Webentwicklung

# Frontend: Browser & HTML

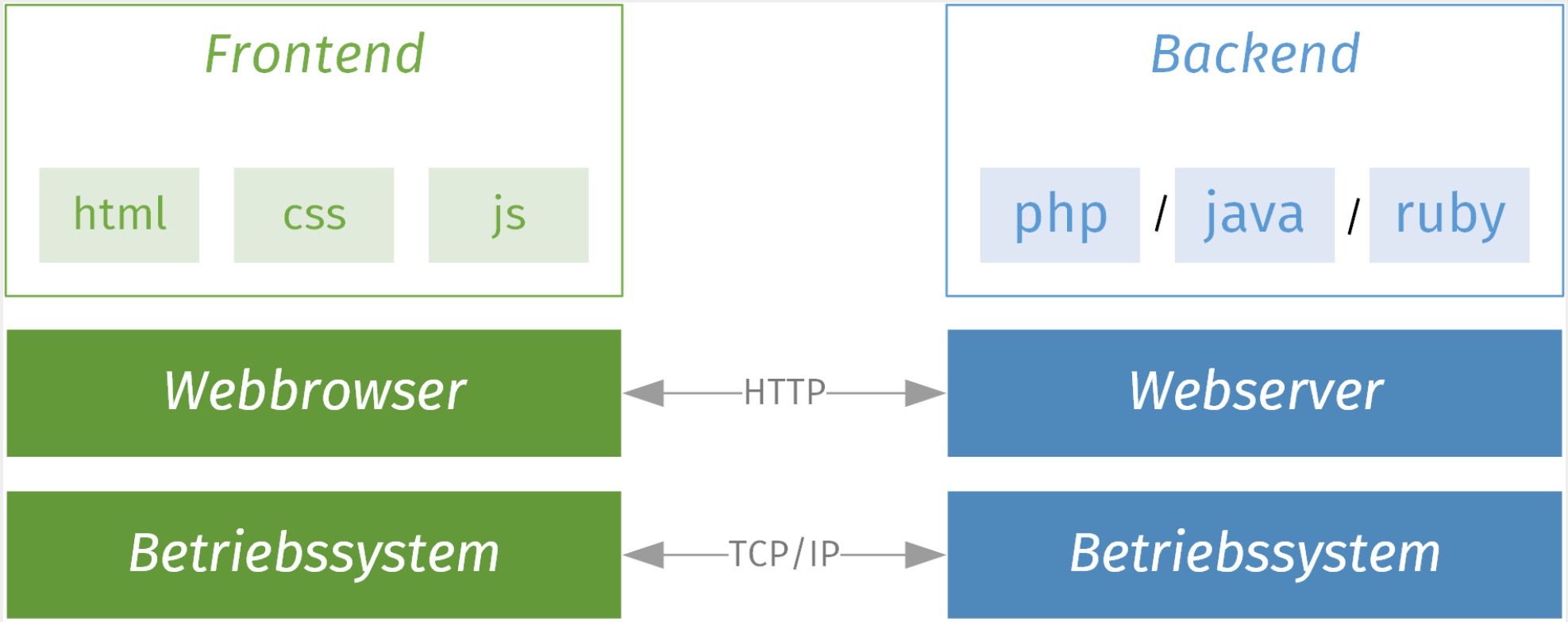
# Inhalt dieser Einheit

1. **Idee und kurze Geschichte**
   - Von SGML über HTML 4 zu HTML 5
2. **Begriffe, Format und Syntax**
   - Dokumente als Bäume, Elemente und Tags
3. **Textgestaltung- und strukturierung**
   - HTML für Inhalte
4. **Listen und Tabellen**
   - Strukturelemente (nicht nur) für Inhalte
5. **Formulare**
   - HTML für Nutzerschnittstellen

# Wdh.: Frontend vs. Backend

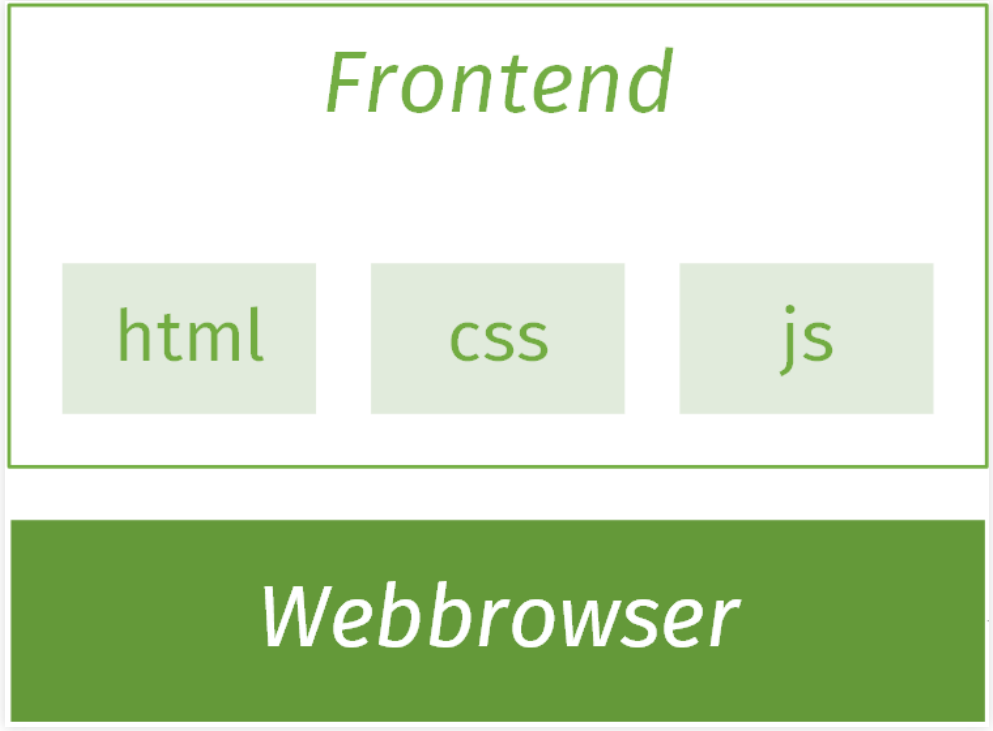
- **Frontend** (“Client”)
  - Inhalte, sowie nutzerseitig ausgeführte/interpretierte Anwendungsteile und Bedienschnittstelle
  - Umgebung: keine Kontrolle seitens der Entwickler
    - Verlassen auf Standards (W3C, WHATWG)
  - Webbrowser: kann “nur” HTML, CSS und JavaScript
- **Backend** (“Server”)
  - stellt Inhalte/Dienste bereit, ruft weitere “Wirkungen” hervor
  - letztlich: HTML, CSS und JavaScript als Ausgabe
  - Implementierung flexibel (z.B. Java, PHP, Ruby, Python, …)

# Wdh.: Frontend vs. Backend



# Themen der nächsten Einheiten

- Heute und nächste Woche (2. und 3. Einheit)
  - **HTML**: Darstellung von Inhalten im Browser, Strukturierung von Dokumenten
- Danach (4. Einheit):
  - **CSS**: optische Gestaltung von Inhalten
- Danach (5. Einheit):
  - **JavaScript**: client-seitige Programmierung



# Idee und kurze Geschichte

*Von SGML über HTML 4 zu HTML 5*

# Vor der Erfindung des Web

- **Problem**: Dokumente sind verstreut
  - Verweise auf andere Dokumente sind lediglich textuell
  - passendes Dokument muss mühsam von Hand gesucht werden
- **Lösung** (1): *Markup* (= “Aufschlag” bei Preisen)
  - Anreichern der Dokumente um besonderen Inhalte, um Verweise explizit als solche zu kennzeichnen (u.a.)
- **Lösung** (2): spezielle Software
  - eine, die sowohl Dokumente anzeigen kann,
  - als auch den Verweisen in den Dokumenten direkt zu neuen Dokumenten folgen kann.

# Die "Software"

> *Software, mit der sowohl Dokumente angezeigt, als auch direkt Verweisen zu weiteren Dokumenten gefolgt werden kann.*

- Wie könnte man das nennen?
- **Browser** (*to browse* = durchstöbern, schmökern)

# Hyper-, Hyper-, …

- **Hypertext**:
  - Text, der nicht zwangsläufig linear ist, bzw.
  - Text, der Verweise (oder Hyperlink) auf andere Texte beinhaltet
  - “hyper”: griechisch, “über” oder “über … hinaus”
- **Hyperlink**:
  - Verweis an einer Stelle eines Hypertexts auf ein anderes Dokument
    - oder eine konkrete Stelle darin
- **Hypertext Markup Language** oder **HTML**:
  - Sprache, um Hypertext zu verfassen und Hyperlinks zu setzen
    - “Auszeichnungssprache”, *keine* Programmiersprache

Quelle: https://www.w3.org/WhatIs.html

# SGML: Basis für HTML

- **Tim Berners-Lee** (ab 1989)
  - suchte einfache Auszeichnungssprache für sein WorldWideWeb
  - nutzte existierende Vorgaben zur Definition solcher Sprachen
    - *Standard Generalized Markup Language* oder **SGML** (*"Metasprache"*)
- SGML: Markup ist *deklarativ* und *präzise*
  - **deklarativ**: Beschreibung der Struktur des Dokuments statt des Verarbeitungsprozess
  - **präzise**: Sauber ausgezeichnete Dokumente, damit sie maschinell, also von Programmen verarbeitet werden können
- Sprache auf SGML-Basis:
  - Beschreibung der *Funktion/Rolle* von Textstücken
    - nicht deren Darstellung
  - Vorteil: unabhängig vom Formatierer/Betrachter (z.B. Browser)

# SGML-Beispiel (hier: DocBook)

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
<article>
   <sect1 id="introduction">
      <title>Hello world introduction</title>
      <para>
         Hello world!
      </para>
   </sect1>
</article>
```

- Teile eines SGML-Systems: (d.h. verarbeitet durch Parser)
  1. SGML-Deklaration (separates Dokument, definiert Zeichensatz)
     - SGML-Deklaration von HTML
  2. *Document Type Definition*, oder *DTD* (Verweis in Zeile 1)
  3. Dokument an sich, beginnend mit einem Element und seinen Kindern

Quelle: http://newbiedoc.sourceforge.net/tutorials/docbook-guide/basic-docbook-guide.html.en

# Versionen: HTML bis HTML 4.01

- Frühgeschichte von HTML ist verworren
  - 1989-1993: **HTML** (z.B: 11/1992, 01/1993)
    - am CERN entwickelt, ohne Versions-Schema
    - kannte Titel, Überschriften, Absätze, Links, Listen
  - 1993: **HTML+**
    - viele neue Ideen, wie Typografie, Bilder, Formulare, Tabellen, Dokumentenbeziehungen
- “Standards”
  - 1995: **HTML 2.0** bei der IETF, als RFC 1866
    - Ideen von HTML und HTML+
  - 1997: **HTML 3.2**, vom W3C
    - neu: Java Applets
  - 1999: **HTML 4.01**, vom W3C
    - JavaScript und Stylesheets, Frames

# Versionen: XHTML und HTML 5

- Wohlstrukturiertes HTML auf XML-Basis (statt SGML)
  - 2000: **XHTML 1.0** als W3C Recommendation
    - “Neuformulierung” von HTML 4 in XML
  - danach mehrere, praktisch kaum relevante XHTML-Standards
- ab 2004: WHATWG arbeitet an HTML5, W3C zieht nach
  - W3C: Versionierung in “Snapshots”
    - Prozess (Jahre!): *Working Draft* → *Candidate Recommendation* → *Proposed Recommendation* → *W3C Recommendation*
    - 2014: **HTML5** W3C Recommendation
    - Aktuell: **HTML 5.1 2nd Edition** W3C Recommendation
    - Pipeline: **HTML 5.2** Candidate Recommendation
  - fortlaufend: **HTML** als Living Standard der WHATWG
    - Standard wird fortgeschrieben, auf Basis von Rückmeldung von Webentwicklern und Browserherstellern
    - Standard ist den Implementierungen immer ein bisschen voraus

Quelle: https://whatwg.org/faq#living-standard

# Aufgabenverteilung

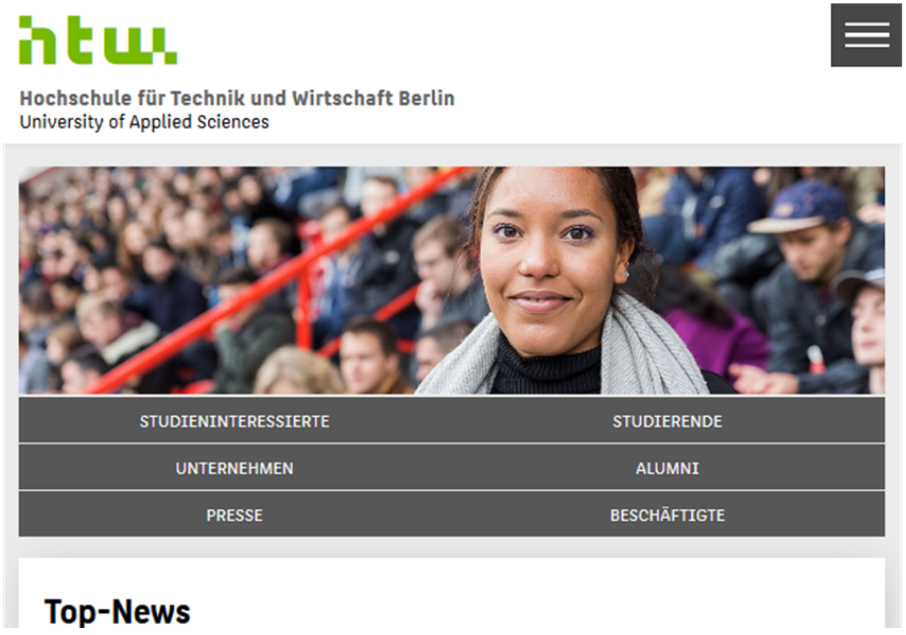
- **Browser** (Firefox, Chrome, Internet Explorer, Safari, …):
  - HTML-Dokumente für den Menschen grafisch anzeigen
- **Standards** (z.B. vom W3C & WHATWG):
  - Dokumentenaufbau festlegen, für konsistente Darstellung
- **Web-Entwickler/innen** (Sie!):
  - Dem Nutzer ermöglichen sein/ihr Ziel zu erreichen, d.h. hier
  - HTML-Dokumente erzeugen (lassen), die von Browsern in der gewünschten Form dargestellt werden
  - Dafür (u.a.):
    1. Wissen, was Browser so alles darstellen können
    2. Entscheiden, was wie dargestellt werden soll
    3. Wissen, wie man dies standard-konform umsetzt

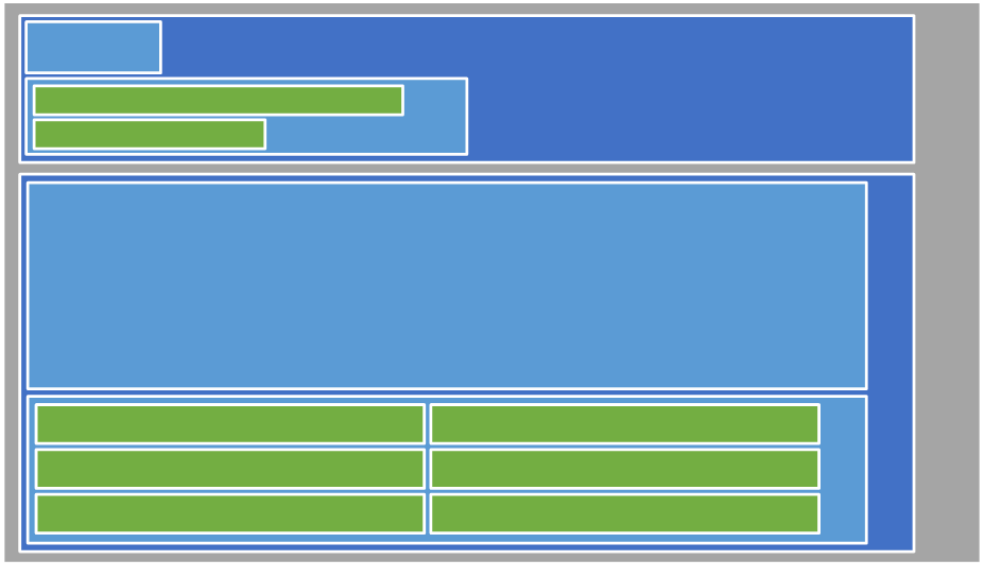
# Begriffe, Format und Syntax

*Dokumente als Bäume, Elemente und Tags*

# HTML-Dokumente: Baumstruktur

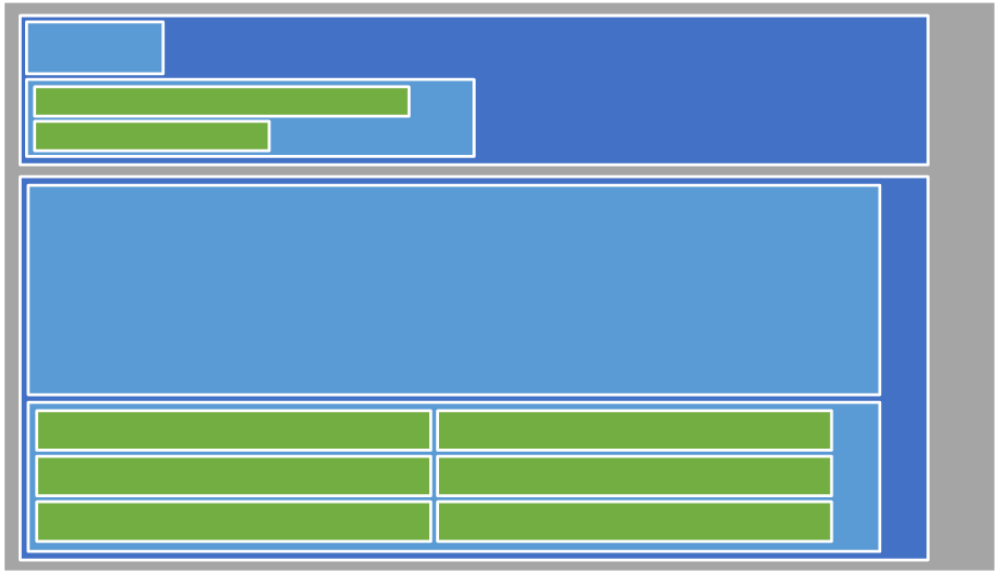
- Webseiten bestehen aus geschachtelten Boxen

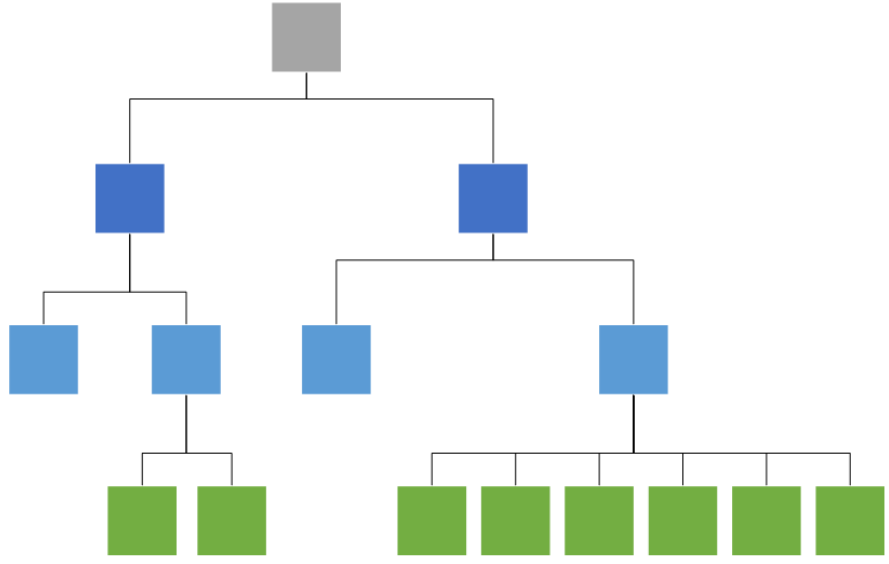


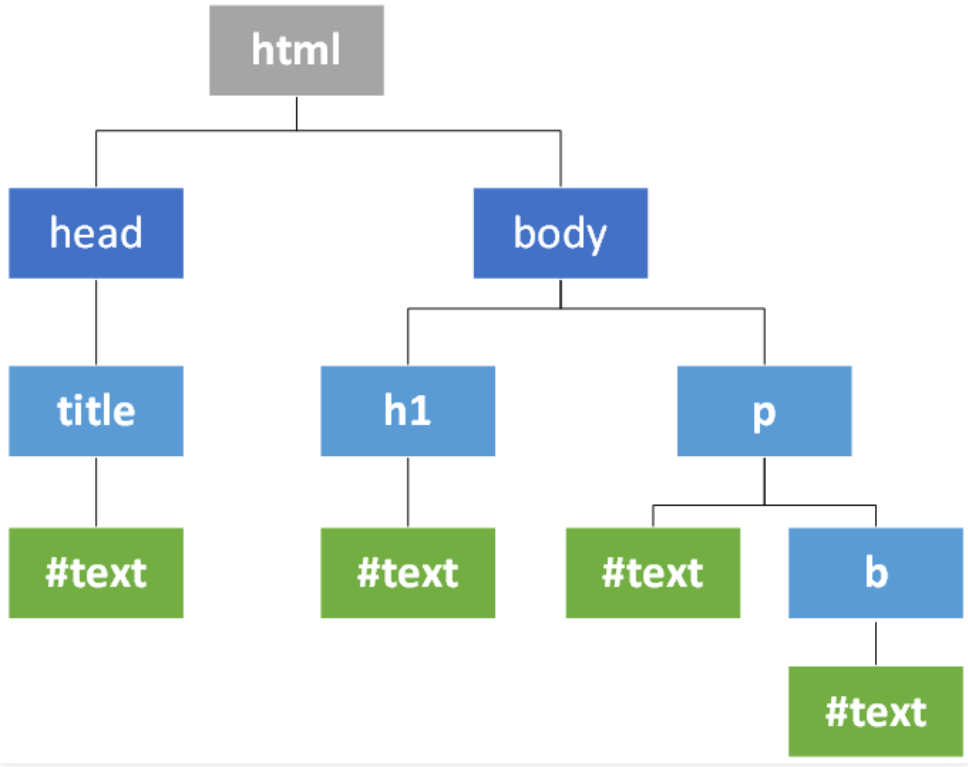
# HTML-Dokumente: Baumstruktur

- Geschachtelte Boxen lassen sich als Baum verstehen





# HTML-Dokumente: Baumstruktur




```
<html>
<head>
  <title>Webentwicklung</title>
</head>
<body>
  <h1>Webentwicklung</h1>
  <p>
    Willkommen auf der Webseite
    zur Veranstaltung
    <b>Webentwicklung!</b>
  </p>
</body>
</html>
```

# Konzepte & Begrifflichkeiten

1. Darstellung im Browser
   - (allermeistens) grafisch
2. HTML-**Dokument**:
   - konzeptionell (und im Arbeitsspeicher): ein Baum
     - **DOM**: Document Object Model
   - besteht aus **Elementen**
     - Wurzelknoten: `html`-Element
     - Jedes Element kann 0, 1, oder mehrere Kind-Elemente haben
3. HTML-**Quellcode**:
   - textuelle Repräsentation (Zeichenkette) des HTML-Dokuments
   - Anfang und Ende der Elemente jeweils durch **Tags** markiert

*Wie schreibt man denn nun*

# HTML-Code?

# Syntax: Tags

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

<!-- Start / Ende -->
Ein <strong>richtig</strong> schöner Tag

<!-- Void-Elemente ("leer") haben keinen Inhalt und keinen
     schließenden Tag, können aber selbstschließend sein -->
<hr> oder auch <hr />

<!-- Schachtelung -->
<h1>Die <strong>Chance</strong> des Tages</h1>
```

- HTML: sehr tolerant, z.B.
  - falsche Tag-Klammerung führt oft doch zu korrektem Baum
- XHTML: striktes Format, z.B.
  - Void-Elemente müssen selbstschließend sein: `<hr />`

# Syntax: Attribute

```
<!-- Schlüssel-Wert-Paar -->
<input value="Goku" placeholder="Name">

<!-- binäre Attribute (kann Wert haben, muss aber nicht) -->
<input disabled>

<!-- Mehrere Werte als leerzeichengetrennte Liste -->
<input class="aktiv obligatorisch hervorgehoben">

<!-- Custom Data Attributes -->
<span class="notifications" data-count="3">Benachrichtigungen</span>
```

- XHTML:
  - Alle Attribute brauchen einen Wert, auch binäre:
    - Korrekt: `<input disabled="disabled" />`

# Globale Attribute (alle Elemente)

- `id`
  - dokumentweit einzigartiger Name zur Identifizierung
  - *fragment identifier*, dadurch “verlinkbar”
- `class`
  - Klassifizierung / semantische Gruppierung mehrerer Elemente
- `title`
  - ergänzender Text zum Element (häufig für Tooltips benutzt)
- `style`
  - zur Anpassung der Optik (siehe Einheit zu CSS)

*Tags und Attribute, okay.*

# Und jetzt?

# HTML-Dokument: Grundgerüst

```
<!DOCTYPE html>
<html lang="de">
   <head>
      <meta charset="utf-8">
      <title> ... </title>
   </head>
   <body>
      ...
   </body>
</html>
```

- Teile
  1. Doctype
  2. HTML-Header
  3. HTML-Body

# 1. Doctype

```
<!DOCTYPE html>
```

- SGML- und XML-Dokumente
  - **wohlgeformt**: syntaktisch korrekt
  - **valide**: erlaubte Tags und Schachtelungen, entsprechend einer Definition (DTD, Document Type Definition)
- Doctype gibt “Geschmacksrichtung” des Dokuments an
  - für *Render-Modus*
- Doctypes von HTML 4.01, z.B.:

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

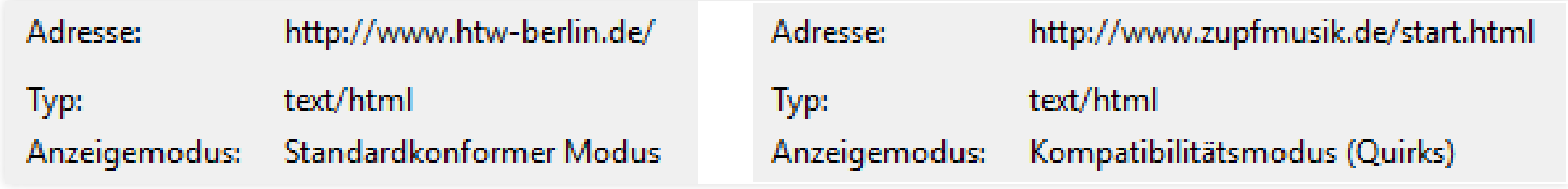
- früher häufig missachtet, teils von Browsern ignoriert

# Exkurs: Browser-Engines

- Arbeitsschritte (Links: definierte Algorithmen):
  1. Byte-Strom in Zeichen-Strom konvertieren (Zeichensatz)
  2. Strom in Tokens zerlegen (*start/end tag*, *character*, *EOF*, …)
  3. Baum aus Tokens erstellen (DOM, *Document Object Model*)
  4. Geometrien erstellen für grafische Darstellung (*Rendering*)
- Viele Details (etwas angestaubt, da von 2011):
  - html5rocks.com/…/howbrowserswork
- Beispiele für Engines
  - **Gecko**: z.B. Firefox
  - **Blink**: z.B. Chrome (früher Webkit), Vivaldi, Opera (früher Presto)
  - **Webkit**: z.B. Safari
  - **Trident/EdgeHTML**: Internet Explorer und Edge
- Wikipedia: Browser-Stammbaum nach Engine

# Exkurs: Render-Modi in Browsern

- Browser und erste Webseiten älter als W3C-Standards
  - Browser konnten nicht einfach die Standard-konforme Darstellung für alle Webseiten aktivieren
- Mehrere Render-Modi, z.B.:



| | |
|---|---|
| Adresse: | http://www.htw-berlin.de/ |
| Typ: | text/html |
| Anzeigemodus: | Standardkonformer Modus |

| | |
|---|---|
| Adresse: | http://www.zupfmusik.de/start.html |
| Typ: | text/html |
| Anzeigemodus: | Kompatibilitätsmodus (Quirks) |

  - **Quirks Mode**: wie Netscape 4/Internet Explorer 5
    - *quirk* = “Macke”
    - https://quirks.spec.whatwg.org/
  - **Standards Mode**: wie in Spezifikation vorgesehen

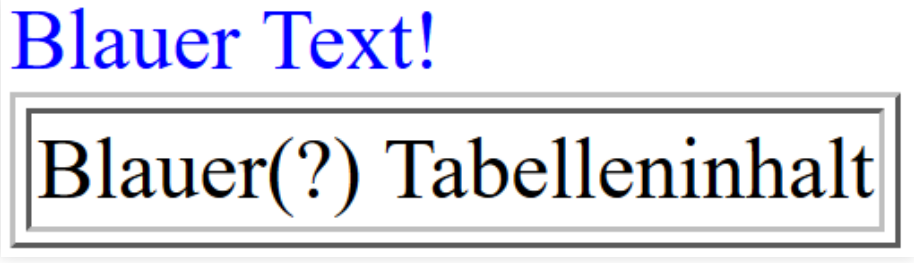
Quelle: https://developer.mozilla.org/de/docs/Quirks_Mode_and_Standards_Mode

# Exkurs: Vergleich der Render-Modi

- Ein deutlicher Unterschied (es gibt viele subtile):

```
<div style="color: blue">
  Blauer Text!
  <table border="1"><tr>
    <td>Blauer(?) Tabelleninhalt</td>
  </tr></table>
</div>
```

**Quirks Mode**




**Standards Mode**

# Exkurs: Ende

- Rat: Bei neuen Projekten, HTML5-Doctype
  - Nur dann können Ihre Dokumente im Standard-Modus dargestellt werden

```
<!DOCTYPE html>
```

- Wenn die DOM-Generierung doch robust ist: Warum sollte man sich an den Standard halten?
  - z.B. Barrierefreiheit
  - z.B. Suchmaschinen

# HTML-Dokument: Grundgerüst

```
<!DOCTYPE html>
<html lang="de">
   <head>
      <meta charset="utf-8">
      <title> ... </title>
   </head>
   <body>
      ...
   </body>
</html>
```

- Teile
  1. Doctype ✓
  2. HTML-Header
  3. HTML-Body

# 2. HTML-Header

```
<!-- Zusatzinformationen zum Dokument -->
<meta charset="utf-8"> <!-- Kodierung -->
<meta name="description" content="Hallo Google-Ergebnis-Liste.">
<meta http-equiv="refresh" content="5; URL=http://andere.domain.de/">

<title>Einführung in HTML</title> <!-- Dokumententitel -->

<!-- Verweis auf externe Ressourcen -->
<link rel="stylesheet" href="style.css">
<link rel="alternate" type="application/atom+xml" href="feed.atom">

<!-- Nicht zwingend nötig -->
<link rel="shortcut icon" href="favicon.ico">
```

- Weitere `meta`-Elemente:
  - https://developer.mozilla.org/de/docs/Web/HTML/Element/m

# Exkurs: Encoding

- HTML-Dateien (Endung `.htm` oder `.html`): textueller Inhalt
  - kein Binärformat, wie etwa DOCX, PDF
- Zeichensatz (“Encoding”, “Charset”)
  - früher: ISO 8859-1 oder Windows-1252 (“ASCII”, “ANSI”)
    - 1 Zeichen = 1 Byte = 8 Bit; d.h. 256 Zeichen
    - lateinisches Alphabet + ein paar Sonderzeichen
  - heute: UTF-8 (“Unicode”)
    - 1 Zeichen = 1-4 Byte = 8-32 Bit; d.h. 1+ Mio. Zeichen
    - “all the characters for all the writing systems of the world, modern and ancient”
- Rat: Verfassen Sie alle neuen Dokumente in Unicode
  - Konfigurieren Sie bei Bedarf Ihren Editor/IDE
  - Lesen: http://www.joelonsoftware.com/articles/Unicode.html

Quelle: http://www.unicode.org/faq/basic_q.html

# ISO 8859-1 als UTF-8 interpretiert


Quelle: https://www.getdigital.de/scheiss-encoding.html

# Sonderzeichen (Exkurs-Ende)

- Benannte **Entities** für geschützte Zeichen
  - und für Zeichen außerhalb des Zeichensatzes (vor Unicode)

| | |
|---|---|
| `< >` | < > |
| `&quot; &` | " & |
| `&auml; &szlig;` | ä ß |

- Darstellung von Unicode-Zeichen in dezimaler oder hexadezimaler Schreibweise

| | |
|---|---|
| `&#9728; &#x1F612;` | ☀ 😒 |

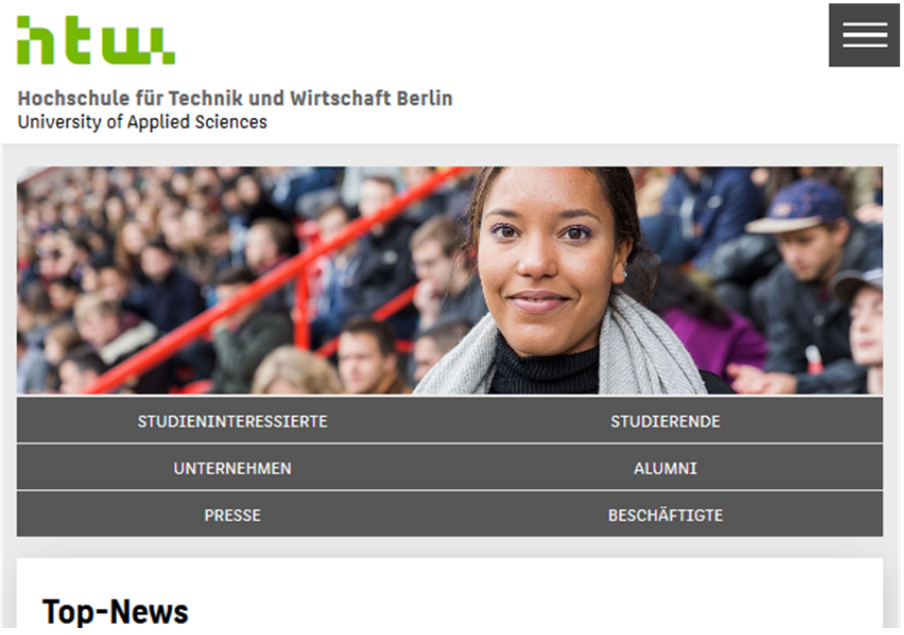
# HTML-Dokument: Grundgerüst

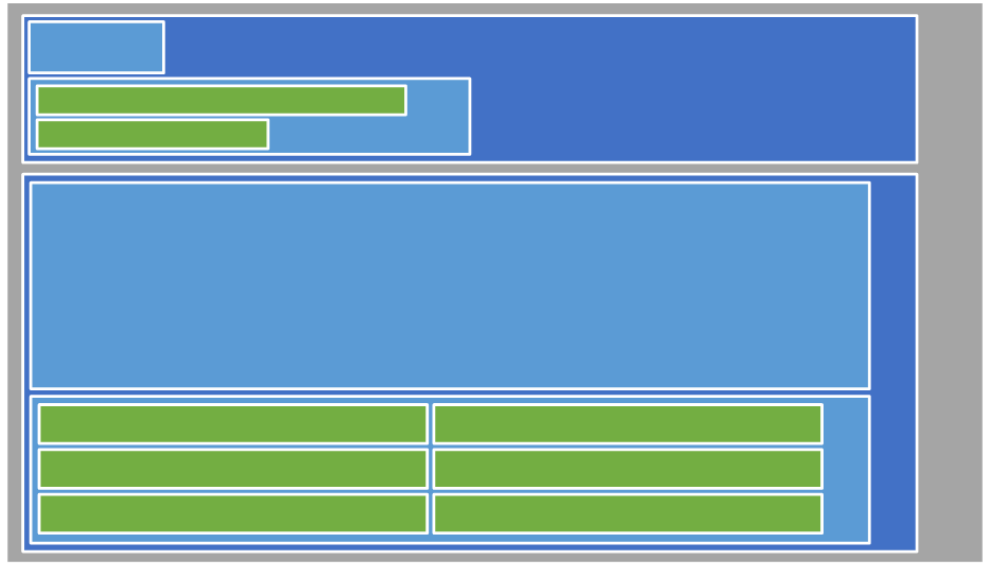
```
<!DOCTYPE html>
<html lang="de">
   <head>
      <meta charset="utf-8">
      <title> ... </title>
   </head>
   <body>
      ...
   </body>
</html>
```

- Teile
  1. Doctype ✓
  2. HTML-Header ✓
  3. HTML-Body

# 3. HTML-Body

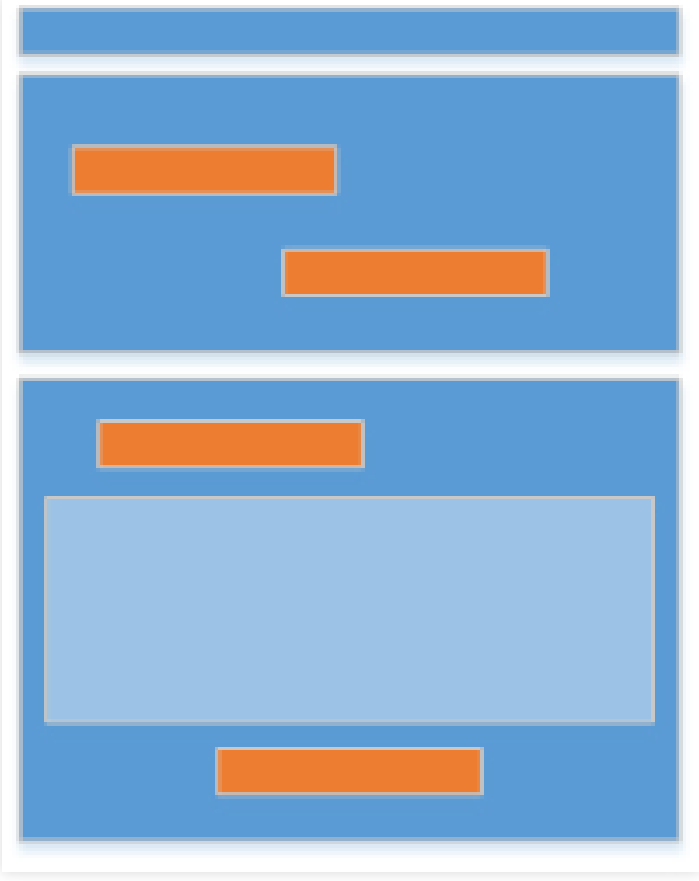
- `body`-Element: die Dokument-Inhalte an sich
  - ineinander geschachtelte Elemente verschiedener Arten



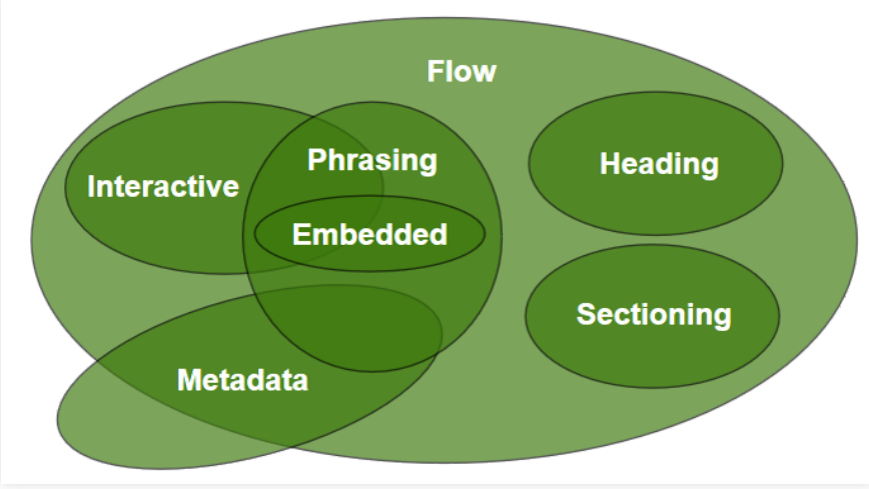



# Element-Arten

- Historisch (Layout): Block- vs. Inline-Elemente



- Heute (Semantisch): Element-Kategorien




Quelle: https://html.spec.whatwg.org/multipage/dom.html#kinds-of-content

# Element-Kategorien

- Kategorien werden auch “Content Models” genannt:

| Name | Beschreibung |
|---|---|
| Metadata | Definiert Präsentation oder Verhalten des restlichen Inhalts |
| Sectioning | Semantische Unterteilung (z.B. Gültigkeitsbereiche von Headern und Footern) |
| Heading | Definiert den Kopf eines Bereichs |
| Phrasing | Zeichnet den Text des Inhalts aus |
| Embedded | Bindet andere Ressourcen ein |
| Interactive | Inhalt, der zur Benutzer-Interaktion gedacht ist |

Quelle: https://html.spec.whatwg.org/multipage/dom.html#kinds-of-content

*Konkreter bitte: Welche*

# Elemente

*gibt es denn nun?*

# Textgestaltung und -strukturierung

*HTML für Inhalte*

# Gliederung: Überschriften

```HTML
<h1>Eins</h1>
<h2>Zwei</h2>
<h3>Drei</h3>
<h4>Vier</h4>
<h5>Fünf</h5>
<h6>Sechs</h6>
```

# Gliederung: Absätze

```HTML
<p>Das ist ein Absatz.</p> <p>Das ist ein weiterer Absatz</p>
<p>       Whitespaces spielen   im Allgemeinen
    in   HTML keine Rolle.</p>
```

Das ist ein Absatz.

Das ist ein weiterer Absatz

Whitespaces spielen im Allgemeinen in HTML keine Rolle.

# Exkurs: Whitespaces

- Whitespaces (d.h. Leerzeichen, Zeilenumbrüche, Tabs) im Quelltext werden als Leerzeichen interpretiert
  - Mehrere Whitespaces hintereinander wirken wie ein Leerzeichen
    - (das passiert bei der DOM-Generierung)
- Whitespaces erzwingen:
  - ` ` für einzelnes Leerzeichen *(non-breaking space)*
  - `<br />` für Zeilenumbruch *(break)*

# Exkurs: Whitespaces bewahren

- Whitespaces bewahren (*preformatted*):

```HTML
<pre>          Mir ist   jedes
            Leerzeichen
     wichtig.</pre>
```

```
          Mir ist   jedes
             Leerzeichen
      wichtig.
```

# Formatierung von Text-Inhalten

```HTML
Das muss <strong>dringend</strong> erledigt werden.
<em>Du</em> wolltest doch <mark>mindestens</mark> 70m<sup>2</sup>!
Das Wort <i>das</i> ist ein <b>Artikel</b>.
<small>Copyright <del>2014</del> 2015 Foo Inc.</small>
```

Das muss **dringend** erledigt werden. *Du* wolltest doch mindestens 70m$^2$!
Das Wort *das* ist ein **Artikel**. Copyright ~~2014~~ 2015 Foo Inc.

# Links und Bilder

```HTML
<img src="wikipedia.png" alt="Wikipedia">
<!-- explizte Größenangaben verhindern Sprünge beim Nachladen
     von Ressourcen -->
<img src="nichts.jpg" style="width: 135px; height: 155px"
  alt="Wikipedia">
<a href="http://de.wikipedia.org/">Wikipedia</a>
```


Wikipedia

[Wikipedia](http://de.wikipedia.org/)

# Links und Bilder

```HTML
<a href="http://wikipedia.org/"><img src="wikipedia.png"></a>

<br>

<a href="http://wikipedia.org/" target="_blank">Wikipedia</a>
<small>(öffnet neuen Tab)</small>
```


Wikipedia (öffnet neuen Tab)

# Linkziele: Anker

```
<!-- Anker ("a") explizit setzen -->
<h1><a name="overview"></a> Übersicht</h1>
<!-- Elemente mit id-Attribut funktioneren auch -->
<h2 id="faq">FAQ</h2>

<!-- ... -->

<a href="#overview">Zur Übersicht</a>
<a href="#faq">zu den FAQ</a>
```

# Listen und Tabellen

*Strukturelemente (nicht nur) für Inhalte*

# Listen: Zwei Haupttypen

1. Unnummerierte Listen `ul` *(unordered list)*
2. Nummerierte Listen `ol` *(ordered list)*

```HTML
<ul>
  <!-- li = "list item" -->
  <li>Tick</li>
  <li>Trick</li>
  <li>Track</li>
</ul>

<ol>
   <li>Schlafen</li>
   <li>Essen</li>
   <li>Lernen</li>
</ol>
```

- Tick
- Trick
- Track

1. Schlafen
2. Essen
3. Lernen

# Listen: Dritter Typ

- Definitions- oder Beschreibungsliste `dl` *(definition list)*

```HTML
<dl>
  <dt>HTTP</dt>
  <dt>HTTPS</dt>
  <dd>Hypertext Transfer Protocol</dd>
  <dt>HTML</dt>                          <!-- definition term -->
  <dd>Hypertext Markup Language</dd> <!-- definition description -->
</dl>
```

# Listen: Schachtelung

```HTML
<ul>
  <li>Vorlesungen
    <ol><li>Intro</li>
        <li>HTML</li>
    </ol>
  </li>
  <li>Übungen
    <ol><li>Arbeitsumgebung
          einrichten</li>
        <li>Einfache
          Webseite</li>
    </ol>
  </li>
</ul>
```

- Vorlesungen
  1. Intro
  2. HTML
- Übungen
  1. Arbeitsumgebung einrichten
  2. Einfache Webseite

# Tabellen: Grundelemente

- Eine Tabelle `table` besteht aus Zeilen `tr` *(table row)*
- Zeilen bestehen aus Zellen `td` *(table data)*

```HTML
<table>
  <tr>
    <td>A1</td> <td>A2</td> <td>A3</td>
  </tr>
  <tr>
    <td>B1</td> <td>B2</td> <td>B3</td>
  </tr>
</table>
```

| | | |
|---|---|---|
| A1 | A2 | A3 |
| B1 | B2 | B3 |

# Tabellen: Viele Möglichkeiten

- Zellen verbinden
  - `td colspan="2"` bzw. `rowspan`
- Spalten- oder Zeilenüberschriften
  - `th` *(table heading)* statt `td`
- Tabelle unterteilen
  - `thead`, `tbody`, und `tfoot` zur Gruppierung von mehreren `tr`s
- Tabelle beschriften
  - `caption`

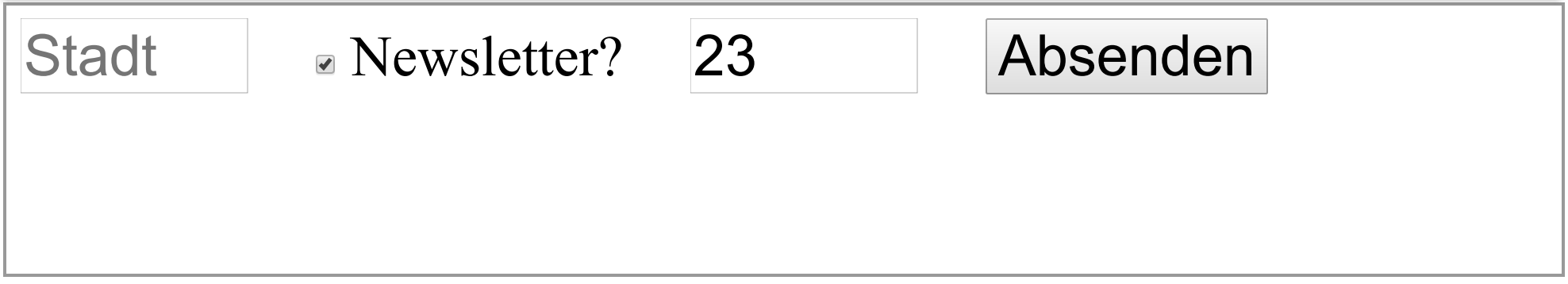
Details hierzu: A Complete Guide to the Table Element

# Formulare

*HTML für Nutzerschnittstellen*

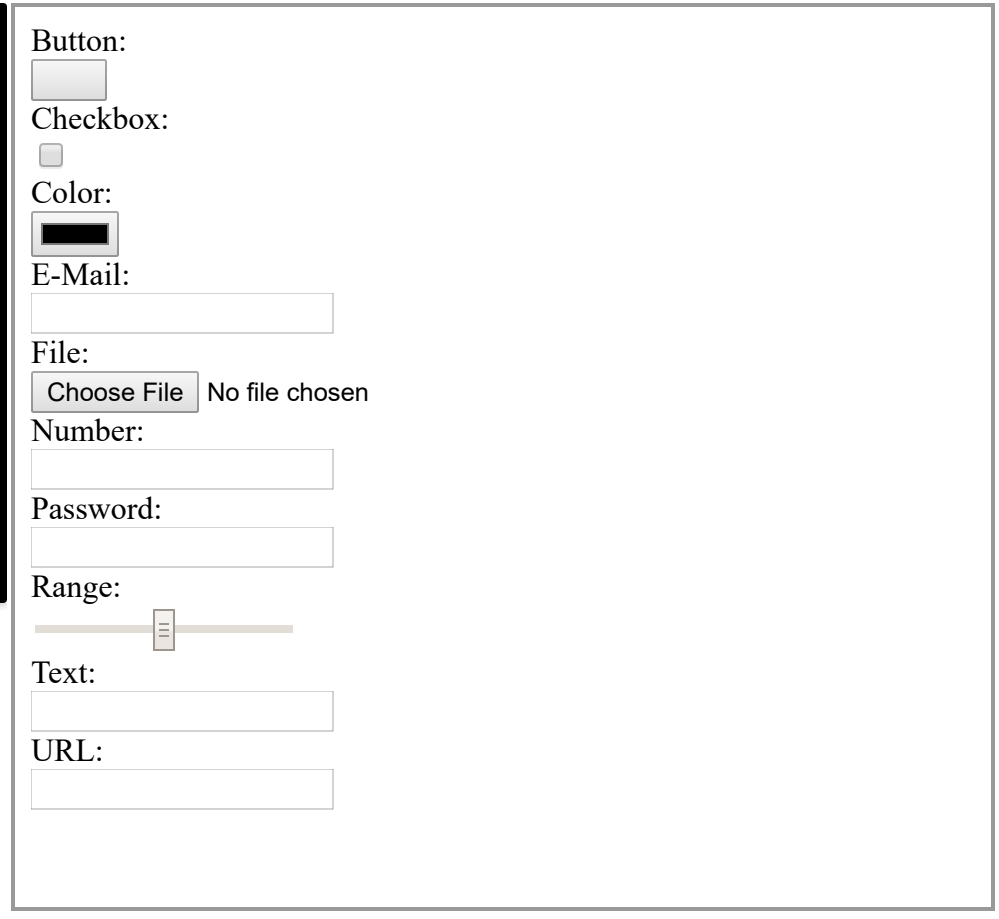
# Formular

```HTML
<form action="index.php" method="POST">
   <input type="text" name="stadt" placeholder="Stadt">

   <input type="checkbox" id="nl" name="newsletter" checked>
   <label for="nl">Newsletter?</label>

   <input type="number" name="alter" value="23">

   <button type="submit">Absenden</button>
</form>
```



# Formulare: Input-Typen

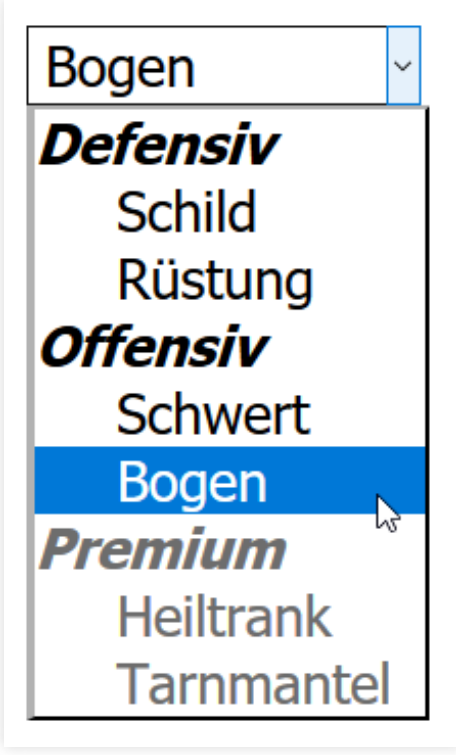
```HTML
Button: <input type="button">
Checkbox: <input type="checkbox">
Color: <input type="color">
E-Mail: <input type="email">
File: <input type="file">
Number: <input type="number">
Password: <input type="password">
Range: <input type="range">
Text: <input type="text">
URL: <input type="url">
```




- Alle Typen:
  - https://html.spec.whatwg.org/multipage/input.html#attr-input-type-keywords

# Formulare: Auswahllisten

```
<select name="produkte">
   <optgroup label="Defensiv">
      <option>Schild</option>
      <option>Rüstung</option>
   </optgroup>
   <optgroup label="Offensiv">
      <option>Schwert</option>
      <option selected>Bogen</option>
   </optgroup>
   <optgroup label="Premium"
      disabled>
      <option>Heiltrank</option>
      <option>Tarnmantel</option>
   </optgroup>
</select>
```



# Zusammenfassung: Heutige Inhalte

- Aufgabe des Webbrowser, Zusammenhänge:
  - Darstellung im Browser
    ⇄ HTML-Dokumentenbaum (Elemente)
    ⇄ HTML-Quellcode (Tags)
    (⇄ Bytestrom)
- allgemeine HTML-Syntax
  - Elemente zur Dokumentenstrukturierung
  - Elemente zur Textstrukturierung
- HTML-Standards
  - Living Standard vs. Snapshots

# Literaturhinweise

- HTML-Spezifikationen
  - HTML 5.1 – W3C Recommendation
  - WHATWG HTML Living Standard

- Zugeschnitten für Praktiker:
  - SelfHTML, inkl. Element-Übersicht
  - HTML-Einführung von Mozilla (Deutsch):
    - Einheiten 1, 2, 3, 4, 5
  - weitere Mozilla-Module (Englisch):
    - Medieneinbindung
    - Tabellen
    - Formulare

# Danke!