



FH Salzburg

VO Web-Technologien

20.09.2023, Oliver Jung

Technik
Gesundheit
Medien

WWW-Dokumente

Hypertext und Hypermedia



WWW ist ein **Hypermedia-System** auch **Hypertext-System**, **WWW-Dokumente** sind durch **Hyperlinks** untereinander verbunden und auf über die Welt verteilten Servern gespeichert. Das WWW wird so zum **verteilten** Hypermedia-System

- Hypermedia-Dokumente überwinden die traditionelle, lineare Dokumentenstruktur – mit dem Link-Mechanismus werden sie Teil eines riesigen Informationsnetzwerks
- Nutzer können ein WWW-Dokument direkt von einem anderen WWW-Dokument aus erreichen, selbst wenn diese auf unterschiedlichen Servern gespeichert sind
- Für die auf verschiedenen Servern gespeicherten WWW-Dokumente ist es schwierig, die Konsistenz von Hyperlinks zu gewährleisten

Hypertext und Hypermedia



WWW-Dokumente werden typischerweise mit einer speziellen deskriptiven Sprache beschreiben: der **Hypertext Markup Language – HTML**

Mit HTML können Autoren:

- **Struktur** eines Dokuments beschreiben, also z.B. Unterteilung in (Unter-)Überschriften und Absätze, Listen, Tabellen, ...
- Hyperlinks in die Dokumente einbetten,
- Multimedia-Komponenten einbinden und
- (bis zu einem gewissen Grad) die graphische Aufbereitung beeinflussen

Als WWW-Dokumente werden aber nicht nur HTML-Dokumente bezeichnet, sondern alle Dokumente, auf die im WWW zugegriffen werden kann

HTML-Dokumente



- HTML-Dokument heißt **Page** (dt. Seite)
- Die Startseite oder Eingangsseite für die Navigation durch ein Informationsangebot im WWW nennt man **Homepage**.
Zunehmend hat sich dieser Begriff aber auch als Bezeichnung für das gesamte Informationsangebot eines Anbieters (die **Website**) durchgesetzt
- Browser fordern ein HTML-Dokument an, interpretieren es und stellen es dann anhand der dort angegebenen HTML-Elemente dar.
- Jedes HTML-Element ist durch einen **Tag** gekennzeichnet. Tags sind als **Markups** in das Dokument eingebettet

HTML-Dokumente



Markup-Sprachen sollten nur für die Beschreibung der Dokumentenstruktur genutzt werden, nicht jedoch für die Beschreibung der Präsentation des Dokuments

- Grundlegende Idee von Markup-Sprachen ist die **Trennung** von **Dokumentenstruktur** und **Dokumentengestaltung**
- Mit **HTML** wird die Dokumentstruktur beschrieben
- **Cascading Stylesheets – CSS** – sind verantwortlich für die Beschreibung der Gestaltung (das Layout) dieser Struktur
 - Wichtigste Anwendungsgebiete für CSS:
 - exakte Definition des Layouts eines HTML-Dokuments
 - Anpassung an unterschiedliche Ausgabemedien
 - zentrales Management von Layouts

HTML-Dokumente



- Mit Hilfe spezieller Tags können Referenzen als **Links** in HTML-Dokumente eingebaut werden, die dort als passive „Wegweiser“ zu anderen Dokumenten dienen
- Die global eindeutige Identifikation der verlinkten Dokumente erfolgt mit Hilfe von **URIs** (Uniform Resource Identifiers)
- URIs können als Zeichenketten („Strings“) kodiert werden und geben in der Ausprägung als **URLs** den „Ort“ (Server) an, an dem ein Dokument gespeichert ist:
 - Name des Zugriffsprotokolls (z.B. http oder https),
 - Name des Servers (meist Domain Name),
 - Pfad (in der Verzeichnisstruktur des Servers)
 - und (Datei-)Name des Dokuments

Dynamisches HTML



- **DHTML – Dynamisches HTML:** Ein HTML-Dokument kann während der Anzeige durch Benutzerinteraktion (oder andere Ereignisse) verändert werden, z.B. Ausklappen eines Menüs, Runterzählen eines Countdowns, ...
- Typische Technik dahinter ist das → **DOM-Skripting** (→ DOM – Document Object Model). Das sind Programme (Skripte), die im Browser ausgeführt werden und auf bestimmte Ereignisse (z.B. Maus-Clicks) reagierend das HTML-Dokument manipulieren
- Während der Anzeige eines Dokuments können Browser und Server mittels → **XMLHttpRequest** kommunizieren, bekannt z.B. aus: AJAX = **A**synchronous **J**avaScript **a**nd **X**ML

Wird in späteren Einheiten zu **Clientseitiger Programmierung** behandelt

XML



- Hauptkritikpunkt an HTML ist eingeschränkte Flexibilität aufgrund der in ihrer Bedeutung unveränderlich festgelegten Markups
- Mit der Einführung von **XML – Extensible Markup Language** – als Meta-Markupsprache ist es möglich, alle nur erdenklichen Dokumenttypen mit einer eigenen, speziellen Syntax zu beschreiben
- XML ist Ausgangspunkt für Reihe Anwendungs-spezifischer Markupsprachen, die genau auf die jeweiligen Anwendungsdomänen oder spezialisierte Ausgabegeräte zugeschnitten sind. Beispiele:
 - **MathML** – Mathematical Markup Language
 - **SVG** – Scalable Vector Graphics
 - **ODF** – Open Document Format for Office Applications
 - ...

```
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>

<food>
  <name>Belgian Waffles</name>
  <price>$5.95</price>
  <description>
    Our famous Belgian Waffles with
    plenty of real maple syrup
  </description>
  <calories>650</calories>
</food>

<food>
  <name>Homestyle Breakfast</name>
  <price>$6.95</price>
  <description>
    Two eggs, bacon or sausage, toast,
    and our ever-popular hash browns
  </description>
  <calories>950</calories>
</food>

</breakfast_menu>
```


SGML



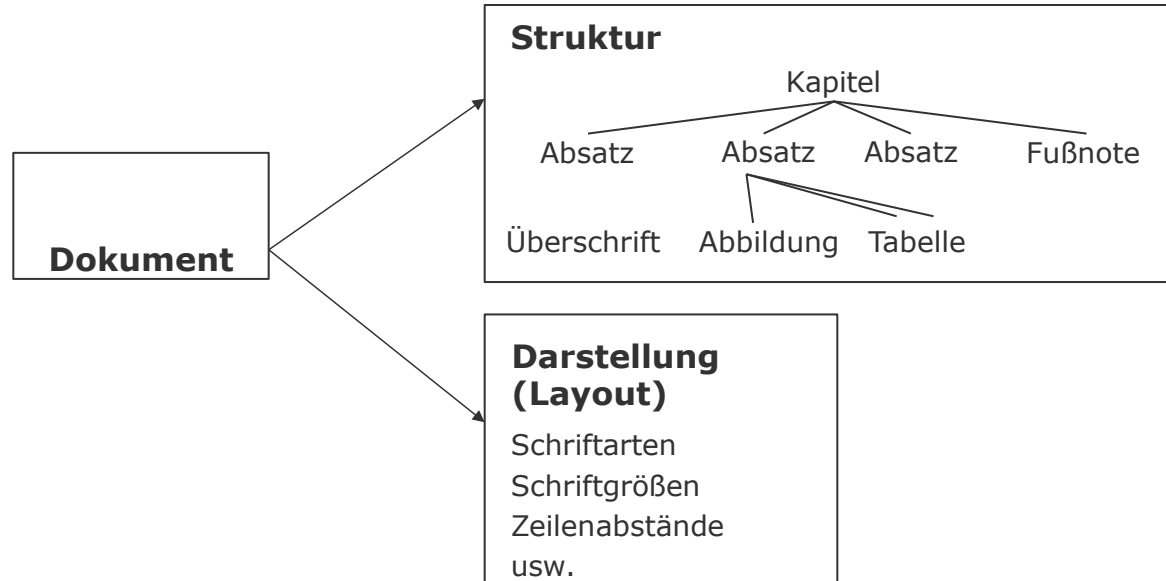
SGML – Standard Generalized Markup Language – wurde in den 1960er entwickelt, als moderne Datenverarbeitung ihren Anfang nahm

- SGML sollte den Austausch strukturierter Daten ermöglichen, unabhängig von Herstellern, Geräten oder Plattformen
- SGML ist eine Meta-Markupsprache, in der sich unterschiedliche Datenformate (genauer: Markup Sprachen für Datenformate) für alle möglichen (Text-)Dokumente beschreiben lassen
- **Grundlegende Idee:** Trennung von
 - **Dokumentstruktur** – logische Unterteilung und Abhängigkeiten der einzelnen Teile eines Dokuments – und
 - **Dokumentendarstellung** – Formatierung und Layout
- **Vorteil:** Aus einem SGML-Dokument könnten viele verschiedene Layout-Varianten generiert werden, z.B. für Druck im Magazin oder als Buch, für die Anzeige auf Computer oder Smartphone, ...

SGML



Trennung zwischen Struktur und Darstellung



SGML



- Strukturgebende Elemente werden mittels sogenannter **Markups** mit unterschiedlichen **Tags** gebildet
 - Markups sind durch Kombinationen spezieller Zeichen gekennzeichnet und werden innerhalb des eigentlichen Dokuments gespeichert
 - Markup-Begrenzer – „<...>“, z.B. <kapitel> ... </kapitel>
 - Interpreter können so die Tags leicht als solche erkennen
- Beziehungen zwischen den strukturellen Elementen eines SGML-Dokumentes werden durch den SGML **Dokumentenbaum** beschrieben
- SGML erlaubt als Metasprache die Definition eigener Dokumentklassen, z.B. für ein „Buch“, um Beschreibung von Instanzen dieser Klasse zu vereinheitlichen und **automatische Syntaxprüfung** zu ermöglichen
 - Dokumentenklassen werden mittels von **DTDs – Document Type Definitions** – festgelegt

Grundlegende Struktur eines HTML-Dokuments



- **HTML** ist eine Markupsprache, die auf SGML basiert
- HTML ist **einfacher** als SGML, da es keine Meta-Markupsprache ist – HTML ist eine Sprache, die „in SGML formuliert“ ist ...
- HTML führt das Hyperlink-Konzept des WWW ein
- **HTML-Dokumente** werden im Browser dargestellt; dieser muss dazu einen HTML-Interpreter besitzen, der die einzelnen strukturgebenden Elemente im Dokument identifizieren und für die sachgerechte Anzeige vorbereiten kann

Grundlegende Struktur eines HTML-Dokuments



- Historisch bedingt wird bei HTML die Trennung von Struktur und Darstellung nicht strikt verwirklicht, die SGML eigentlich fordert, sondern HTML bietet auch Reihe von Möglichkeiten zur Formatierung des Inhalts
- Dennoch können (und sollten) Autoren in HTML keine direkten Formatierungsanweisungen verwenden und stattdessen die Darstellung des Dokuments mittels separater Formatierungsanweisungen beschreiben
→ **Cascading Stylesheets (CSS)**
- Viele der ursprünglichen Formatierungsmöglichkeiten in HTML gelten heute als „ausgemustert“; insbesondere mit Einführung von HTML5 wurde eine ganze Reihe dieser alten Elemente gestrichen

Grundlegende Struktur eines HTML-Dokuments



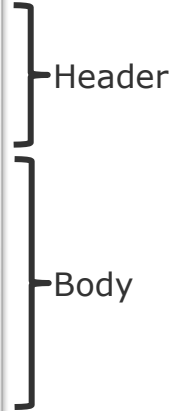
- Ein **HTML-Dokument** beginnt immer mit der Angabe der verwendeten HTML-Version, z.B.
 - `<DOCTYPE html>` für HTML5-Dokumente
- Ein HTML-Dokument besteht immer aus einem **Header** und einem **Body**
- Der **Header** enthält **Informationen über das Dokument**, die nicht zum eigentlichen Inhalt gehören, z.B.
 - Titel, Autor, Keywords, Kodierung, Sprache, ...
- Der **Body** enthält **eigentlichen Inhalt** des Dokuments, z.B.
 - Überschriften, Absätze, Listen, Tabellen, Formulare, Referenzen zu Bildern, Hyperlinks, usw.

Grundlegende Struktur eines HTML-Dokuments



Grundgerüst eines HTML-Dokuments

```
<!DOCTYPE html>
<html>
  <head>
    <title>About Me</title>
    ...
  </head>
  <body>
    <h1>Meine Introseite</h1>
    <p>lorem ipsum...</p>
    ...
  </body>
</html>
```



Grundlegende Struktur eines HTML-Dokuments



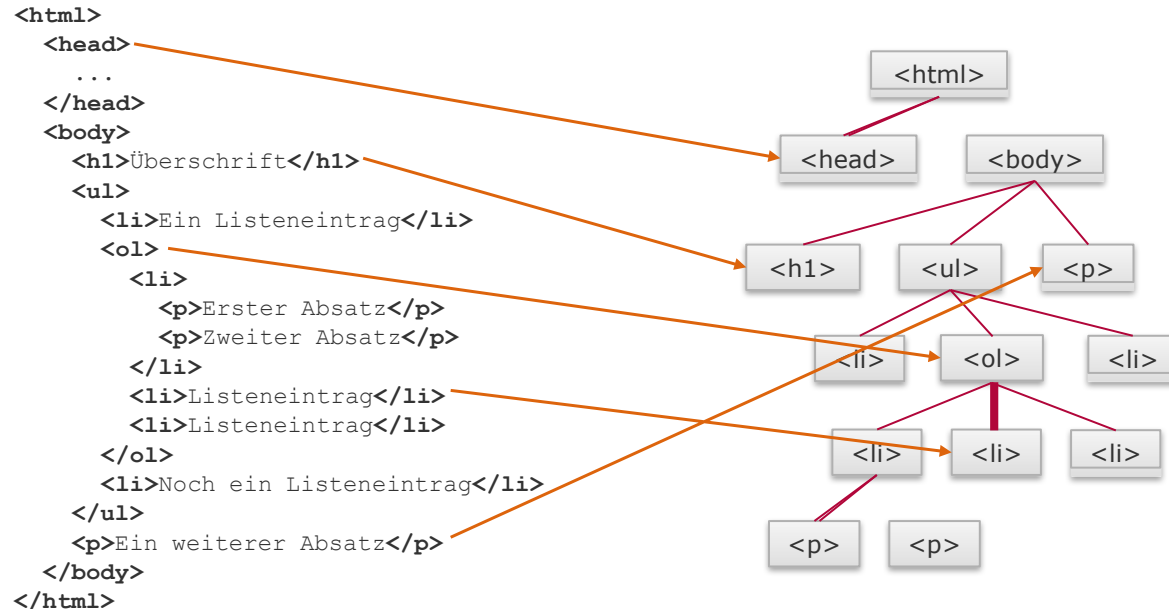
- HTML-Elemente werden – wie in SGML – durch **Markup** deklariert
- Markup wird durch Markup-Bezeichner – sogenannte **Tags** – gekennzeichnet, die jeweils von den Markup-Begrenzern `<` und `>` umschlossen werden
- Strukturelle Elemente des HTML-Dokuments werden jeweils von einen **Start-Tag** und einem **End-Tag** eingefasst, z.B.
 - `<p> Absatz </p>` (engl. „Paragraph“)
- **End-Tags** können in einigen Fällen weggelassen werden
- Tags können **Attribute** für zusätzliche Informationen enthalten, z.B. `<p id="absatz1">...</p>`
- Die aktuelle HTML-Version – HTML5 – kennt etwa 120 verschiedene Tags

Grundlegende Struktur eines HTML-Dokuments



- HTML-Elemente werden immer sequenziell angeordnet oder ineinander verschachtelt (niemals „über Kreuz“)

➔ dadurch ergibt sich Baumstruktur: **HTML-Dokumentenbaum**



Document Object Model – DOM



- **Document Object Model – DOM** – ist standardisierte Schnittstelle für **programmatischen Zugriff** auf Elemente eines HTML-Dokuments
- Zugriff auf DOM erfolgt mit Clientseitiger Programmierung (JavaScript)
- DOM erlaubt
 - Navigation durch Dokumentenbaum („Auswählen“ von einzelnen oder mehreren HTML-Elementen)
 - Hinzufügen, Löschen oder Verschieben von HTML-Elementen
 - Auslesen oder Ändern von Element-Inhalten (Text)
 - Auslesen oder Ändern von Attributen
- Beispiele:
 - `document.getElementsByTagName("p")` – selektiert alle <p>-Tags
 - `document.getElementById("Liste").children` – selektiert alle Unterelemente („Kindknoten“) des Elements mit der Id „Liste“

HTML Syntax und guter Stil



Neben den bereits vorgestellten Regeln zum HTML-Grundgerüst

- Angabe eines DOCTYPEs
- Kapselung des Dokuments in **<html>**-Tags
- Verwendung von Header und Body

gibt es weitere Regeln (**Syntax**), die HTML-Autoren beachten müssen oder Konventionen („**Coding Conventions**“), die sie beachten sollten:

- **Alle Tags schließen:** obwohl Browser recht „nachsichtig“ bei der Interpretation von HTML sind und viele Autorenfehler automatisch beheben, sollten geöffneten Tags immer auch geschlossen werden – „**wohlgeformtes HTML**“
 - gut: `<div><p>Das ist ein Absatz.</p></div>`
 - schlecht: `<div><p>Das ist ein Absatz.</div>`

HTML Syntax und guter Stil



- **Leere Elemente schließen:** Manche HTML-Elemente bestehen nicht aus Start- und End-Tag, da sie keine Inhalte haben können, z.B. der **
**-Tag für einen Zeilenumbruch
 - HTML5 fordert nicht, dass leere Elemente geschlossen werden, andere HTML-Versionen aber schon
 - gut: `
`
 - nicht ganz so gut: `
`
- **Keine kreuzweise Verschachtelung** von HTML-Tags: Enthält ein HTML-Element weitere Elemente, *müssen* diese korrekt verschachtelt sein
 - richtig: `<p><i>Kursiver Text</i></p>`
 - falsch: `<p><i>Kursiver Text</p></i>`

HTML Syntax und guter Stil



- Einheitlich **Kleinbuchstaben** verwenden für **Tag-Namen**

- gut: `<p>Das ist ein Absatz.</p>`
- schlecht: `<P>Das ist ein Absatz</P>`
- schlecht: `<P>Das auch.</p>`

- Einheitlich **Kleinbuchstaben** verwenden für **Attributnamen**

- gut: `<p class="info">`
- schlecht: `<p CLASS="info">`

- **Werte von Attributen in Anführungszeichen**

- gut: `<p title="Ein Absatz">`
- schlecht: `<p title=Absatz>`
- ganz schlecht: `<p title=Ein Absatz>`

Wert („Ein Absatz“) enthält ein Leerzeichen, „Absatz“ würde daher als neues Attribut interpretiert werden

HTML Syntax und guter Stil



■ Keine Leerzeichen in Attributzuweisungen

- gut: `<p class="info">`
- schlecht: `<p class = "info">`

■ Keine unnötigen Leerzeichen und Leerzeilen

- keine leeren Zeilen ohne triftigen Grund
- Leerzeilen nur zwischen größeren logischen Blöcken
- nicht jedes Element einrücken
- Einrückung nicht mit Tabulator-Taste, sondern mit 2 Leerzeichen

■ Dateinamen

- Dateinamen in Kleinbuchstaben mit Unterstrichen (`html_syntax.html`)
- Dateiendung für HTML: **.html** oder **.htm**
- Dateiendung für CSS: **.css**
- Dateiendung für JavaScript: **.js**

Dokumentenkopf und Meta-Tags



- Kopf (Header) eines HTML-Dokuments enthält *Informationen über das Dokument* – **Meta-Informationen**, z.B.
 - Titel, Name des Autors, Beschreibung, etc.
- Meta-Information sind wichtig für automatische Verarbeitung des Dokuments, z.B. für dessen Aufnahme in den Suchindex einer Suchmaschine
- Kopf eines HTML-Dokuments wird umschlossen von `<head>...</head>`
- Folgende Tags können im HTML-Header verwendet werden:
 - **title:** Titel der Seite zur Anzeige in der Browser-Kopfleiste
 - **base:** Basis-URL für alle URLs in der HTML-Datei; alle *relativen URLs* werden dann (vorne) ergänzt um diese Basis-URL
 - **style:** Definition von CSS-Styles für das HTML-Dokument
 - **link:** Einbettung externer Ressourcen, z.B. CSS-Dateien
 - ...

Dokumentenkopf und Meta-Tags



■ Folgende Tags können in einem HTML-Header verwendet werden:

- **script:** → JavaScript für das Dokument oder Einbettung einer externen JavaScript-Datei
- **noscript:** alternativer Seiteninhalt für Browser, die kein JavaScript ausführen können
- **meta:** Metainformationen zum Dokument, z.B.
 - `<meta name="author" content="Max Mustermann">`
 - `<meta name="keywords" content="HTML, CSS, XML, ...">`
 - `<meta charset="UTF-8">` (Zeichenkodierung des Inhalts)
 - `<meta http-equiv="refresh" content="5">`
(Lädt nach 5 Sekunden die Seite neu)

HTML-Elemente für Inhalte: Einführung



- Eigentlicher Inhalt eines HTML-Dokuments befindet sich im **Body**
- Body wird eingefasst von `<body>...</body>`
- HTML unterscheidet zwei Kategorien struktureller Elemente:
 - **Block-Elemente** – bestimmen Block-basierte Strukturen im Body des Dokuments
 - können weitere Block-Elemente oder Inline-Elemente enthalten
 - zwei aufeinanderfolgende Block-Elemente werden grundsätzlich **untereinander** angeordnet und nehmen die **volle Breite** ein
 - **Inline-Elemente** – befinden sich im Fließtext
 - können weitere Inline-Elemente enthalten
 - können am Zeilenende umgebrochen werden
 - Mittels CSS kann für viele Elemente festgelegt werden, ob diese als Block- oder Inline-Element behandelt werden sollen

HTML-Elemente für Inhalte: Text- und Struktur-Elemente



Die wichtigsten **grundlegenden Text- und Strukturelemente** sind:

- **<p>** – Paragraph, Absatz
- **<div>** – Division, Abschnitt
- **<h1>** bis **<h6>** – (Zwischen-)Überschriften absteigender Wichtigkeit
- **** – Abschnitte in Fließtext (Inline-Element)
- **<blockquote>** – Zitate
- **<i>**, **** – italics, emphasis, Hervorhebungen (Inline-Element, meist als schräggestellter Text dargestellt)
- **** – wichtiger Text (Inline-Element, meist fettgedruckter Text)

Die folgenden Tags kommen grundsätzlich **ohne schließenden End-Tag** aus:

- **<hr>** – horizontal rule, horizontale Linie
- **
** – break, Zeilenumbruch

HTML-Elemente für Inhalte: Hyperlinks



Herzstück des WWW sind **Hyperlinks**

- Hyperlinks können voneinander getrennte, unabhängige Informationsressourcen miteinander verbinden
- HTML kennt verschiedene Typen von Hyperlinks:
 - **Unsichtbare Links**, z.B. zwischen HTML-Dokumenten und CSS-Stylesheets, werden im Hintergrund automatisch vom Browser bearbeitet – **<link>**
 - **Sichtbare Links** werden durch einen Mausklick des Nutzers aktiviert und führen zu anderen Informationsressourcen im Web, die über ihre URIs identifiziert werden – **<a>**

HTML-Elemente für Inhalte:

Hyperlinks



<a>-Element kann für verschiedene Zwecke genutzt werden:

- Markierungen (Ankerpunkte) – ``
 ziel definiert einen eindeutigen Bezeichner für einen Ankerpunkt innerhalb eines Dokuments, zu dem ein Browser springen kann
- Sprung zu einem Ankerpunkt – `Verlinkter Text`
 Klick auf den Link Text springt zu einem Ankerpunkt mit dem Namen **ziel**
- Sprung zu einem externen Ziel – `Verlinkter Text`
 - URI kann absolut oder relativ angegeben werden, z.B.
 - Sprung zu anderem Server mit absolutem URI:
 `Google-Suche`
 - Sprung zu HTML-Dokument auf gleichem Server mit relativem URI:
 `Impressum`

HTML-Elemente für Inhalte:

Hyperlinks



<a>-Element kann für verschiedene Zwecke genutzt werden:

- Sprung zu externer Ressource und zu einem Ankerpunkt kann auch kombiniert werden, z.B.

```
<a href="https://example.com/pages/home#anker">  
    Homepage von Example  
</a>
```

Klick auf den Link öffnet Seite und springt zu angegebenem Anker

- **<a>-Tags** kennt neben **href** auch weitere **Attribute**, z.B.
 - **target**: bestimmt, wo verlinkte Ressource geöffnet werden soll
 - `Verlinkter Text`
Öffnet Link in neuem Fenster oder Tab (je nach Browser-Einstellung)
 - `Verlinkter Text`
Öffnet Link in gleichem Fenster oder Tab (Default-Target)

Bilder



Wichtigste „verlinkte Ressource“ in HTML-Dokumenten sind **Bilder**

- Haben eigenen Tag: **** (ohne Endtag)
- Wichtige Attribute:
 - **src:** URL, wo das Bild liegt (relativ oder absolut)
 - **alt:** alternativer Text, der angezeigt wird, wenn das Bild nicht geladen oder angezeigt werden kann, z.B. wegen schlechter Verbindung oder mit Screen Readern
 - ➔ sollte **immer** angegeben werden
 - **height / width:** Höhe und Breite, in der das Bild auf der HTML-Seite angezeigt werden soll
 - Wird nur ein Wert angegeben, wird der andere automatisch entsprechend der Original-Seitenverhältnisse vom Browser berechnet
 - Alternativ: **style="height: Höhe; width: Breite"**

Bilder



Beispiel

- Bilder können auch mit einem Link versehen werden

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

Ein Klick auf das Bild öffnet den Link in einem neuen Fenster

HTML-Elemente für Inhalte: Listen



HTML kennt verschiedene Typen von Listen

■ Ungeordnete Listen

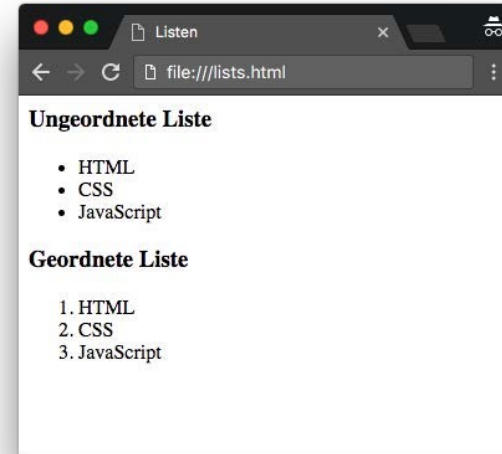
- Nicht-nummeriert mit Aufzählungszeichen
- Werden durch ****-Tags erzeugt
- Listenelemente stehen in ****-Tags

■ Geordnete Listen

- Engl. „ordered list“, nummeriert
- Werden durch ****-Tags erzeugt
- Listenelemente wieder in ****-Tags

■ ...

```
<h3>Ungeordnete Liste</h3>
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>
```



HTML-Elemente für Inhalte: Listen

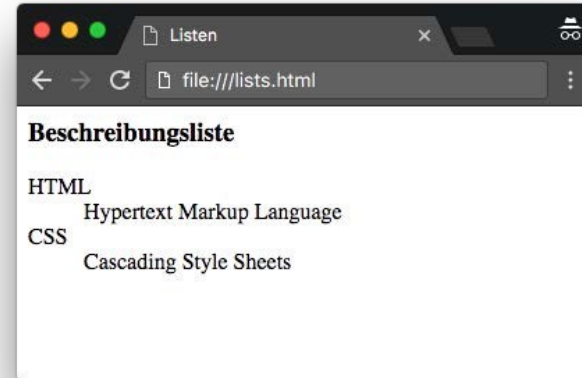


HTML kennt verschiedene Typen von Listen

■ Beschreibungslisten

- Engl. „description lists“
- Werden durch **<dl>**-Tag erzeugt
- Listenelemente bestehen aus
 - Begriffen (terms) – **<dt>**
 - Erläuterungen (descriptions) – **<dd>**
- Bei allen Listentypen kann die Darstellung mit CSS angepasst werden, z.B.
 - Form des Aufzählungszeichens
 - Art der Nummerierung

```
<h3>Beschreibungsliste</h3>
<dl>
  <dt>HTML</dt>
  <dd>Hypertext Markup Language</dd>
  <dt>CSS</dt>
  <dd>Cascading Style Sheets</dd>
</dl>
```



HTML-Elemente für Inhalte:

Tabellen



- Tabellen in HTML werden durch die Tags `<table>...</table>` erzeugt
- Tabellen bestehen aus rechteckigen Zellen, die in Zeilen und Spalten organisiert sind
 - **`<tr>`** erzeugt eine neue Zeile (table row) in einer Tabelle
 - **`<td>`** erzeugt eine neue Zelle in einer Tabellenzeile
 - **`<th>`** erlaubt die Auszeichnung spezieller Zellen für den Tabellenkopf (table header)
- Innerhalb einer Tabelle können Tabellenkopf und Tabellenfuß auch explizit festgelegt werden
 - **`<thead>`**, **`<tbody>`** und **`<tfoot>`** bezeichnen dann die einzelnen Bereiche innerhalb der Tabelle
 - Bereiche enthalten dann jeweils Zeilen und Spalten
 - Unterteilung erlaubt differenziertere Behandlung bei Skripting und Formatierung

HTML-Elemente für Inhalte:

Tabellen



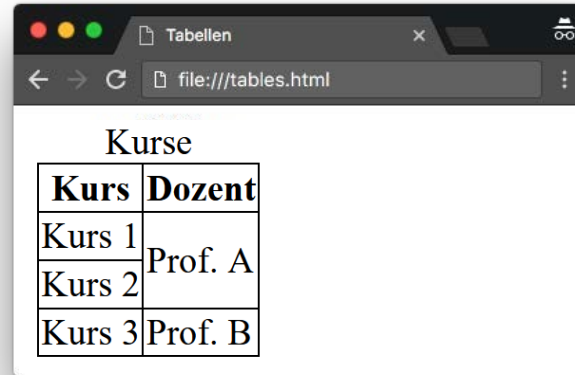
- **Benachbarte Zellen** können mit speziellen Attributen für `<td>` **zusammengefasst** werden
 - Nebeneinanderliegende Zellen: **colspan**
`<td colspan="2">Zelle geht über 2 Spalten</td>`
 - Übereinanderliegende Zellen: **rowspan**
`<td rowspan="2">Zelle geht über 2 Zeilen</td>`
- Tabellen können mit Beschriftung versehen werden
 - **<caption>**-Tag wird direkt als erstes innerhalb der Tabelle eingefügt
 - Ohne weitere Formatierung wird Caption oberhalb der Tabelle angezeigt
- Formatierung der Tabelle (Rahmen, Abstände, Ausrichtung, usw.) erfolgt mittels CSS

HTML-Elemente für Inhalte: Tabellen



Beispiel

```
<table>
  <caption>Kurse</caption>
  <tr>
    <th>Kurs</th>
    <th>Dozent</th>
  </tr>
  <tr>
    <td>Kurs 1</td>
    <td rowspan="2">Prof. A</td>
  </tr>
  <tr>
    <td>Kurs 2</td>
  </tr>
  <tr>
    <td>Kurs 3</td>
    <td>Prof. B</td>
  </tr>
</table>
```

A screenshot of a web browser window titled 'Tabellen'. The address bar shows 'file:///tables.html'. The page content displays a table with the caption 'Kurse'. The table has two columns: 'Kurs' and 'Dozent'. The first row contains 'Kurs 1' and 'Prof. A'. The second row contains 'Kurs 2' and 'Prof. A'. The third row contains 'Kurs 3' and 'Prof. B'.

Kurs	Dozent
Kurs 1	Prof. A
Kurs 2	
Kurs 3	Prof. B

Etwas CSS für bessere Lesbarkeit:

```
<style>
  table, td, th {
    border: 1px solid black;
    border-collapse: collapse;
  }
</style>
```

HTML-Elemente für Inhalte:

Formulare



- Zur Übertragung von Daten vom Browser zum Server, die über bloße Anforderung von Dokumenten hinaus gehen, gibt es HTML **Formulare**
- Formulare können für unterschiedliche Zwecke eingesetzt werden:
 - Anforderung ähnlich strukturierter Informationen, z.B.
 - Filter für Daten, Paginierung, ...
 - Spezifikation von Suchanfragen in großen Datenmengen
 - Übermittlung eigener Datensätze an einen Server
 - Individuelle Nutzerinteraktion, z.B. in Online Shops
 - Übermittlung von Dateien an einen Server
 - ...

HTML-Elemente für Inhalte:

Formulare



- Formulare werden mit dem **<form>**-Tag erzeugt:
`<form action="/login" method="POST">...</form>`
 - **action:** Gibt das Ziel des Formulars an – also eine URL, an die die Formulardaten mit HTTP gesendet werden
 - **method:** Gibt die HTTP-Methode an, mit der das Formular gesendet wird – also z.B. GET oder POST
- Es gibt eine **Reihe grundlegender Eingabefelder**, z.B.
 - **<input type="text">** – Texteingabefeld
 - **<input type="password">** – Texteingabefeld für Passwörter, Eingabe wird nicht angezeigt
 - **<input type="radio">** – „Radio Buttons“, runde Knöpfe für die Auswahl einer von mehreren Optionen
 - **<input type="checkbox">** – Boxen mit Häkchen für Mehrfachauswahl von Optionen

Texteingabe

.....

☒ Wahr
☐ Falsch

☒ HTML
☒ CSS

HTML-Elemente für Inhalte: Formulare



- Es gibt eine **Reihe grundlegender Eingabefelder**, z.B.

- Auswahlfeld zum Ausklappen („Dropdown-Liste“)

```
<select name="bestellung">  
  <option>Hamburger</option>  
  <option>Cheeseburger</option>  
  <option>Veggieburger</option>  
</select>
```

A light gray rounded rectangle representing a dropdown menu. It contains three items: a checked checkbox followed by 'Hamburger', 'Cheeseburger', and 'Veggieburger'.

- Textfeld für mehrzeiligen Text

```
<textarea rows="5" cols="20"></textarea>
```

A rectangular box with a thin gray border. Inside, the text 'Mehrzeilige Eingabe für Text' is written in a blue font. A small double-slash icon is in the bottom right corner.

- Button zum Absenden des Formulars

```
<input type="submit" value="Senden" />
```

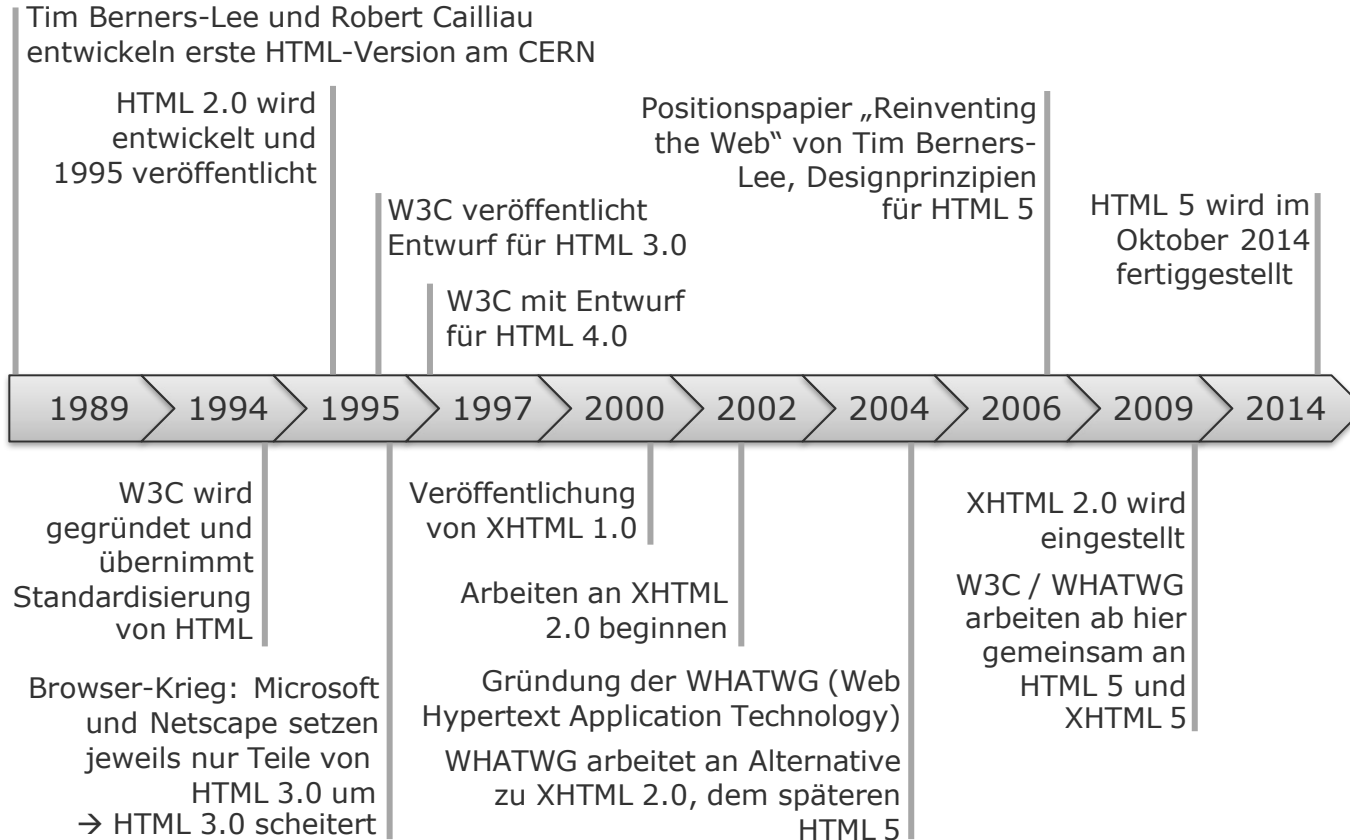
A rounded rectangular button with a light gray border and the text 'Senden' in a dark gray font.

HTML-Entitäten



- **Entitäten** werden bei Markup-Sprachen verwendet, um häufig vorkommende Zeichenketten zu verwalten und wiederzuverwenden
- HTML-Entitäten haben das Format **&Kennung;**
- Entitäten werden zum einen für Sonderzeichen verwendet, die für das Markup vorgesehen sind, z.B.
 - `<` – `<`; bzw. `>` – `>`;
 - `&` – `&`;
- Außerdem werden Entitäten für sprachspezifische Sonderzeichen genutzt, z.B.
 - Deutsche Umlaute: `ä` – `ä`; / `Ä` – `Ä`; / `ß` – `ß` / ...
 - Accents: `Á` – `Á`; / `ê` – `ê`; / `ò` – `ò`; / ...
 - Griechisches Alphabet: `π` – `π`; / `Ω` – `Ω`; / ...
 - Andere, z.B. `©` – `©`; / `½` – `½`; / `€` – `€`; / ...

Kurze Geschichte von HTML: von HTML 1.0 über XHTML zu HTML 5



HTML 5 – Ziele und Paradigmen



Nachdem HTML-Arbeitsgruppe des W3C zerbrach, konzentrierte sich die WHATWG auf die Entwicklung eines praxisorientierten Standards

- Neuer Standard sollte gemeinsam von Nutzern, Autoren, Designern, Browser-Herstellern, ... entwickelt werden
- **Entwicklungsparadigma:** „Nutzer vor Autoren vor Entwicklern vor Spezifikationsdesignern vor „reiner Lehre“
- HTML 5 solle
 - Stück für Stück entwickelt werden
 - abwärtskompatibel sein
 - einfach zu benutzen sein
 - noch stärker auf Trennung zwischen Struktur und Layout achten

HTML 5 – Was ist neu?



- HTML 5 basiert nicht länger auf SGML
 - Keine Document Type Definition (DTD)
 - Automatische Validierung (Syntaxprüfung) mit bestehenden Tools nicht mehr möglich, spezielle Werkzeuge werden benötigt
- HTML 5 führt Reihe „**sprechender**“ **Strukturelemente** ein, die eine bessere Strukturierung erlaubt
- HTML 5 bietet Reihe neuer Elemente für **Multimedia-Inhalte**
- Für **Formulare** gibt es viele neue Tags und Attribute
- **<canvas>**-Element ist Zeichenfläche für Animationen
- SVG (Scalable Vector Graphics) und MathML können direkt in HTML-Dokumenten verwendet werden
- Viele Elemente und Attribute für die Formatierung werden nicht mehr unterstützt

Semantische Elemente



- Meistverwendetes strukturierendes Block-Element in HTML 4.0 ist `<div>`-Tag
- HTML 5 führt dafür eine Reihe „sprechender Strukturelemente“, sogenannte **semantische Elemente** (Elemente mit „Bedeutung“) ein, z.B.
 - `<section>`
 - `<article>`
 - `<header>`, `<footer>`
 - `<aside>`
 - `<nav>`
 - ...
- Maschinen (z.B. Suchmaschinen) können so leichter identifizieren, welche Teile des Dokuments wichtig sind oder welche für die Navigation verwendet werden

Multimedia-Elemente



- Audio und Video konnten bisher nur mittels Browser-Plugins in HTML-Dokumenten verwendet werden, z.B. Windows Media Player, Quicktime, VideoLAN client, Adobe Flash, usw.
 - Neue Tags für **<audio>** und **<video>** erlauben es, Multimedia-Dateien direkt mit dem Browser abzuspielen
 - Wie alle Elemente sind auch die Multimedia-Tags Teil des
→ Document Object Models (DOM)
 - Bedienelemente (Play, Pause, Lautstärke, ...) können mit CSS gestaltet werden
- The Chrome media player interface, featuring a play button, a progress bar with a blue slider, a timestamp of 0:00 / 0:01, a volume icon, and a blue volume slider.
- Chrome
- The Safari media player interface, showing a play button, a 30-second skip forward button, a volume icon, and a full-screen button.
- Safari
- The Firefox media player interface, displaying a play button, a progress bar, a timestamp of 0:00, a timestamp of 0:02, a volume icon, and a volume level indicator.
- Firefox
- Komplexere Multimedia-Steuerung (z.B. Sprungmarken, Playlisten, usw.) können mit JavaScript realisiert werden

Multimedia-Elemente



- Betriebssysteme bzw. Browser müssen für Wiedergabe sogenannte **Codecs** (Verfahren für Kodierung/Dekodierung) unterstützen
- Vorgesehene Codecs für HTML 5 sind:
 - Audio: mp3, wma, AAC, Ogg Vorbis
 - Video: H.264, Ogg Theora, WebM
- Alternative Quellen (z.B. für unterschiedliche Codecs oder Bitraten) können mit **<source>**-Tag angegeben werden
- (Früher) **problematisch**: Standard legt nicht fest, welche Codecs unterstützt werden müssen, Browser-Hersteller handhaben das unterschiedlich, z.B. bei Video-Codecs:
 - Internet Explorer und Safari unterstützen nur H.264
 - Chrome, Firefox und Opera unterstützen alle Codecs
 - Früher mussten Inhaltenanbieter mehrere Codecs bereitstellen, heute reicht in der Regel H.264 aus

Multimedia-Elemente



Beispiel: <audio>-Tag mit verschiedenen Quellen

```
<audio>
  <source src="/audio/track1.wma" type="audio/x-ms-wma">
  <source src="/audio/track1.mp3" type="audio/mpeg">
  <p>Wenn Sie diesen Text sehen, unterstützt Ihr
    Browser den HTML5 Audio-Tag nicht</p>
</audio>
```

- ❑ Der Browser spielt das erste <source>-Element ab, dessen Codec er versteht
- ❑ Unbekannte Tags ignoriert ein Browser, daher ist es wichtig, einen Hinweis für ältere Browser hinzuzufügen

Formulare



- HTML 5-Formulare basieren auf Standard Web Forms 2.0
 - Abwärtskompatibel zu Web Forms 1.0 (gehört zu HTML 4.0)
- Gegenüber bisherigen HTML-Formularen gibt es einige Neuerungen, z.B.
 - Input-Elemente können auch außerhalb von Formularen stehen und diesen mit dem **form**-Attribut zugeordnet werden
 - `<input type="checkbox" form="registrierung">`
AGB akzeptiert
 - Neue Elemente: `<datalist>`, `<keygen>`, `<output>`
 - Neue Input-Typen, z.B. für Zahlen, Zeit und Datum, Farbwerte
 - Neue Attribute für Formularelemente, z.B.
 - für Eingabevalidierung: `min`, `max`, `required`, ...
 - mit Platzhalter

Formulare



Beispiele für neue Features

- Zahlenbereiche können mit Slider ausgewählt werden

```
<input type="range" min="0" max="10">
```



- Mit dem <datalist>-Element können Vorschläge (mit Auto-Vervollständigung) zu Texteingabe-Feldern ergänzt werden

```
<input type="text" list="browsers">
```

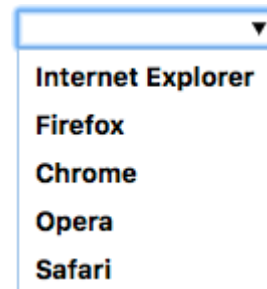
```
<datalist name="browsers">
```

```
  <option>Internet Explorer</option>
```

```
  <option>Firefox</option>
```

```
  ...
```

```
</datalist>
```



- Eingabefelder nur für numerische Werte

```
<input type="number" value="100">
```



Formulare



Beispiele für neue Features

- Eingabefelder mit Eingabevalidierung, z.B. für Email-Adressen

```
<input type="email">
```

```
<input type="url">
```

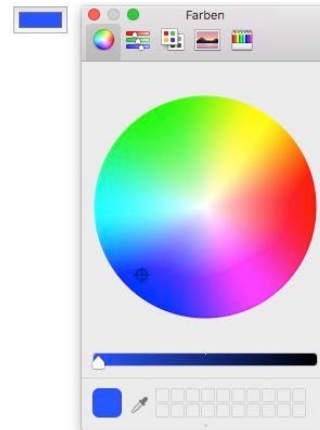
Bei fehlerhaften Eingaben wird Formular nicht gesendet

info

! Die E-Mail-Adresse muss ein @-Zeichen enthalten. In der Angabe "info" fehlt ein @-Zeichen.

- Farbauswahl mit „Color Picker“
(nicht für jeden Browser)

```
<input type="color">
```

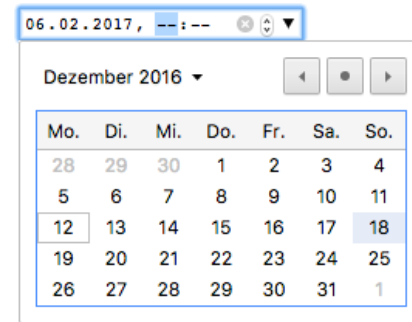


Formulare



Beispiele für neue Features

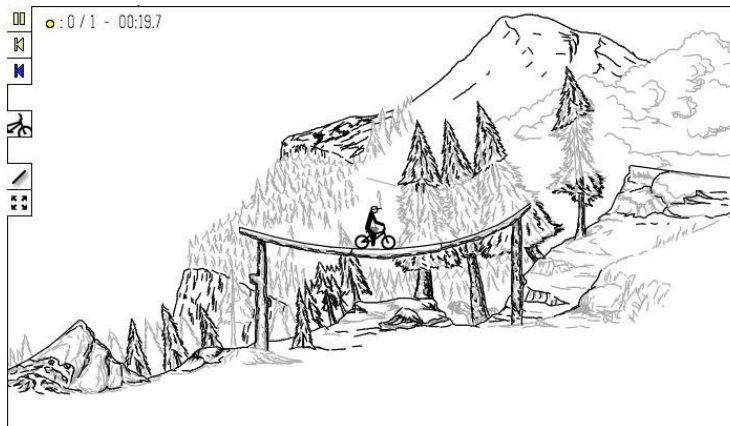
- Eingabefelder für Zeit und Datum, z.B.
`<input type="datetime-local">`
 - analog: date, time, month, week
 - Auswahl-Dialog für manche Browser
- Attribute für **Eingaberestriktionen**
 - max, min: höchste/niedrigste Werte
 - required: Feld muss ausgefüllt werden
 - pattern: regulärer Ausdruck für Eingabemuster
- Weitere neue Attribute
 - placeholder: Hinweis auf erwartete Eingabe
`<input type="email" placeholder="Ihre E-Mail-Adresse">`



Zeichenfläche – Canvas



- **<canvas>-Tag** bietet Zeichenfläche für Bitmap-Grafiken
- Ursprünglich von Apple für WebKit-Browser entwickelt, mittlerweile in den WHATWG-Standard integriert
- Zeichnen auf den Canvas mit JavaScript (JS), Animation der Zeichnungen ebenfalls mit JS möglich
- Unterstützte Zeichenelemente z.B.
 - Linien
 - Rechtecke
 - Kreise
 - Bögen
 - Bézier-Kurven
 - Farbverläufe
 - ...



<http://canvasrider.com>

Zeichenfläche – Canvas



Beispiel: Zeichnen eines blauen Quadrats

```
<canvas id="example" width="250" height="250">
```

Ihr Browser unterstützt HTML 5 Canvas nicht.

```
</canvas>
```

```
<script type="text/javascript">
```

```
    // Zugriff auf Canvas-Element über Variablen
```

```
    var example =
```

```
    document.getElementById('example'); var context  
    = example.getContext('2d');
```

```
    // Blau als Füllfarbe setzen
```

```
    context.fillStyle = "rgb(0,0,255)";
```

```
    // Quadrat zeichnen: Parameter sind obere linke
```

```
    // Ecke (10,10) und untere Rechte Ecke (100,100)
```

```
    context.fillRect(10,10,100,100);
```

```
</script>
```

Weitere Features und Browserunterstützung



- HTML 5 führt eine ganze Reihe weitere Features ein, z.B.
 - neue Tags wie z.B. <time>, <meter> oder <progress>
 - komplette Übersicht: <http://www.w3schools.com/tags/> oder <http://www.selfhtml5.org/>
- HTML 5 streicht Reihe von Tags aus HTML 4.01, vor allem im Bereich der Formatierung, z.B.
 - <center>, , <u>, <frame>, <frameset>, ...
- Ein großer Teil der HTML 5-Features werden heute von aktuellen Versionen der gängigen Browser unterstützt – aber noch läuft nicht alles überall
 - Übersicht: <http://caniuse.com/>

Wichtige HTML Elemente



Block Elemente

Tag Name	Kurz Beschreibung
<div>	Division
<h1> - <h6>	Heading – Überschriften
<nav>	Navigation
<section>, <article>	Allgemeine Sektion, Artikel
<main>	Hauptinhalt
<header>	Kopfzeile
<footer>	Fußzeile
<p>	Paragraph
/ & 	(Un-)Geordnete Listen mit Elementen
<blockquote>	Zitat
<table>	Tabelle
<tr>, <th>, <td>	Tabellen- zeile, kopfzelle, zelle
<svg>/<canvas>	Zeichnungen
<video>/<audio>	Video und Audio einbetten
<form>	Formulare

Inline Elemente

Tag Name	Kurz Beschreibung
	„to span over“ Inline Element
<i>	Text Hervorhebungen
<a>	Anchor, Link
	Bild
<input>	Eingabe (verschiedene Typen!)
<button>	Button

Nicht sichtbare Elemente

Tag Name	Kurz Beschreibung
<head>	HTML Dokument Kopf
<title>	Titel der Seite
<style>	Eingebettetes CSS
<meta>	Meta Informationen für das Dokument
<script>	Eingebettetes JavaScript



FH Salzburg

VO Web-Technologien

20.09.2023, Oliver Jung