

# HTML & CSS

Dr. F.-K. Koschnick, Sybit GmbH

# Hypertext Markup Language (HTML)

- 1989: Tim Berners-Lee (CERN) beschreibt Grundzüge des WWW
- 1992: Tim Berners-Lee stellt ersten Entwurf zu HTML vor
- 1994: MIT und CERN vereinbaren Gründung der W3 Organisation
- ....
- 2017: HTML 5.2 (W3C Recommendation, 14 December 2017)  
<https://www.w3.org/TR/html5>

# Hypertext Transfer Protocol (HTTP)

- Basiert auf TCP (transmission protocol)
- Standardport ist 80, für https ist es 443
- Zustandsloses Protokoll (nur isolierte Transaktionen)
  - Sessions müssen also mit Zusatzmechanismen verwaltet werden (z.B. Cookies)
- Request – Response – Muster
- Neun definierte http-Methoden

# HTTP - Methoden

Methode	Beschreibung
OPTIONS	prüft, welche Methoden auf einer Ressource zur Verfügung stehen.
TRACE	gibt die Anfrage zurück, wie sie der Zielservers erhält.
HEAD	fordert Metadaten der angegebenen Ressource an. HEAD entspricht also GET, nur dass kein Nachrichtenrumpf erwartet wird.
GET	fordert die angegebene Ressource vom Server an. GET weist keine Nebeneffekte auf, d.h. der Zustand des Servers wird nicht verändert.
DELETE	fordert die Löschung der angegebenen Ressource.
PUT	PUT fordert das Anlegen einer Ressource unter dem angegebenen URI mit den angehängten Daten. Wenn die Ressource bereits existiert, wird sie geändert. Ein erfolgreiches Anlegen einer neuen Ressource wird mit dem Statuscode 201 (Created) angezeigt, im Falle einer geänderten Ressource 200 (OK) oder 204 (No Content).
PATCH	PATCH ein Teil der angegebenen Ressource wird geändert. PATCH bewirkt also Nebeneffekte auf dem Server. (Definiert seit Juni 2010 RFC 5789.)
POST	übermittelt Daten an die angegebene Ressource, die der Server weiterverarbeiten soll. Das Ergebnis dieser Verarbeitung kann eine neue Ressource sein, deren URI der Server mit dem Statuscode 201 (Created) zurücksendet oder mit 200 (OK) bzw. 204 (No Content) quittiert.
CONNECT	um eine HTTPS-Verbindung über einen HTTP-Proxy herzustellen.

# HTTP - Methoden

Methode	Beschreibung
OPTIONS	prüft, welche Methoden auf einer Ressource zur Verfügung stehen.
TRACE	gibt die Anfrage zurück, wie sie der Zielservers erhält.
HEAD	fordert Metadaten der angegebenen Ressource an. HEAD entspricht also GET, nur dass kein Nachrichtenrumpf erwartet wird.
GET	fordert die angegebene Ressource vom Server an. GET weist keine Nebeneffekte auf, d.h. der Zustand des Servers wird nicht verändert.
DELETE	fordert die Löschung der angegebenen Ressource.
PUT	PUT fordert das Anlegen einer Ressource unter dem angegebenen URI mit den angehängten Daten. Wenn die Ressource bereits existiert, wird sie geändert. Ein erfolgreiches Anlegen einer neuen Ressource wird mit dem Statuscode 201 (Created) angezeigt, im Falle einer geänderten Ressource 200 (OK) oder 204 (No Content).
PATCH	PATCH ein Teil der angegebenen Ressource wird geändert. PATCH bewirkt also Nebeneffekte auf dem Server. (Definiert seit Juni 2010 RFC 5789.)
POST	übermittelt Daten an die angegebene Ressource, die der Server weiterverarbeiten soll. Das Ergebnis dieser Verarbeitung kann eine neue Ressource sein, deren URI der Server mit dem Statuscode 201 (Created) zurücksendet oder mit 200 (OK) bzw. 204 (No Content) quittiert.
CONNECT	um eine HTTPS-Verbindung über einen HTTP-Proxy herzustellen.

**REST** (Red background): GET, DELETE, PUT, POST

**HTML** (Blue background): OPTIONS, TRACE, HEAD, PATCH, CONNECT

# HTTP: Transaktion

Eine Transaktion in HTTP funktioniert nach dem Request-Response-Muster und besteht jeweils aus den folgenden vier Schritten:

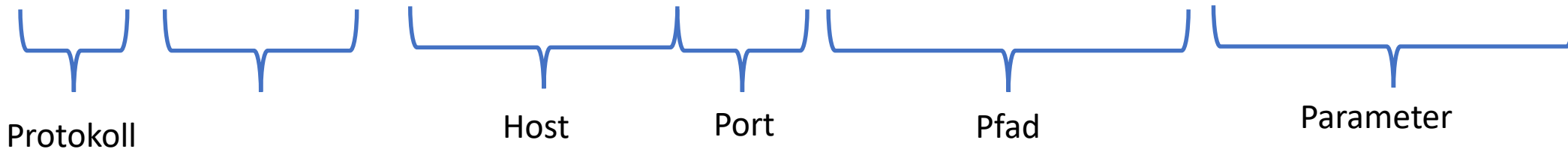
1. Verbindungsaufbau
2. **Anfrage (request)**, ausgehend vom Client
3. **Antwort (response)** des Servers
4. Verbindungsabbau

Damit ist HTTP zustandslos (stateless)! Im Prinzip vergisst der Server nach jedem Response, wer der Client war. Man kann das mit Cookies oder SessionIds etc. umgehen.

*Bemerkung für „ganz Schlaue“:* Auch Keep-Alive, ein Aufrechterhalten der Verbindung, ändert an der Zustandslosigkeit nichts.

# Aufbau der URL, URL Encoding

http://user:pw@www.xyz.de:8080/pfad/index.html?p1=abc&p2=def#ressource



Authentifizierung

Schlecht,  
besser mit post

## URL Encoding

_	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	:	;	<	=	>	?	@	[	\	]	{		}
%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	4	5	5	5	7	7	7
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	A	B	C	D	E	F	0	B	C	D	B	C	D

# HTTP: Response Code

Code	Beschreibung
200	OK
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

1XX – Informationen

2XX – Erfolgreiche Operationen

3XX – Umleitung

4XX – Client-Fehler

5XX – Server-Fehler



# Begriff „Markup“

- Begriff aus Druckindustrie: Layouter fügt Anmerkungen/Markierungen (Tags) hinzu
- Markup-Sprache (ML) = Auszeichnungssprache
- Beispiele
  - das Wort wird **<b>fett</b>** dargestellt
  - \section{Überschrift 1}
  - <h1>Große Überschrift</h1>

# Dokumente

- (Text-) Dokumente bestehen aus
  - Struktur
    - Kapitel, Abschnitte, Verweise, Fußnoten, Aufzählungen usw.
  - Inhalt/Daten
    - Text, Bilder, Audio, Video
  - Format/Darstellung
    - Schriftarten, -größen, -formate, Farben, Positionen

# Trennung von Layout und Daten

- (Text-) Dokumente bestehen aus

- Struktur

- Kapitel, Abschnitte, Verweise, Fußnoten, Aufzählungen usw.

- Inhalt/Daten

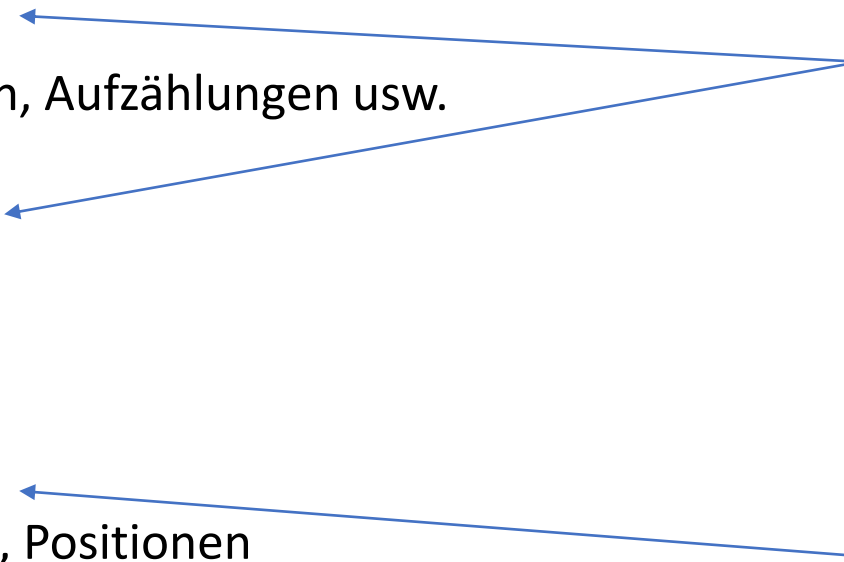
- Text, Bilder, Audio, Video

- Format/Darstellung

- Schriftarten, -größen, -formate, Farben, Positionen

HTML

CSS



# HTML-Dokument

```
<!DOCTYPE html>  
<html>  
  <head>  
  </head>  
  <body>  
  </body>  
</html>
```

<!DOCTYPE html>

Deklaration (HTML5), einmal am Anfang des Dokuments, nicht case sensitive

<tagname>Inhalt ...</tagname>

wohlgeformt

<html> ... </html>

Wurzelement

<head>... </head>

Metadaten

<body>... </body>

sichtbarer Teil des Dokuments

# HTML-Dokument

```
<!DOCTYPE html>
<html>
  <head>
    <title>Einfache HTML-Seite</title>
  </head>
  <body>
    <h1>Große Überschrift</h1>
    <h2>Kleinere Überschrift</h2>
    <p>Ein Abschnitt, in dem Text stehen kann und dem eine Liste folgt.</p>
    <ul>
      <li>Erster Wert der Liste</li>
      <li>Zweiter Wert der Liste</li>
    </ul>
    <p>Noch ein Absatz mit einem <strong>fetten Wort</strong> und einem <em>kursiven Wort</em> und einem Link zur
      <a href="https://www.sybit.de">Sybit GmbH</a>.
    </p>
  </body>
</html>
```

Head

Body

Tags

Attribute

Gut erklärt!

## Große Überschrift

### Kleinere Überschrift

Ein Abschnitt, in dem Text stehen kann und dem eine Liste folgt.

- Erster Wert der Liste
- Zweiter Wert der Liste

Noch ein Absatz mit einem **fetten Wort** und einem *kursiven Wort* und einem Link zur [Sybit GmbH](https://www.sybit.de).

# HTML-Elemente

`<tagname attribut1="xyz" attribut2="abc">Textinhalt</tagname>` (Element mit Inhalt)

`<tagname attribut1="xyz" attribut2="abc" />` (leeres Element)

Beispiel: Bild (img-Tag, leeres Element)

``

## TIPPS:

- Leere Elemente immer schließen (`<tagname />`), Beispiel `<br>` -> `<br />`  
(Wohlgeformt wegen möglicher Parser)
- Bei img-Tag möglichst immer Attribut *alt* verwenden  
(1. Barrierefreiheit / Screen Reader für Blinde und 2. Alternativer Text, falls Bild nicht angezeigt werden kann)

# HTML: Tags \*Auszug

- Dokument <html>, <head>, <body>, <title>, <meta>
- Listen <ul>, <ol>, <li>
- Gliederung <h1>..<h4>, <p>, <br>
- Bereiche <div>, <span>
- Links <a href="http://XX">, <a href="mailto:ab@cd.ef">
- Auszeichnung <pre>, <code>, <em>, <strong>, <cite>
- Formatierung <table>, <tr>, <td>
- Formulare <form action="XY">, <input>, <select>, <option>
- Bilder 

# HTML: Beispiel Tabelle

```
<table border="1">
```

```
<tr>
```

```
  <th>Spalte 1</th> }  
  <th>Spalte 2</th> }
```

```
</tr>
```

```
<tr>
```

```
  <td>Zeile 1, Spalte 1</td> }  
  <td>Zeile 1, Spalte 2</td> }
```


```
</tr>
```

```
<tr>
```

```
  <td>Zeile 2, Spalte 1</td> }  
  <td>Zeile 2, Spalte 2</td> }
```

```
</tr>
```

```
</table>
```



Spalte 1	Spalte 2
Zeile 1, Spalte 1	Zeile 1, Spalte 2
Zeile 2, Spalte 1	Zeile 2, Spalte 2

tr: Table Row  
th: Table Header  
td: Table Data



# HTML: Beispiele in w3 HTML-Tutorial

<https://www.w3schools.com/html/>

# HTML: Beispiel Formular

```
<form action= "relative/url" method="POST">  
  <input type="text" name="anmeldename" id="an" placeholder="Name"/>  
  <input type="password" name="passwort" id="pwd"/>  
  
  <input type="submit" value="Abschicken"/>  
  <input type="reset" value="zurücksetzen"/>  
</form>
```

# HTML: Sonderzeichen

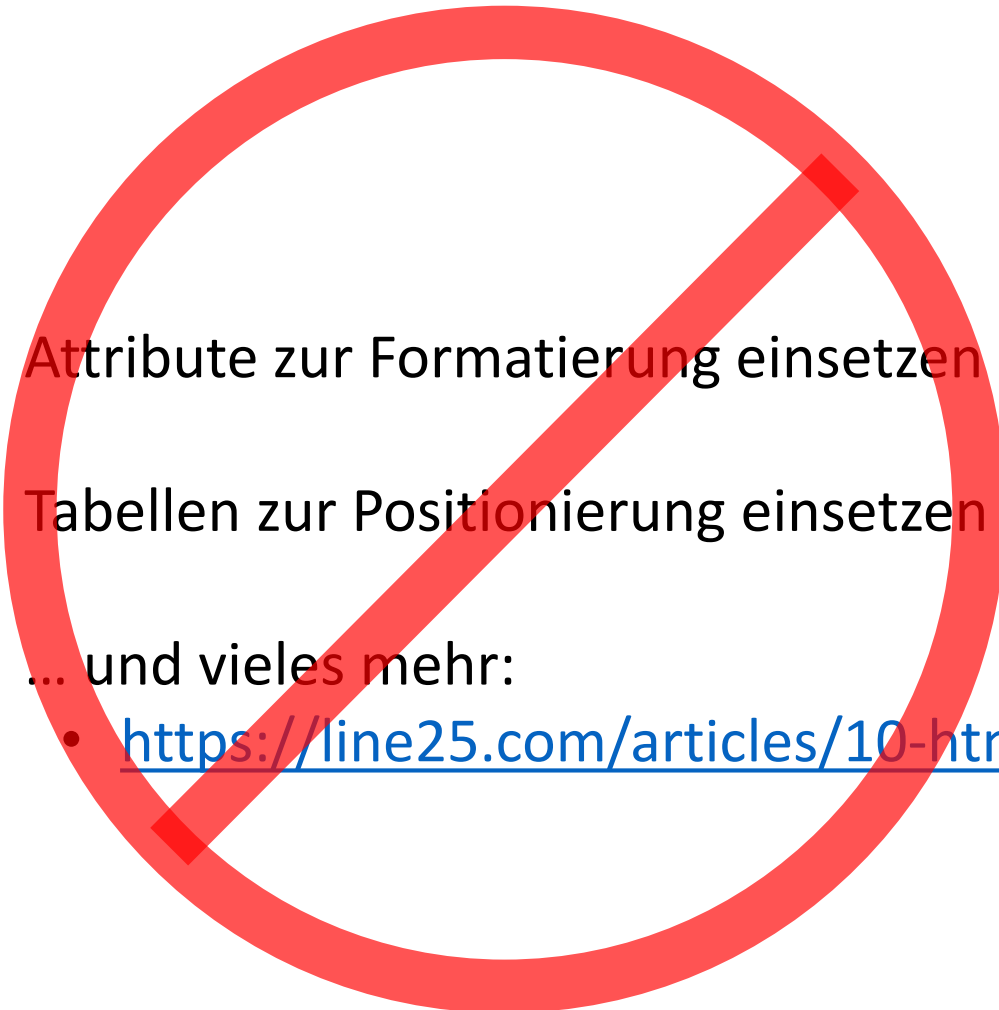
Zeichen	HTML-Befehl	Erläuterung
ä	&auml;	a Umlaut
Ä	&Auml;	A Umlaut
ß	&szlig;	sz Ligatur
€	&euro;	Euro
φ (phi klein)	&#966; &#x3C6; &phi;	966 = x3C6 Unicode von φ (phi klein)

<https://wiki.selfhtml.org/wiki/Referenz:HTML/Zeichenreferenz>

**BESSER:** UTF-8 oder ISO-8859-1

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
  </head>
  .....
```

# HTML, don't dos

- 
- Attribute zur Formatierung einsetzen
  - Tabellen zur Positionierung einsetzen
  - ... und vieles mehr:
    - <https://line25.com/articles/10-html-tag-crimes-you-really-shouldnt-commit>

# Trennung von Layout und Daten -> CSS

- (Text-) Dokumente bestehen aus

- Struktur

- Kapitel, Abschnitte, Verweise, Fußnoten, Aufzählungen usw.

- Inhalt/Daten

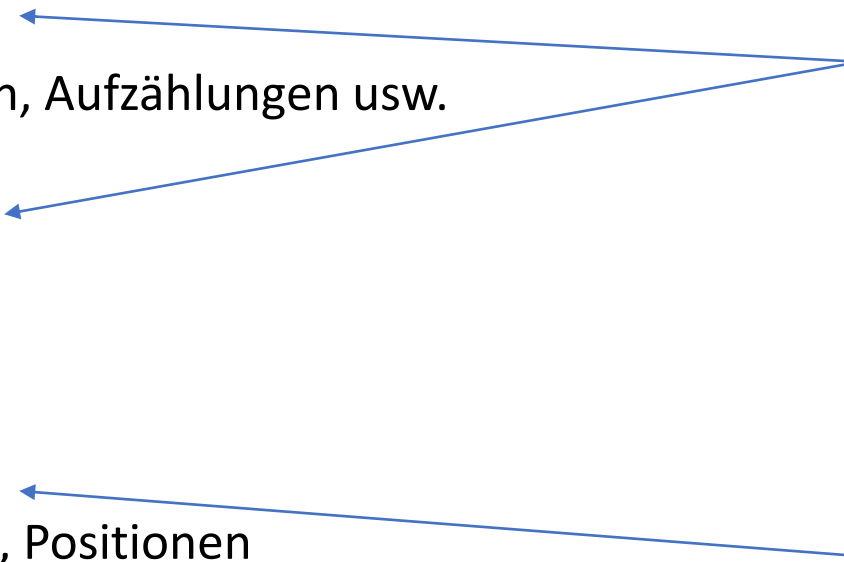
- Text, Bilder, Audio, Video

- Format/Darstellung

- Schriftarten, -größen, -formate, Farben, Positionen

HTML

CSS



# Cascading Style Sheets (CSS)

TIPP 1:  
**link** ist schneller,  
daher besser

- Festlegung von Layout/Formatierung
- Werden im Head definiert (extern/intern)

TIPP 2:  
CSS immer am Anfang  
des Headers einbinden,  
Darstellung schneller

## Extern: Option 1

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

HTML5

## Extern: Option 2

```
<head>  
  <style type="text/css">@import "style.css";</style>  
</head>
```

## intern

```
<head>  
  <style type="text/css">  
    <!--  
      HIER_DIE_ANGABEN  
    -->  
  </style>  
</head>
```

# HTML Selbststudium (Tutorium)

Browser address bar: [https://www.w3schools.com/html/html\\_tables.asp](https://www.w3schools.com/html/html_tables.asp)

Navigation bar: Apps, Projectile Prod, Anmelden - Sybit Inf, System Dashboard, Zentraler Jenkins, Sybit Artifactory, SonarQube, Talentsoft Prodsyste, Talentsoft Testsystem, Documentation - Em, Jenkins Slave Nodes, SSH Tutorial for Win

Menu bar: HTML, CSS, JAVASCRIPT, SQL, PHP, BOOTSTRAP, HOW TO, JQUERY, W3.CSS, ANGULARJS, XML, MORE, REFERENCES

## HTML5 Tutorial

- HTML HOME
- HTML Introduction
- HTML Editors
- HTML Basic
- HTML Elements
- HTML Attributes
- HTML Headings
- HTML Paragraphs
- HTML Styles
- HTML Formatting
- HTML Quotations
- HTML Comments
- HTML Colors
- HTML CSS
- HTML Links
- HTML Images
- HTML Tables**
- HTML Lists
- HTML Blocks
- HTML Classes
- HTML Id
- HTML Iframes
- HTML JavaScript
- HTML File Paths

## HTML Tables

[< Previous](#) [Next >](#)

### HTML Table Example

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

[Try it Yourself >](#)

### Defining an HTML Table

# HTML (try it yourself)

Run >

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  font-family: arial, sans-serif;
  border-collapse: collapse;
  width: 100%;
}

td, th {
  border: 1px solid #dddddd;
  text-align: left;
  padding: 8px;
}

tr:nth-child(even) {
  background-color: #dddddd;
}
</style>
</head>
<body>

<h2>HTML Table</h2>

<table>
<tr>
  <th>Company</th>
  <th>Contact</th>
  <th>Country</th>
</tr>
<tr>
  <td>Alfreds Futterkiste</td>
  <td>Maria Anders</td>
  <td>Germany</td>
</tr>
<tr>
  <td>Centro comercial Moctezuma</td>
  <td>Francisco Chang</td>
  <td>Mexico</td>
</tr>
<tr>
  <td>Ernst Handel</td>
```

style-Tag,  
Trennung von  
Inhalt und Layout

Result Size: 1298 x 777

### HTML Table

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy



# CSS (CSS Zen Garden)



## CSS ZEN GARDEN

*The Beauty of CSS Design*

[VIEW ALL DESIGNS](#)



A demonstration of what can be accomplished through CSS-based design. Select any style sheet from the list to load it into this page.

Download the example  HTML FILE and  CSS FILE

### THE ROAD TO ENLIGHTENMENT

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, broken CSS support, and abandoned browsers.

We must clear the mind of the past. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WASP, and the major browser creators.

MID CENTURY MODERN

*by* Andrew Lohman

GARMENTS

*by* Dan Mall

STEEL

*by* Steffen Knoeller

APOTHECARY



<http://www.csszengarden.com> (CSS Zen Garden)

# CSS (CSS Zen Garden)



<http://www.csszengarden.com> (CSS UNDER THE SEA)

# CSS: Wirkung auf HTML

[https://www.w3schools.com/css/css\\_intro.asp](https://www.w3schools.com/css/css_intro.asp)

# CSS: Syntax

- Syntax

**selector** {**css-element**:**wertangabe**;}

- Beispiele

**h1** {**font-family**:**arial**; **color**:**green**;}

**p** {**font-weight**:**bold**;}

# CSS: Selektoren

**Elemente** (p, h1, ul, div, body, table...)

Stylesheet: **p,li** {**font**:**arial**; **color**:**green**;}

HTML <p>Dieser Text würde grün erscheinen</p>

**Klassen**

Stylesheet: **.fett** {**font-weight**:**bold**;}

HTML <p class="fett">Dieser Text würde fett erscheinen</p>

**IDs**

Stylesheet: **#box1** {**top**:**50px**;}

HTML <div id="box1">Dieser Text wäre positioniert (s.u.)</div>

**Pseudoformate** (Links: link|visited|hover..)

Stylesheet: **a:visited** {**text-decoration**:**none**;}

HTML <a href="index.html">Home</p>

# CSS: Beispiele von Angaben

## Schriftformatierung

font-family:"Times New Roman", Times, serif

font-style:italic

font-size:x-small / font-size:12pt

font-weight:bold

color:red / color:#FF0000

## Schriftausrichtung

text-align:left (right, center, justify)

line-height:20px



# CSS: Positionierung

## Elemente

- <div>: Erzwingt Zeilenumbruch; für Positionierung und Formatierung
- <span>: Text ist fortlaufend, zur Formatierung

## Beispiel

- HTML <div id="box1">Dieser Text wäre positioniert </div>
- Stylesheet: `#box1 {position:relative;top:50px;left:9px;width:150px;height:50px;background:red;}`

# CSS: Cascading Order

## Priorität

1. Inline style (inside an HTML element\*)
2. External and internal style sheets (in the head section)
3. Browser default

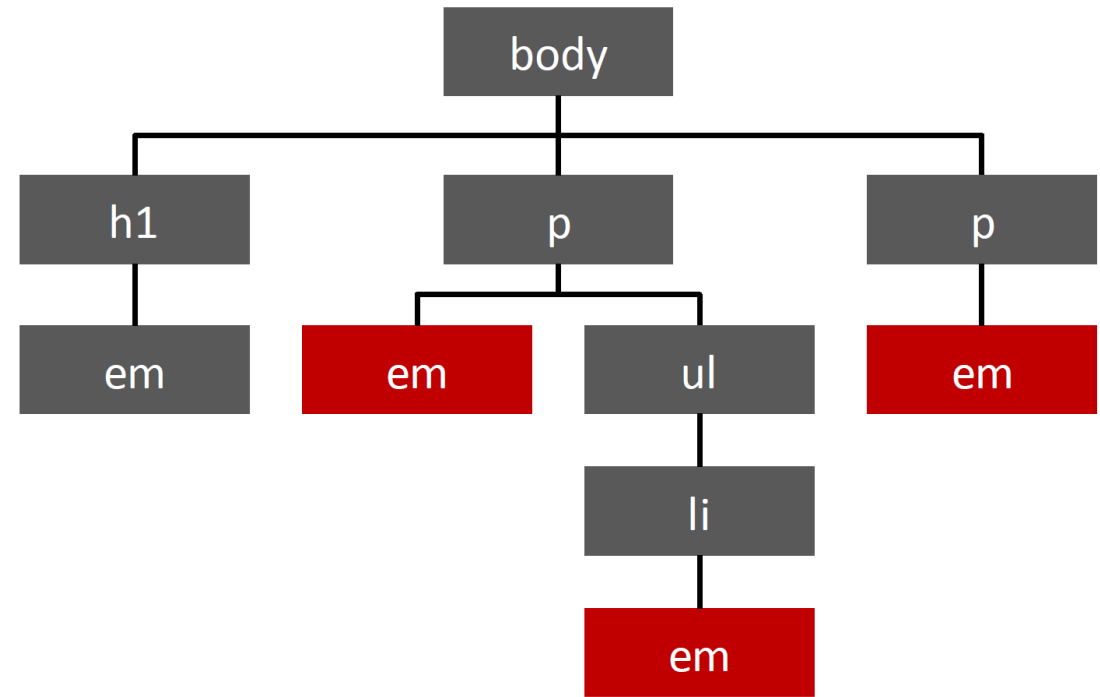
**\*)Inline Style (keine gute Praxis)**

`<h1 style="color:blue;margin-left:30px;">This is a heading</h1>`



# CSS: Selectoren

```
<body>  
  <h1>Überschrift <em>wichtig</em></h1>  
  <p>Selektoren <em>erster</em> Teil  
    <ul>  
      <li>Ein <em>Listenelement</em>  
    </ul>  
  </p>  
  <p>Unser <em>zweiter</em> Absatz</p>  
</body>
```

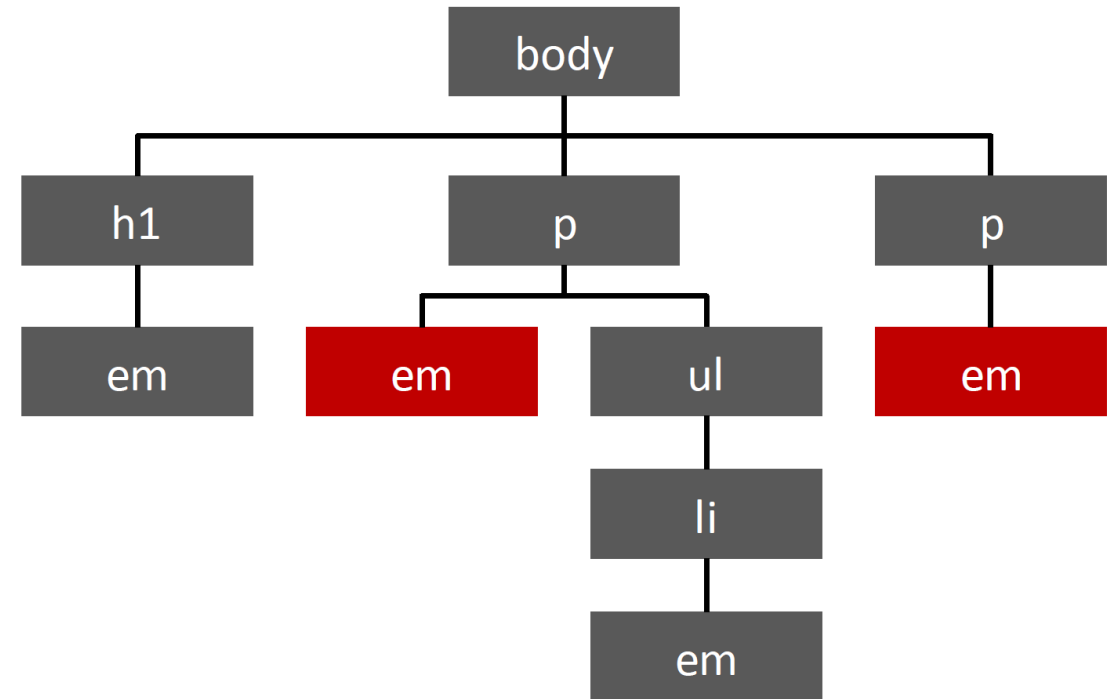


**p em {color : red;}**

(em-Element innerhalb eines p-Elements)

# CSS: Selectoren

```
<body>  
  <h1>Überschrift <em>wichtig</em></h1>  
  <p>Selektoren <em>erster</em> Teil  
    <ul>  
      <li>Ein <em>Listenelement</em>  
    </ul>  
  </p>  
  <p>Unser <em>zweiter</em> Absatz</p>  
</body>
```

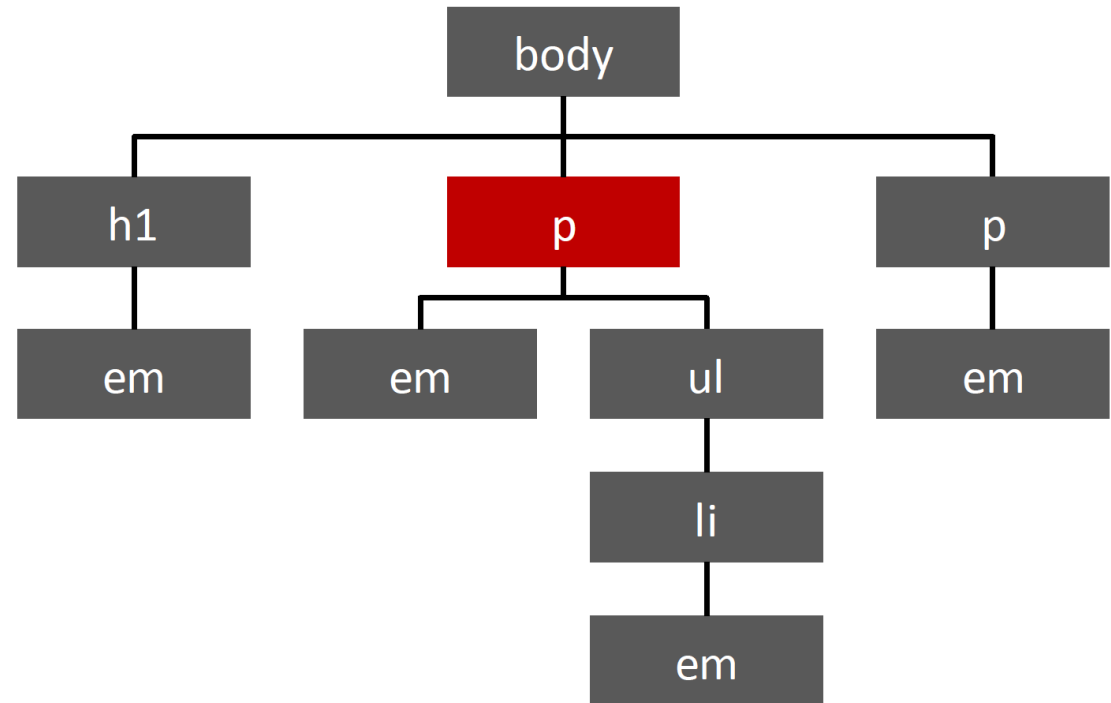


**p > em {color : red;}**

(em-Element direkt unterhalb p-Element)

# CSS: Selectoren

```
<body>  
  <h1>Überschrift <em>wichtig</em></h1>  
  <p>Selektoren <em>erster</em> Teil  
    <ul>  
      <li>Ein <em>Listenelement</em>  
    </ul>  
  </p>  
  <p>Unser <em>zweiter</em> Absatz</p>  
</body>
```

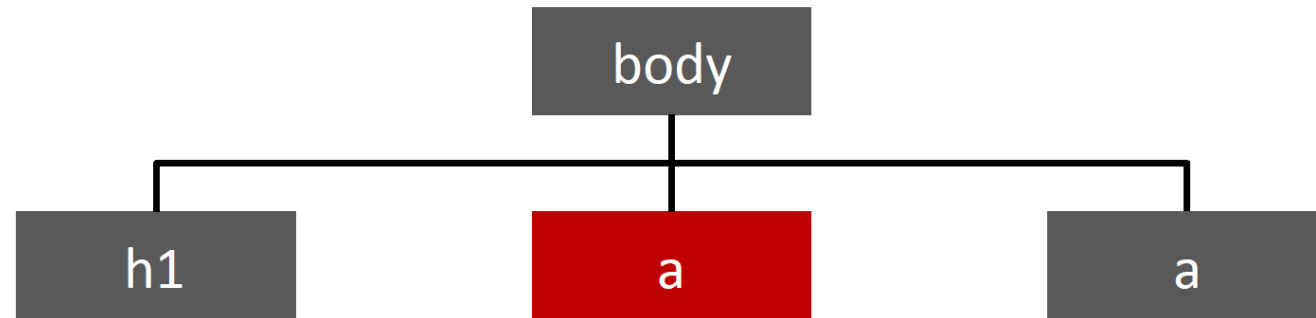


**h1 + p {color : red;}**

(p-Element in gleicher Ebene nach h1-Element)

# CSS: Selectoren

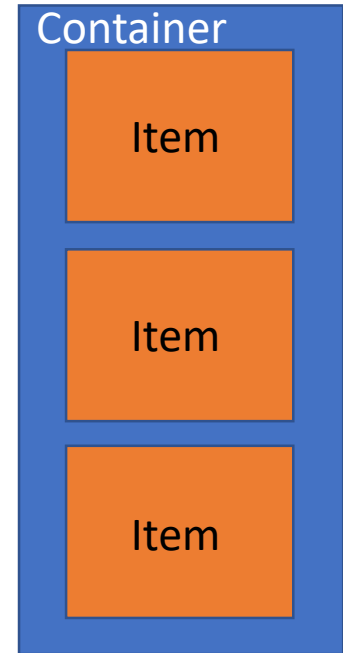
```
<body>  
  <h1>Überschrift <em>wichtig</em></h1>  
  <a href="www.ab.de">Link 1</a>  
  <a href="www.ce.de">Link 2</a>  
</body>
```



**a[href^="www.ab.de"] {color : red;}**

(a-Element dessen Attributwert mit "www.ab.de" beginnt )

# CSS: Fortgeschrittene (Beispiel Flexbox)



```
.container {  
    display: flex;  
    flex-direction: row | row-reverse | column | column-reverse;  
    flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

```
.item {  
    order: <integer>; /* default is 0 */  
    flex-grow: <number>; /* default 0 */  
    flex: flex-grow flex-shrink flex-basis | auto | initial | inherit
```

none entspricht: 0 0 auto

auto entspricht: 1 1 auto

initial entspricht: 0 1 auto (default)

inherit

[https://www.w3schools.com/cssref/css3\\_pr\\_flex-grow.asp](https://www.w3schools.com/cssref/css3_pr_flex-grow.asp)

[https://www.w3schools.com/cssref/css3\\_pr\\_flex-shrink.asp](https://www.w3schools.com/cssref/css3_pr_flex-shrink.asp)

[https://www.w3schools.com/cssref/css3\\_pr\\_flex-basis.asp](https://www.w3schools.com/cssref/css3_pr_flex-basis.asp)

[https://www.w3schools.com/css/css3\\_flexbox.asp](https://www.w3schools.com/css/css3_flexbox.asp)

# HTML: Limitierung

HTML ist eine nichterweiterbare Markup-Sprache für die Darstellung

HTML ist KEIN Austauschformat, wie beispielsweise XML oder JSON

# HTML

<https://validator.w3.org/>

<https://wiki.selfhtml.org/wiki/Startseite>

<https://www.w3schools.com/html/default.asp>

<https://www.w3schools.com/css/default.asp>

**CSS-Framework des W3C (frei verfügbar):** <https://www.w3schools.com/w3css/default.asp>

**Responsives Framework Bootstrap:** <https://www.w3schools.com/bootstrap/default.asp>