- Attribut `action`: URL, welche beim Absenden des Formulares aufgerufen wird und an welche die eingegeben Daten übertragen werden (Standardwert: URL des aktuellen Dokuments)
- Attribut `method`: HTTP-Methode, mit der das Übertragen der Daten an den Server erfolgt

ERINNERUNG: HTTP-METHODEN

Methode Zweck

GET Lesen einer Ressource

POST Aktualisieren und Anhängen von Daten (auch:
Formulardaten übertragen)

PUT Neuanlage von Daten

DELETE Löschen von Daten

HEAD Anfordern der HTTP-Header (ohne Body)

ERINNERUNG: HTTP-METHODEN

Methode Zweck

GET Lesen einer Ressource

POST Aktualisieren und Anhängen von Daten (auch:
Formulardaten übertragen)

! Bei Formularen können die Methoden GET oder POST
verwendet werden (Standardeinstellung: `method="get"`).

FORMULAR MIT GET-METHODE

1. Formular:

```
<form action="http://foo.de/person.php"  
      method="get">  
    ...  
</form>
```

Vorname:

Nachname:

2. Daten werden an die URL gehängt (Anfrage-Teil):

```
http://foo.de/person.php?vorname=Tyler&nachname=Durden
```

3. HTTP-Request (Body ist leer):

```
GET /person.php?vorname=Tyler&nachname=Durden HTTP/1.1  
Host: foo.de
```

FORMULAR MIT POST-METHODE

1. Formular:

```
<form action="http://foo.de/person.php"  
      method="post">  
    ...  
</form>
```

Vorname:
Tyler

Nachname:
Durden

Absenden

2. Daten sind in URL nicht sichtbar:

```
http://foo.de/person.php
```

3. HTTP-Request (Daten im Body der HTTP-Nachricht):

```
POST /person.php HTTP/1.1  
Host: foo.de  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 29
```

```
vorname=Tyler&nachname=Durden
```

GET VS. POST

GET

Daten werden im Anfrage-Teil der URL übertragen

Daten sind einfach über die Adresszeile des Browsers manipulierbar

Je nach enthaltenen Zeichen müssen die Daten kodiert werden, z.B. "Tyler Dörden" → "Tyler%20D%C3%B6rden"

Datengröße durch maximale URL-Länge beschränkt (~ 2000 Zeichen)

Daten bleiben auch in Lesezeichen erhalten, da sie Teil der URL sind

Keine Nebeneffekte beim wiederholten Senden von Daten

Anwendung zur Abfrage von Daten (z.B. Suchanfragen)

POST

Daten werden im Body der HTTP-Anfrage übertragen

Daten sind schwieriger zu manipulieren

Daten können binär übertragen werden

Datengröße unbeschränkt

Daten werden in Lesezeichen nicht mitgespeichert

Wiederholtes Senden von Daten hat Nebeneffekte (z.B. doppelte Bestellung)

Anwendung zum Speichern von Daten (z.B. Posting, Bestellung)

EINGABEELEMENTE

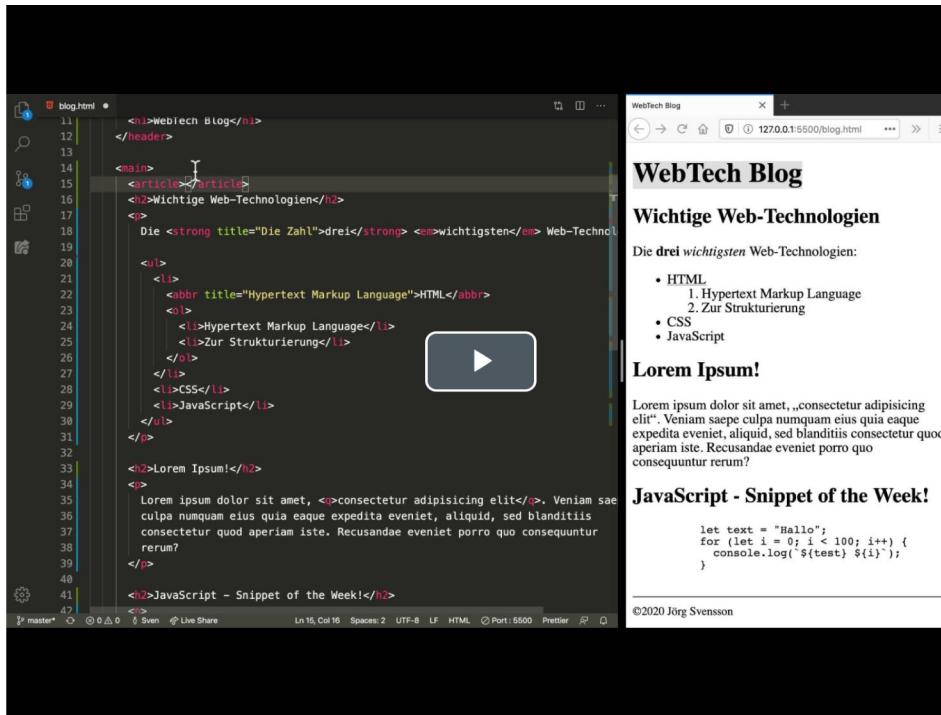
- HTML bietet (insbesondere seit HTML5) eine große Menge von Eingabefeldern
- ! Für alle Eingabeelemente gilt: Setzen des name-Attributes sorgt dafür, dass die Inhalte des Eingabefeldes von Formularen mitgesendet werden

input

- Das wichtigste Element ist `input` (Standalone-Tag)
- Über dessen Attribut `type` kann die Art des dargestellten Eingabefeldes festgelegt werden
- Im Folgenden betrachten wir einige Beispiele mit den jeweils wichtigsten Attributen

VIDEO: EINGABEELEMENTE

Video in ILIAS:



Quellcode zum Video: [GitLab](#) ↗

EINZEILIGES TEXTEINGABEFELD

`type="text"` (Standardwert für `input`):

```
<form>
  Suchbegriff:
  <input type="text" name="suchbegriff"
         size="20" maxlength="20"
         value="Eine Sucheingabe">
</form>
```

Suchbegriff: Eine Sucheingabe

- Attribut `name`: Name für den Formularwert (im HTTP-Request)
- Attribut `size`: Anzeigelänge in Zeichen
- Attribut `maxlength`: Maximal eingebbare Zeichen
- Attribut `value`: Wert zur Vorbelegung des Eingabefeldes

PASSWORTEINGABEFELD

`type="password":`

```
<form>
    Passwort:
    <input type="password" name="pw"
           size="15" maxlength="10"
           value="geheim123">
</form>
```

Passwort:

- Attribute analog zum einzeiligen Texteingabefeld
 - Eingegebene Zeichen werden maskiert (z.B. als Punkte oder Sterne)
- !** Lediglich Schutz vor "neugierigen Blicken", jedoch keine Verschlüsselung - Übertragung per HTTP erfolgt im Klartext!

SCHALTFLÄCHEN

`type="submit"`, `type="reset"`, Element `button`:

```
<form>
  <input type="text" id="meinText" name="meinText"
         value="Ein Text"><br>
  <input type="submit" value="Absenden">
  <input type="reset" value="Zurücksetzen">
  <button type="button" onclick="switchToUpperCase()">
    <strong>Groß</strong>buchstaben!
  </button>
</form>
```

A screenshot of a web browser window displaying a form. The form contains a text input field with the placeholder "Ein Text". Below it are three buttons: "Absenden", "Zurücksetzen", and a button labeled "Großbuchstaben!" which contains the text "Groß**buchstaben!**".

- `type="submit"`: Schaltfläche zum Absenden der Formulardaten an die unter `action` angegebene URL
- `type="reset"`: Schaltfläche zum Zurücksetzen aller Formularfelder auf die Initialwerte
- Element `button` als Alternative zu `input`, Vorteil: Kann weitere HTML-Elemente (wie z.B. Bilder) enthalten

RADIOBUTTONS

`type="radio":`

```
<form>
    Bitte wählen:
    <p>
        <input type="radio" name="gericht"
               value="wahl1">Wahlessen 1<br>
        <input type="radio" name="gericht"
               value="wahl2">Wahlessen 2<br>
        <input type="radio" name="gericht"
               value="aktion" checked>Aktionsteller<br>
    </p>
</form>
```

Bitte wählen:

- Wahlessen 1
- Wahlessen 2
- Aktionsteller

- Radiobuttons mit dem gleichen Wert im Attribut `name` gehören zu einer Gruppe
- Innerhalb dieser Gruppe ist lediglich ein Wert auswählbar
- Attribut `value`: Wert, der beim Absenden übertragen wird
- Attribut `checked`: Markiert den standardmäßig vorausgewählten Radiobutton

CHECKBOXEN

`type="checkbox":`

```
<form>
    Bitte Interessen auswählen:
    <p>
        <input type="checkbox" name="interessen"
               value="js">JavaScript<br>
        <input type="checkbox" name="interessen"
               value="html">HTML<br>
        <input type="checkbox" name="interessen"
               value="css">CSS<br>
    </p>
</form>
```

Bitte Interessen auswählen:

- JavaScript
- HTML
- CSS

- Attribute analog zu Radiobuttons
- Unterschied: Es sind mehrere Werte auswählbar (Übertragung an den Server z.B.: `interessen=js&interessen=css`)

EINGABEELEMENTE FÜR ZAHLEN

`type="number", type="range":`

```
<form>
    Zahl eingeben:
    <input type="number" min="10" max="40"
           value="10"><br>
    Zahl aus Bereich einstellen:
    <input type="range" min="1" max="100"
           step="5"><br>
    <input type="submit">
</form>
```

Zahl eingeben:

Zahl aus Bereich einstellen:

- `number`: Browser verhindert nicht-numerische Eingaben
- Attribute `min/max` definieren untere/obere Schranke für Eingaben
- Attribut `step` definiert Schrittweite bei Änderung des Wertes

EINGABEELEMENTE FÜR URLs UND E-MAIL-ADRESSEN

`type="url", type="email":`

```
<form>
URL eingeben:
<input type="url" value="https://w3c.org"><br>
E-Mail-Adresse einstellen:
<input type="email" size="30" multiple
      value="sven.joerges@fh-dortmund.de"><br>
<input type="submit">
</form>
```

URL eingeben:
E-Mail-Adresse einstellen:

- Browser überprüft die Eingaben, ob sie gültige URLs/E-Mail-Adressen sind
- Ergebnis der Prüfung kann über die CSS-Pseudoklassen `:valid` und `:invalid` dargestellt werden
- Attribut `multiple` ermöglicht die Eingabe mehrerer E-Mail-Adressen (kommasepariert)

EINGABEELEMENTE FÜR DATUM UND UHRZEIT

`type="date", type="time", type="datetime-local":`

```
<form>
  Datum:
    <input type="date" value="2021-12-10"
           min="2021-12-01" max="2021-12-31"><br>
  Uhrzeit:
    <input type="time" value="12:00"
           min="10:00" max="16:00"><br>
  Datum und Uhrzeit:
    <input type="datetime-local"
           value="2021-12-10T12:00"><br>
    <input type="submit">
</form>
```

Datum:

Uhrzeit:

Datum und Uhrzeit:

- Format für Datumsangaben: YYYY-MM-DD
- Format für Zeitangaben: HH:MM
- Format für kombinierte Angabe: YYYY-MM-DDTHH:MM

DATEIAUSWAHL UND -UPLOAD

`type="file":`

```
<form method="post" enctype="multipart/form-data">
    Bitte Datei auswählen:
    <input type="file" name="bild"
           accept="image/*">
</form>
```

Bitte Datei auswählen:
 Keine ausgewählt

- Browser erzeugt Schaltfläche, die einen Dateiauswahldialog öffnet
- Attribut `accept` erlaubt Einschränkung der Dateiauswahl auf bestimmte Dateitypen (im Beispiel: Bilddateien)
- Damit der Inhalt der Datei an den Server übertragen wird, muss im `form`-Element die Methode POST sowie die Kodierung `multipart/form-data` für die Daten angegeben werden

FARBAUSWAHL

`type="color":`

```
<form>
    Bitte Farbe auswählen:
    <input type="color" value="#000000">
</form>
```

Bitte Farbe auswählen: 

- Öffnet einen Farbauswahldialog
- Attribut `value` für Vorbelegung

WEITERE EINGABEELEMENTE

Neben `input` und `button` bietet HTML noch weitere Eingabeelemente:

- Mehrzeilige Textfelder: `textarea`
- Auswahl-/Dropdown-Listen: `select`
- Vorschlagslisten: `datalist`

MEHRZEILIGE TEXTFELDER

Element `textarea`:

```
<form>
    Kommentar eingeben:<br>
    <textarea cols="60" rows="8" maxlength="1000">
        ...
    </textarea>
</form>
```

Kommentar eingeben:

 Lorem ipsum dolor sit amet,
 consectetur adipisicing elit, sed
 do eiusmod tempor incididunt ut
 labore et dolore magna aliqua. Ut
 enim ad minim veniam, quis nostrud
 exercitation ullamco laboris nisi
 ut aliquip ex ea commodo
 consequat. Duis aute irure dolor //

- Attribut `cols`: Sichtbare Breite des Textfeldes
- Attribut `rows`: Anzahl der sichtbaren Zeilen (sind mehr Textzeilen im Feld enthalten wird ein Scrollbalken eingeblendet)
- Attribut `maxlength`: Maximale Anzahl eingebbarer Zeichen

AUSWAHL- UND DROPODOWN-LISTEN

Element select:

```
<form>
  <p>Einen Browser auswählen:</p>
  <select name="browser">
    <option value="firefox">Firefox</option>
    <option value="chrome" selected>Chrome</option>
    <option value="vivaldi">Vivaldi</option>
  </select>
  </p>
  <p>Mehrere Browser auswählen:</p>
  <select name="browsers" size="3" multiple>
    <option value="firefox">Firefox</option>
    <option value="chrome">Chrome</option>
    <option value="vivaldi" selected>Vivaldi</option>
  </select>
  </p>
  <p>Einen Browser auswählen (gruppiert):</p>
  <select name="browsers">
    <optgroup label="Gecko">
      <option value="firefox">Firefox</option>
    </optgroup>
    <optgroup label="Blink">
      <option value="chrome">Chrome</option>
      <option value="vivaldi" selected>Vivaldi</option>
    </optgroup>
  </select>
  </p>
</form>
```

Einen Browser auswählen: Chrome ▾

Firefox
Chrome
Vivaldi

Mehrere Browser auswählen:

Einen Browser auswählen (gruppiert): Vivaldi ▾

AUSWAHL- UND DROPDOWN-LISTEN (2)

select-Tag:

- Attribut `name`: Name für den Formularwert (im HTTP-Request)
- Attribut `size`: Anzahl der sichtbaren Einträge (Standardwert ist `size="1"` → Dropdown-Liste)
- Attribut `multiple`: Ermöglicht eine Mehrfachauswahl

option-Tag: Beschreibt einen Listeneintrag

- Attribut `value`: Wert, der beim Absenden übertragen wird
- Attribut `selected`: Vorabselektion

optgroup-Tag: Gruppieren von Listeneinträgen NP

Attribut `label`: Name der Gruppe

VORSCHLAGSLISTEN

Element `datalist`:

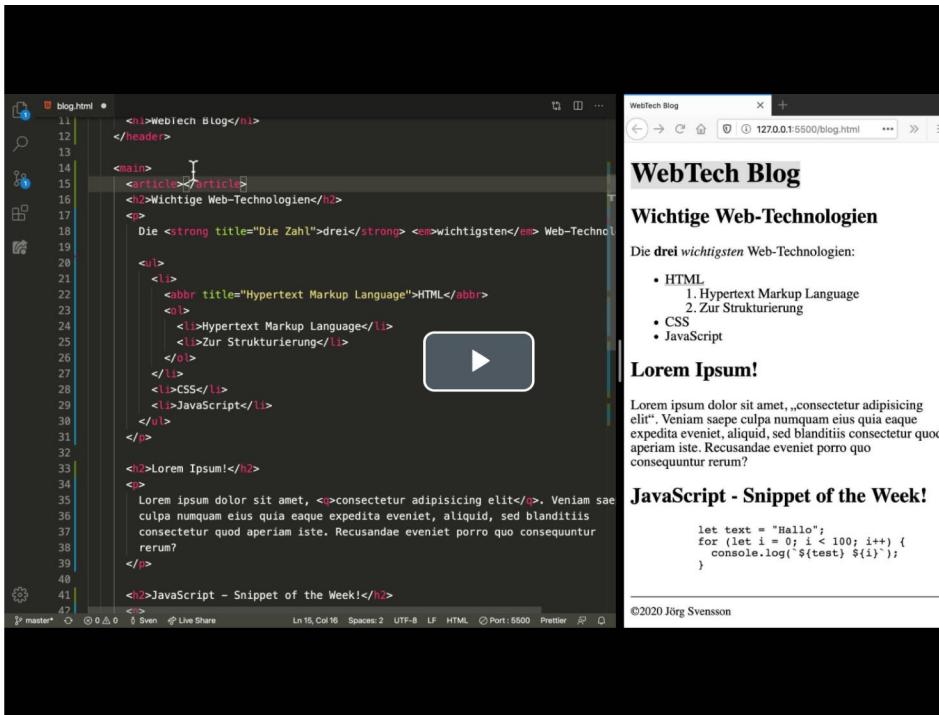
```
<form>
  Browser:
    <input list="browserliste" name="browser">
    <datalist id="browserliste">
      <option value="Chrome">
      <option value="Firefox">
      <option value="Vivaldi">
      <option value="Edge">
    </datalist>
</form>
```

Browser: Chrome

- `datalist`-Element definiert die Liste der Vorschläge
- Einzelne Vorschläge werden mit dem `option`-Element angegeben
- Die so definierte Vorschlagsliste wird dann über eine eindeutige `id` mittels `list`-Attribut z.B. an ein einzeiliges Textfeld gebunden
- Die Vorschlagliste erscheint dann z.B., wenn das Textfeld den Fokus erhält

VIDEO: FORMULARE STRUKTURIEREN

Video in ILIAS:



Quellcode zum Video: [GitLab](#) ↗

FORMULARE STRUKTURIEREN

Element label:

```
<form>
  <label for="vorname">Vorname</label>
  <input name="givenname" id="vorname" value="Mia"><br>
  <label for="nachname">Nachname</label>
  <input name="surname" id="nachname" value="Wallace">
</form>
```

Vorname
Nachname

- Beschriftungen von Eingabefeldern sollten mit `label` vorgenommen werden (Barrierefreiheit, besseres Klickverhalten z.B. bei Radiobuttons)
- Eingabeelement bekommt dazu einen eindeutigen Identifikator über das `id`-Attribut
- Das Attribut `for` des Labels verweist dann auf diese ID und bindet damit das Label an das Eingabeelement

FORMULARE STRUKTURIEREN

Element label (2):

```
<form>
  <label>Vorname
    <input name="givenname" value="Mia">
  </label><br>
  <label>Nachname
    <input name="surname" value="Wallace">
  </label>
</form>
```

Vorname	Mia
Nachname	Wallace

- Alternativ kann das Label auch an das Eingabeelement gebunden werden, indem das Eingabeelement in das Label verschachtelt wird
- Die Attribute `id` und `for` sind bei dieser Vorgehensweise nicht mehr notwendig

FORMULARE STRUKTURIEREN (2)

Elemente **fieldset** und **legend**:

```
<form>
  <fieldset>
    <legend>Daten</legend>
    <label for="vorname">Vorname</label>
    <input name="givenname" id="vorname"
           value="Mia"><br>
    <label for="nachname">Nachname</label>
    <input name="surname" id="nachname"
           value="Wallace">
  </fieldset>
  <fieldset>
    <input type="submit">
  </fieldset>
</form>
```

The screenshot shows a web form with a single fieldset. Inside the fieldset, there is a legend with the text "Daten". Below the legend are two input fields: one for "Vorname" containing "Mia" and another for "Nachname" containing "Wallace". A submit button labeled "Senden" is located at the bottom of the fieldset.

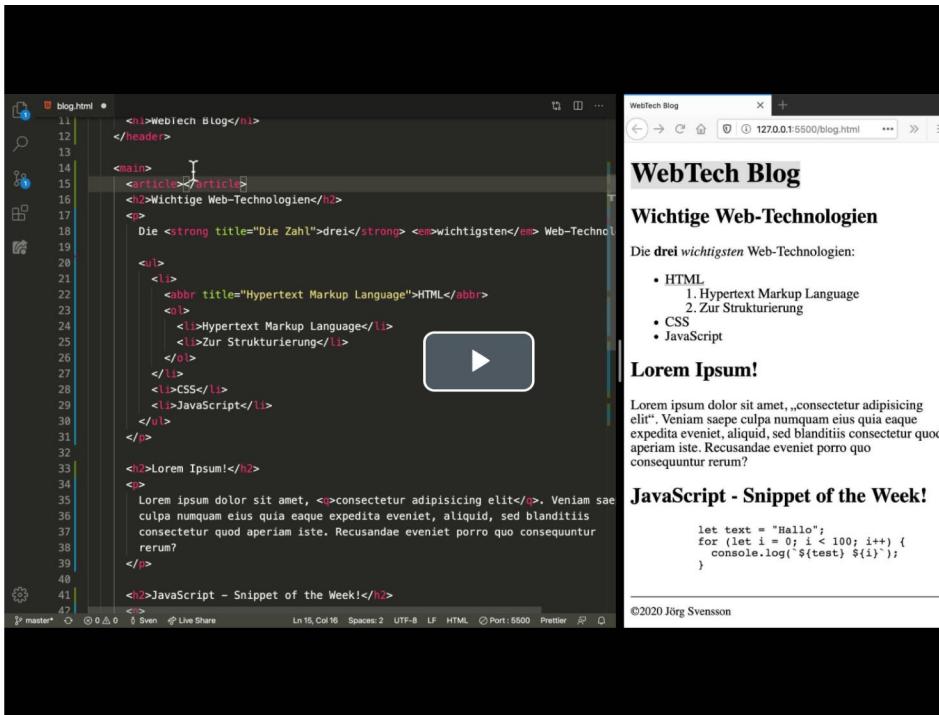
- **fieldset** erlaubt die Zusammenfassung mehrerer Formularelemente zu einer logischen Gruppe
- **legend** ist ein Unterelement von **fieldset** und weist der Gruppe eine Beschriftung zu

UNTERSTÜTZENDE ATTRIBUTE

- Für viele Eingabeelemente gibt es zusätzliche Attribute, die unterstützende Funktionen zur Verfügung stellen
- Damit lassen sich z.B. Validierungen durchführen, für die man früher JavaScript bemühen musste
- Beispiele: Autofokus, Erforderliche Eingaben, Platzhalter, Wertebereich

VIDEO: UNTERSTÜTZENDE ATTRIBUTE

Video in ILIAS:



Quellcode zum Video: [GitLab](#) ↗

AUTOFOKUS

Attribut **autofocus**:

```
<form>
  Vorname:
  <input name="vorname" value="Raoul" autofocus><br>
  Nachname:
  <input name="nachname" value="Duke">
</form>
```

Vorname: Raoul
Nachname: Duke

- Fokus ist nach Laden des HTML-Dokuments direkt auf dem Element mit dem **autofocus**-Attribut
- Sollte auf einer Seite nur einmal vorkommen

ERFORDERLICHE EINGABEN

Attribut `required`:

```
<form>
  Name: <input name="name" required><br>
  <input type="submit">
</form>
```

Name:

- Formular wird nicht abgesendet, wenn es noch unausgefüllte Eingabeelement mit `required`-Attribut gibt
- Browser visualisieren diese Felder speziell

PLATZHALTER

Attribut `placeholder`:

```
<form>
  Name:
  <input name="name"
         placeholder="Max Mustermann"><br>
  <input type="submit">
</form>
```

Name: Max Mustermann
Senden

- Gibt einen Inhalt als Platzhalter an, der angezeigt wird, solange das Eingabeelement noch keinen Wert hat
- Der Platzhalter verschwindet, sobald ein Wert eingetragen wird
- Browser visualisieren den Platzhalter typischerweise speziell (z.B. grau)

WERTEBEREICH FESTLEGEN

Attribute min, max und pattern:

```
<form>
    Benutzer-ID:
    <input name="id"
           pattern="[a-z]+[0-9]{3}"><br>
    Alter:
    <input type="number" name="age"
           min="18" max="100"><br>
    <input type="submit">
</form>
```

Benutzer-ID: ungültig

Alter: 15

Senden

- Attribut **pattern**: Angabe eines regulären Ausdrucks, der die erlaubten Eingaben festlegt
- Attribute **min/max** definieren untere/obere Schranke für numerische Eingaben
- Formular wird nicht abgesendet, wenn es noch Eingabeelemente mit ungültigen Werten enthält

WEB-TECHNOLOGIEN

C: HTML

10: EXKURS - REGULÄRE AUSDRÜCKE

EXKURS: REGULÄRE AUSDRÜCKE

- Ein regulärer Ausdruck ist ein Muster, welches eine Menge von Zeichenketten beschreibt
- Beispiel: „FH-Kennung beginnt mit 5 alphanumerischen Zeichen und endet mit 3 Ziffern“
- Formal:
 - Reguläre Ausdrücke beschreiben die **regulären Sprachen**
 - Zu jedem regulären Ausdruck existiert ein **endlicher Automat**, der die entsprechende reguläre Sprache akzeptiert



Tipp zum Ausprobieren: [Regex Pal ↗](#), [regex101 ↗](#)

EXKURS: REGULÄRE AUSDRÜCKE (2)

Bereiche:

Beispiel

Bedeutung

[acf]

Eines der Zeichen a, c oder f

[^acf]

Ein Zeichen außer a, c oder f

[1-3a-zA-C]

Eine der Ziffern 1, 2 oder 3, oder eines der Zeichen a, b, c, A, B, C

[1-3] [a-c] [A-
C]

1, 2 oder 3, gefolgt von a, b oder c, gefolgt von A, B, C
z.B.: 1bB, 3aC, 2bA

.

Ein beliebiges Zeichen (außer Zeilenumbruch)

EXKURS: REGULÄRE AUSDRÜCKE (3)

Quantoren:

Beispiel	Bedeutung
[acf] *	Keines oder beliebig viele der Zeichen a, c, f z.B.: a, acccc, ffa
[1-3] +	Eine oder mehr der Ziffern 1, 2, 3
[acf] ?	Keines oder eines der Zeichen a, c, f
[1-3] { 3 }	Genau drei der Ziffern 1, 2, 3 z.B.: 111, 123, 311
[1-3] { 4 , }	Vier oder mehr der Ziffern 1, 2, 3
[acf] { 2 , 5 }	Zwei bis fünf der Zeichen a, c, f

EXKURS: REGULÄRE AUSDRÜCKE (3)

Gruppen und Logik:

Beispiel

Web(Engineering)?

Web
(Engineering | Technologien)

Bedeutung

Zeichen zwischen den Klammern werden zu einem Element zusammengefasst (*capturing group*),
Beispiel passt auf Web und Web Engineering

| definiert Alternativen,
Beispiel passt auf Web
Technologien und Web
Engineering

EXKURS: REGULÄRE AUSDRÜCKE (4)

Zeichenklassen:

Beispiel Bedeutung

\d Eine Ziffer (entspricht [0-9])

\D Alles außer einer Ziffer (entspricht [^0-9])

\w Ein alphanumerisches Zeichen oder Unterstrich (entspricht [0-9a-zA-Z_])

\W Alles außer einem alphanumerischen Zeichen oder Unterstrich
(entspricht [^0-9a-zA-Z_])

\s Ein Unicode-Leerzeichen (z.B. Leerzeichen, Umbruch, Tabulator)

\S Alles außer einem Unicode-Leerzeichen

WEB-TECHNOLOGIEN

C: HTML

11: KOPFDATEN

GRUNDSTRUKTUR EINES HTML-DOKUMENTS

```
<!DOCTYPE html>
<html>
    <!-- Kopfdaten des HTML-Dokuments, werden nicht dargestellt -->
    <head>
        <title>Titel meiner Web-Seite</title>
        <meta charset="utf-8">
    </head>

    <body>
        Mein sichtbarer Inhalt
    </body>

</html>
```

KOPFDATEN EINES HTML-DOKUMENTS

Elemente innerhalb des head-Elements:

- title
- base
- link
- style
- script
- meta

TITEL DES HTML-DOKUMENTS

Element title:

```
<title>Ein aussagekräftiger Titel</title>
```



Muss für valides HTML zwingend vorhanden sein!

BASIS-URL DES HTML-DOKUMENTS

Element base:

```
<base href="https://beispiel.de/">
```

- Legt eine Basis-URL für das gesamte HTML-Dokument fest
- Diese Basis-URL wird für alle relativen URLs im HTML-Dokument verwendet (z.B. Hyperlinks, `img`-Elemente)
- Beispiel:

```
<!-- Wird aufgelöst nach: https://beispiel.de/bilder/logo.png -->

```

BEZIEHUNG ZU EXTERNEN DOKUMENTEN

Element link:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

- Attribut `rel` gibt die Art der Beziehung an
- Wichtigste Anwendung: `rel="stylesheet"` (externes CSS-Stylesheet einbinden*)
- Viele weitere Werte für `rel` möglich → [Link-Types \(WHATWG-Standard\)](#) ↗ NP

* Später mehr dazu!

DOKUMENTGLOBALE CSS-STILE

Element style:

```
<style>
  /* Färbt alle Überschriften erster Ordnung grün */
  h1 {
    color: green;
  }
</style>
```

CSS-Styles direkt im HTML-Dokument einbetten 

* Später mehr dazu!

SKRIPTE

Element `script`:

```
<!-- Direktes Einbetten von Skript-Code -->
<script>
  // Nerviges Popup beim Laden der Seite anzeigen
  window.onload = alert("Nerviges Popup!");
</script>

<!-- Direktes Einbetten von Skript-Code -->
<script src="javascript.js"></script>
```

- Element kann Skript-Code direkt einbetten oder auf eine externe Datei verweisen, die den Code enthält
- Typischerweise JavaScript als Skriptsprache*

* Später mehr dazu!

METADATEN ZUM HTML-DOKUMENT

Element meta, einige Beispiele:

```
<!-- Zeichenkodierung des HTML-Dokumentes -->
<meta charset="utf-8">

<!-- Autor, Schlüsselwörter, Beschreibung
     (z.B. interessant für Suchmaschinen) -->
<meta name="author" content="Sven Jörges">
<meta name="keywords" content="html, css, javascript">
<meta name="description" content="Dies ist eine sinnvolle Beschreibung.">

<!-- Hinweis für Webcrawler von Suchmaschinen,
     hier: nicht in Suchindex aufnehmen -->
<meta name="robots" content="noindex">

<!-- HTML-Dokument alle 30 Sekunden neu laden -->
<meta http-equiv="refresh" content="30">

<!-- Nach 5 Sekunden weiterleiten auf anderedomain.de -->
<meta http-equiv="refresh" content="5, URL=https://anderedomain.de/">
```