

# HTML & CSS

Prof. Dr.-Ing. Andreas Heil

 Licensed under a Creative Commons Attribution 4.0 International license. Icons by The Noun Project.

v2.0.0

# **Teil 1: Hypertext Markup Language**

# Auszeichnungssprachen (Markup Languages)

- Ursprung (engl.): *marking up documents*
- Auszeichnungen oder Formatierungen heben sich syntaktisch unterscheidbar vom Text ab
- Beispiele für Auszeichnungssprachen
  - LaTeX
  - XML
  - HTML
  - JSON?

# JSON

JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.<sup>1</sup>

▶ JSON ist also keine Auszeichnungssprache

# HTML

- **H**ypertext **M**arkup **L**anguage
- Auszeichnungselemente um den Inhalt eines Dokumentes zu beschreiben (Struktur, Formatierung)
- Markup directives to describe content (structure, formatting)
- Deklarative Sprache
- Annotationen mittels **HTML Tags** `< >`
- Groß-/Kleinschreibung ist irrelevant

## Beispiel

```
<p>Hallo Welt!</p>
```

# Anatomy von HTML Tags

Opening Tag      Closing Tag

`<p>Hallo Welt!</p>`

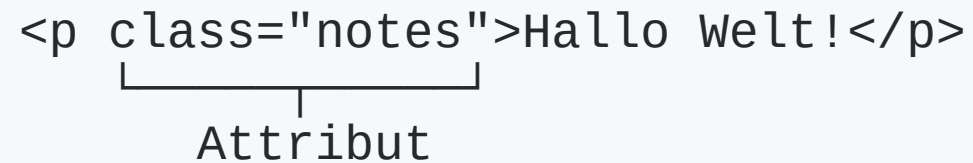
Inhalt

Element



`<p class="notes">Hallo Welt!</p>`

Attribut



# Sonderzeichen

Literal	HTML-Zeichenfolge
<	&lt;
>	&gt;
"	&quot;
'	&apos;
&	&amp;
nonbreaking space	&nbsp;

# HTML Dokument

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Eine HTML-Seite</title>
  </head>
  <body>
    <p>Hallo welt!!</p>
  </body>
</html>
```



# Header - Meta Tag

- Informationen über das Dokument
- Wird nicht angezeigt
- Maschinenlesebar

## Beispiel

```
<meta charset="utf-8">
```

# Meta Tag und Viewport

- *Viewport* ist der Bereich der Seite, die der Anwender sieht
- Abhängig vom Endgerät (Desktop, Mobilgerät)
- Kann über *meta*-Tag kontrolliert werden

## Beispiel

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- [https://www.w3schools.com/html/example\\_withoutviewport.htm](https://www.w3schools.com/html/example_withoutviewport.htm)
- [https://www.w3schools.com/html/example\\_withviewport.htm](https://www.w3schools.com/html/example_withviewport.htm)

# Header - Link Tag

- Beschreibt Relation zwischen der Seite und externen Quellen
- Oft, Verweis auf externe Ressource (Style Sheets)
- Besonderheit: Das HTML-Element ist leer und enthält nur Attribute

## Beispiel

```
<link rel="stylesheet" href="styles.css">
```

# XHTML

- Extensible Hypertext Markup Language
- Erweiterung aus XML and HTML
- Wesentlich restriktiver als vanilla HTML
- Warum?
  - Fehlerhaftes HTML (z.B. fehlende schließende Tags)
  - Fehlende, inkonsistente Anführungszeichen (z.B. bei Attributen)
  - Fehlende Tags (z.B. `<head>` , `<title>` oder `<body>` )
  - Probleme bei der Interpretation durch den Browser

# XHTML Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Eine XHTML Seite</title>
  </head>
  <body>
    <p>Hallo Welt!</p>
    <br />
  </body>
</html>
```

# XHTML - Grundlegende Regeln

- Jedes Tag muss geschlossen werden
- Kurzform für `<p></p>` ist `<p />`
- Anführungszeichen sind zwingend erforderlich
- `<html>`, `<head>`, `<title>`, und `<body>` sind verpflichtend
- `<!DOCTYPE>` declaration is necessary at the top of the file
- HTML-Tags und Attribute in Kleinbuchstaben
- Keine »attribute minimization«

# Attribute Minimization

In HTML möglich

```
<input name="angemeldet" type="checkbox" checked />
```

In XHTML erforderlich

```
<input name="name" type="checkbox" checked="checked" />
```

# DOCTYPE

- Die `<!DOCTYPE>`-Anweisung ist **kein** HTML-Tag
- Hinweis für den Browser, was er im Dokument zu erwarten hat
- Bei HTML 4 bzw. XHTML muss ein Doctype immer auf ein DTD (Document Type Definition) verweisen
- In HTML 5 einfacher, hier genügt  
`<!DOCTYPE html>`
- Unterschiedliche Doctypes erlauben unterschiedliche HTML-Tags



# HTML Übersicht

w3schools<sup>2</sup>

- Als Übersicht
- Als Referenz
- <https://www.w3schools.com/>

## Teil 2: Cascading Style Sheets

# Anti-Style

```
<p>  
  <font face="Arial">Willkommen an der Hochschule Heilbronn</font>  
  Wie bilden <b>die</b>, <i>besten</i>, und <u>tollsten</u>  
  <font size="+4" color="red">Software-Entwickler</font> aus!  
</p>
```

Wie bilden **die** *besten* und TOLLSTEN **Software-Entwickler** aus!

Aus vielen Gründen keine gute Idee

- Accessibility
- Trennung von »Code« und Darstellung
- Und ja, das hat man »früher« so gemacht

# Problem hinter CSS

Wie rendert der Browser eigentlich die vorherigen Tags, z.B. wo kein Font angegeben wurde?

- Browser nutzt »irgendeinen« Default
- Das HTML enthält das *was*, der Browser kümmert sich um das *wie*
- Warum? HTML ist eine deklarative Sprache (s. vorher)

Früher:

- Überschreiben von Default-Werten mit Attributen

```
<table border="3" bordercolor="black">  
...  
</table>
```

# Lösung

CSS adressiert exakt diese Probleme

- Verwendung einen spezifischen Styles anstelle der Defaults in (verschiedenen) Browser
- Keine Attribute zur Darstellung an allen möglichen HTML-Tags

# Konzepte hinter CSS

- Inhalt ist in der HTML-Datei
- Informationen zur Formatierung in separaten Dateien (Style Sheets/.css-Dateien)
- Anwendung durch `class`-Attribute (z.B. `<p class="em">` )
- Wiederverwendung möglich: CSS-Klassen einmal definieren und an andere Stelle wiederverwenden #
- Eine zentrale Änderung am Style Sheet ermöglicht alles mit einer Änderung anzupassen
  - Farbe, Schriftart, Größe etc.
  - Schnelle Anpassung an Kundenwünsche, Customizing, Corporate Design etc. (z.B: Wordpress Themes)

# CSS Rules

```
selektorliste {  
  eigenschaft: wert;  
  [weitere eigenschaft:wert; Paare]  
}
```

## Beispiel

```
strong {  
  color: red;  
}
```

# Selektoren

## Via Tag-Name

HTML:

```
<h1>Willkommen an der Hochschule Heilbronn!</h1>
```

CSS:

```
h1 {  
  color: blue  
}
```



# Selektoren (Forts.)

## Via Klassen-Attribut

HTML:

```
<p class="large">...</p>
```

CSS:

```
.large {  
  font-size: 18pt;  
}
```

# Selektoren (Forts.)

## Via Tag und Klasse

HTML:

```
<p class="large">...</p>
```

CSS:

```
p.large {  
  font-size: 32pt;  
}
```

# Selektoren (Forts.)

## Via der ID eines HTML-Elements

HTML:

```
<p id="titel">...<p>
```

CSS:

```
#titel {  
  font-weight: bold;  
}
```

# References