RWTH AACHEN UNIVERSITY

JÜLICH
FORSCHUNGSZENTRUM

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Web-Engineering und Internettechnologien

Prof. Dr. Volker Sander
Prof. U. Stegelmann
Thomas Dondorf, M.Sc. (dondorf@lfi.rwth-aachen.de)

v2.2.1, 05.04.2018

Themen der Vorlesung

Einführung
- Motivation, HTML, CSS, HTTP

PHP
- Syntax, Funktionen, OOP

JavaScript
- Syntax, jQuery, AJAX, OOP, Node.js

Server
- Apache, nginx

Java
- Servlets, Java Server Pages (JSP)

Sicherheit
- Protokolle, OWASP

Begleitet durch
- Praktika (Übungen) und Projekte
- ILIAS-Kurs: Web-Engineering und Internettechnologien

Literatur

Prüfungsvorbereitung

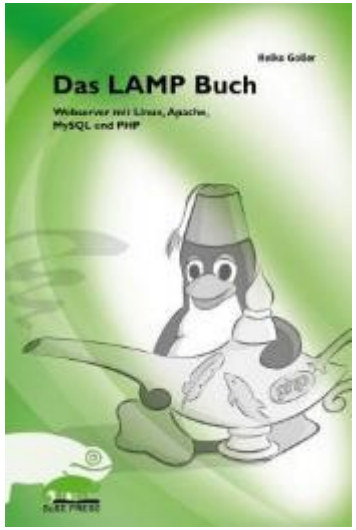
- Folien (inkl. Verweisen), Übunge, Praktika und Projekt ausreichend

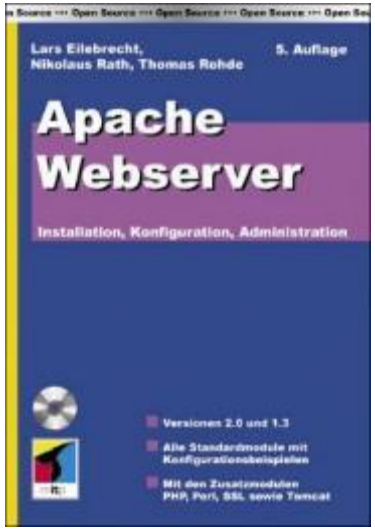
Darüber hinaus

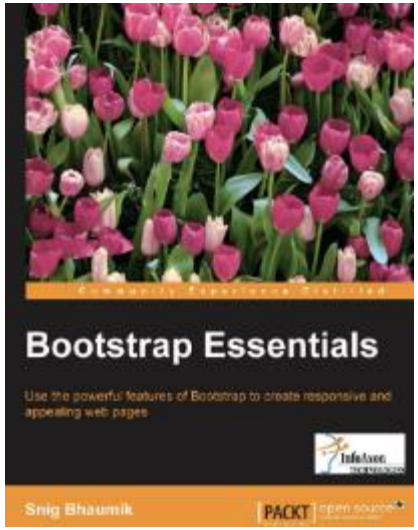
- Empfehlenswert sich mit Literatur zu beschäftigen
- Zeitschriftenartikel lesen
- „Auf dem Stand bleiben“

Literatur


Jörg Barres
Webtechnologien
All in One
Eine praxisorientierte Einführung in moderne Webtechnologien
Javascript SQL jQuery HTML PHP CSS
traicon.net

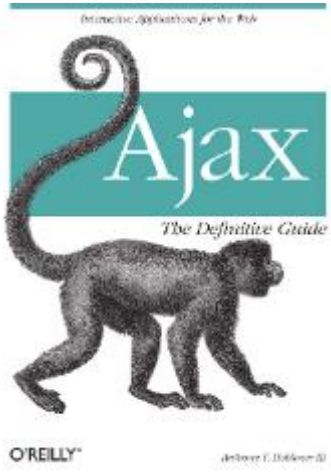

Heiko Gollor
Das LAMP Buch
Webserver mit Linux, Apache, MySQL und PHP

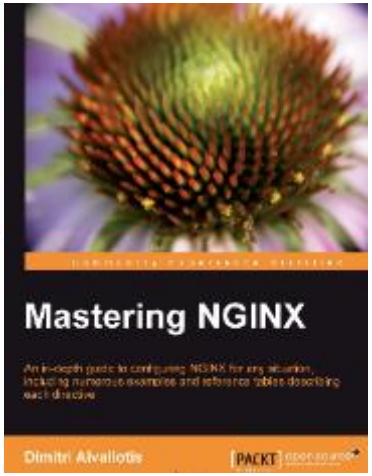

Lars Eilebrecht, Nikolaus Rath, Thomas Rohde
5. Auflage
Apache Webserver
Installation, Konfiguration, Administration
Versionen 2.0 und 1.3
Alle Standardmodule mit Konfigurationsbeispielen
Mit den Zusatzmodulen PHP, Perl, SSL sowie Tomcat


Bootstrap Essentials
Use the powerful features of Bootstrap to create responsive and appealing web pages
Snig Bhaumik
PACKT open source


Chuck Hudson
Tom Leadbetter
HTML5
Developer's Cookbook
Developer's Library


O'REILLY
3rd Edition
PHP Cookbook
David Sklar & Adam Trachtenberg


Ajax
The Definitive Guide
O'REILLY


Mastering NGINX
Dimitri Aivaliotis
PACKT open source


O'REILLY
Security for Web Developers
John Paul Mueller

Literatur

Josh Lockhart: *PHP: The "Right" Way*

- PHP best practices, kostenlos online lesen: www.phptherightway.com

C. Kunz, S. Esser: *PHP-Sicherheit*, 2008

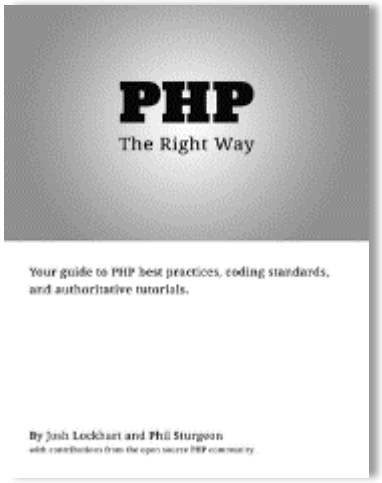
- Sicherheit in Webanwendungen

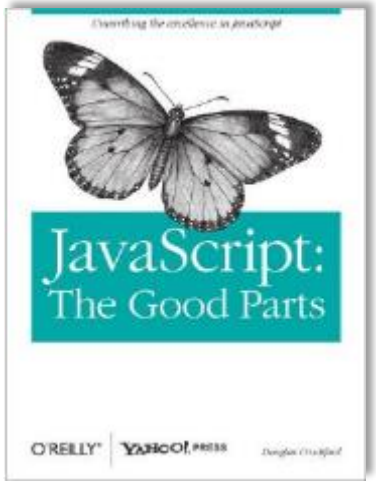
Douglas Crockford: *JavaScript: The Good Parts*, 2008

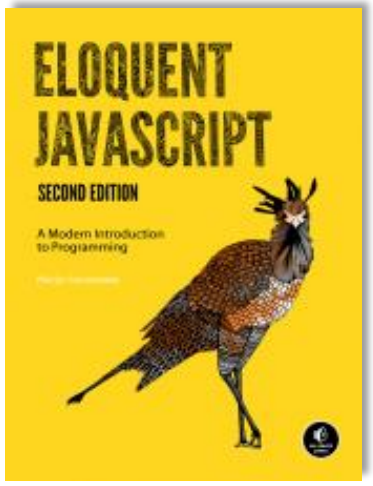
- Pflichtlektüre für jeden JavaScript-Entwickler

Marijn Haverbeke: *Eloquent JavaScript*, 2011

- „Second Edition" (2014) online lesen: eloquentjavascript.net



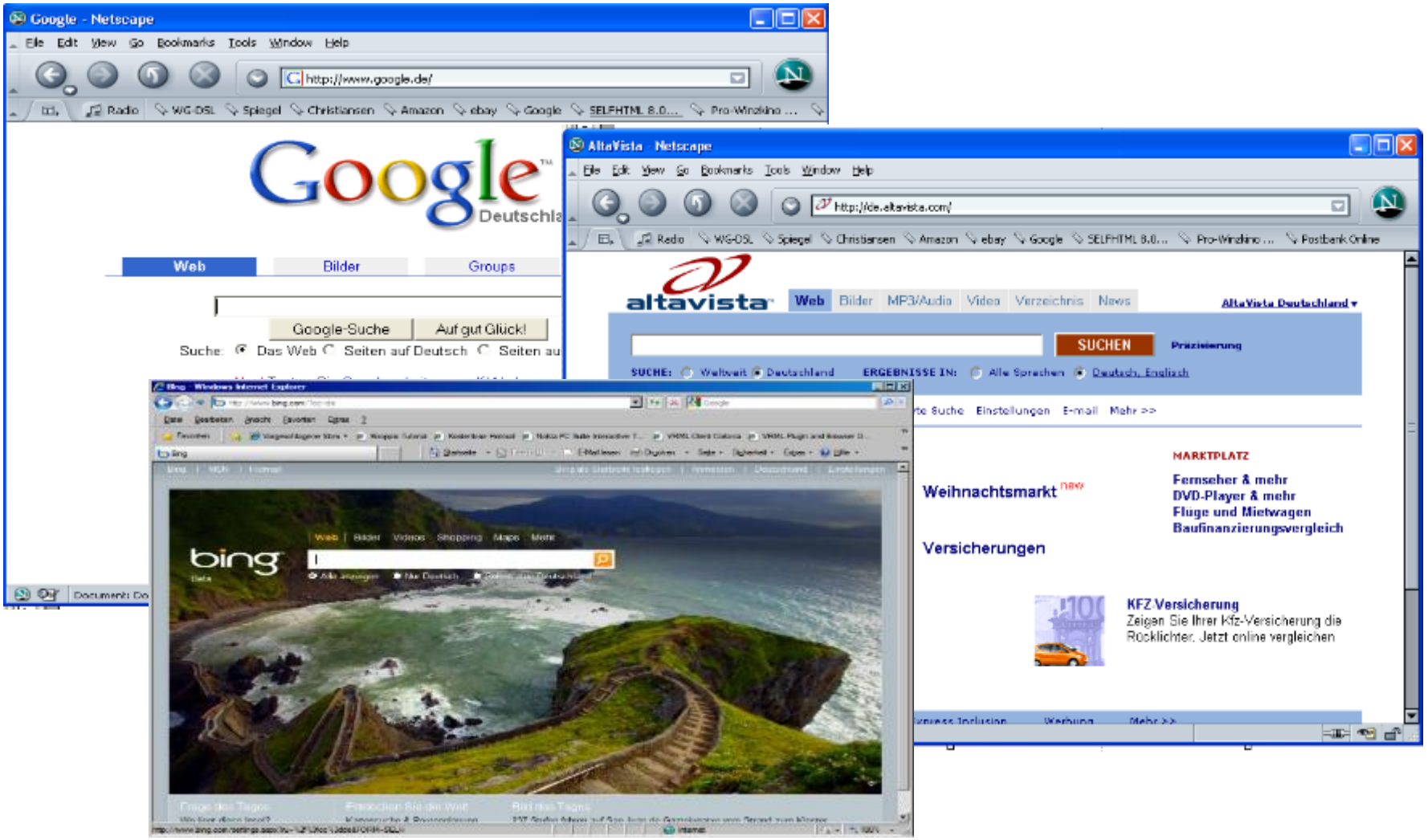




Einführung in Internettechnologien

Motivation, HTML, CSS, HTTP

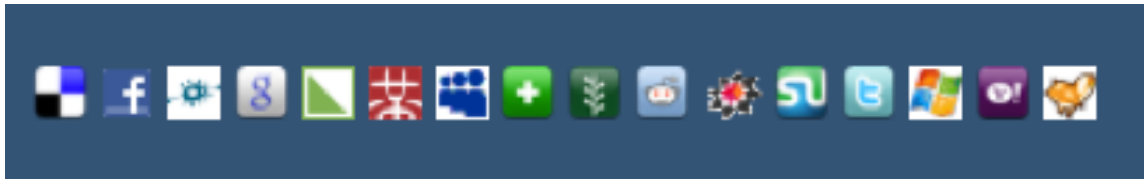
Motivation


Google - Netscape
File Edit View Go Bookmarks Tools Window Help
http://www.google.de/
Google
Web
Bilder
Groups
Google-Suche
Auf gut Glück!
Suche: Das Web Seiten auf Deutsch
AltaVista - Netscape
http://de.altavista.com/
altavista
Web Bilder MP3/Audio Video Verzeichnis News
SUCHEN
Präzisierung
Einstellungen E-mail Mehr >>
MARKTPLATZ
Weihnachtsmarkt
Fernseher & mehr
DVD-Player & mehr
Flüge und Mietwagen
Baufinanzierungsvergleich
Versicherungen
KFZ Versicherung
Zeigen Sie Ihrer Kfz-Versicherung die Rücklichter. Jetzt online vergleichen
bing

Motivation

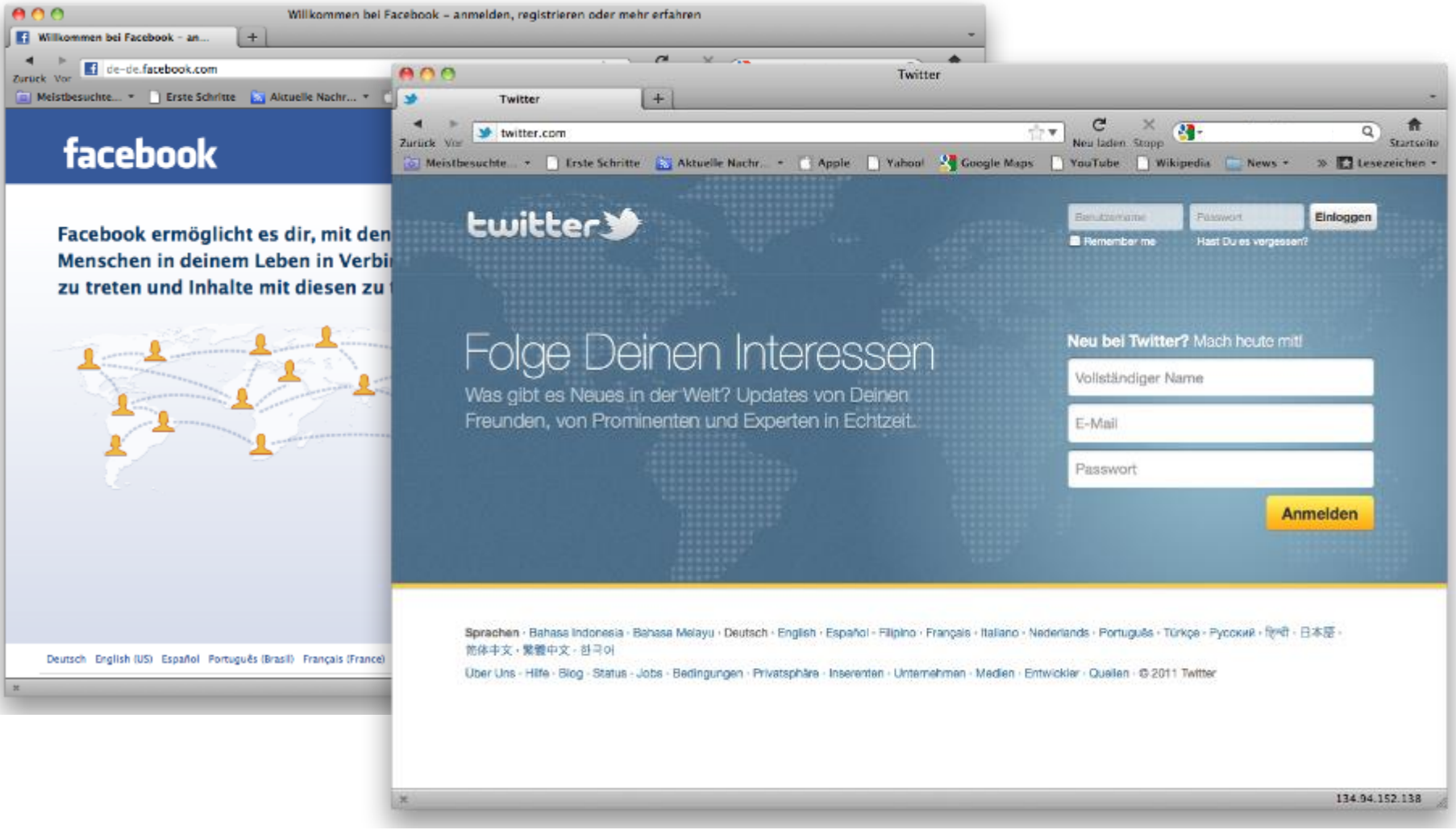
Vom passiven „Informations"-Nutzer zum aktiven Nutzer

- Der „Blogger", der „Twitterer"
- OFFICE online (MS) oder Google Drive oder Apple iCloud…

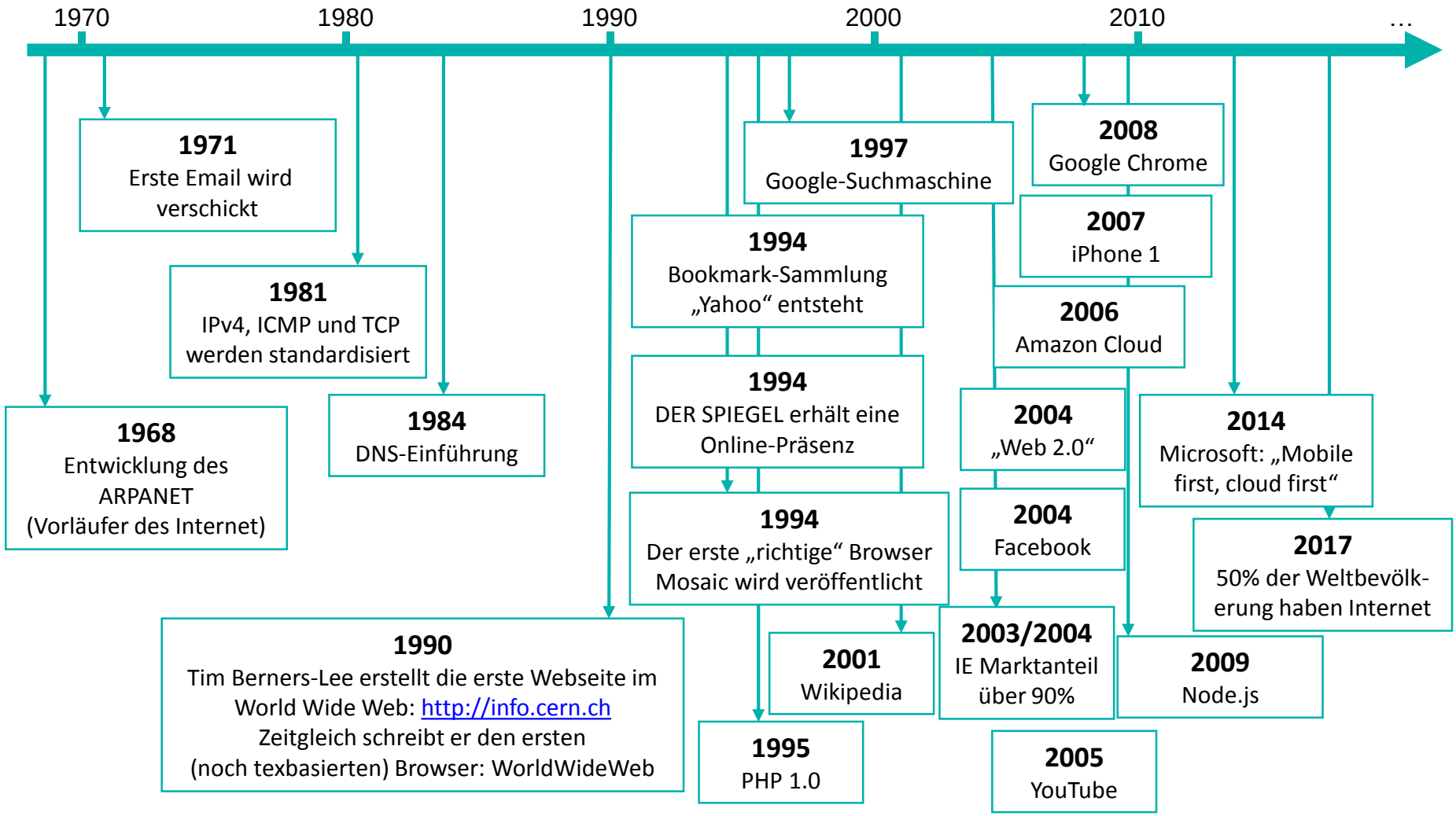


- Foren
- Empfangen von RSS-Feeds
- Podcasts, YouTube, iTunes U
- „eLearning Systeme", „Campus"

Motivation


Willkommen bei Facebook – anmelden, registrieren oder mehr erfahren
facebook
Facebook ermöglicht es dir, mit den
Menschen in deinem Leben in Verbi
zu treten und Inhalte mit diesen zu
Deutsch English (US) Español Português (Brasil) Français (France)
Twitter
twitter.com
twitter
Einloggen
Remember me
Hast Du es vergessen?
Folge Deinen Interessen
Was gibt es Neues in der Welt? Updates von Deinen Freunden, von Prominenten und Experten in Echtzeit.
Neu bei Twitter? Mach heute mit!
Vollständiger Name
E-Mail
Passwort
Anmelden
Sprachen · Bahasa Indonesia · Bahasa Melayu · Deutsch · English · Español · Filipino · Français · Italiano · Nederlands · Português · Türkçe · Русский · हिन्दी · 日本語 · 简体中文 · 繁體中文 · 한국어
Über Uns · Hilfe · Blog · Status · Jobs · Bedingungen · Privatsphäre · Inserenten · Unternehmen · Medien · Entwickler · Quellen · © 2011 Twitter
134.94.152.138

Die Evolution des Internets



HTML – HyperText Markup Language

HyperText Markup Language

HTML ist die Beschreibungssprache des WWW

- Hypertext: Dokumente, die Verweise auf andere Dokumente enthalten
- universelle, plattformunabhängige Markup-Sprache
- Markup steht dabei für Auszeichnung und bedeutet, dass Teile eines Textes als besondere strukturelle Elemente ausgezeichnet werden können. Beispiele: Überschriften, Tabellenstrukturen, …

Eigenschaften von HTML

- HTML dient der Definition des Aufbaus einer Internetseite: Elemente, Strukturen, Verweise (Hyperlinks), referenzierte Elemente (Grafiken, Multimedia...)
- Seit 1992 standardisiert und zunehmend fortentwickelt
- Im Jahr 2014 wurde HTML 5 standardisiert

HyperText Markup Language

HTML, und was gibt es noch?

- HTML: Inhalt und Aufbau/Layout der Seite
- CSS: Design/Style des Dokumentes
- JavaScript: Logik, Programmierung innerhalb des Dokumentes

Vergleiche mit anderen Sprachen

	Web	LaTeX	Java	Android	JavaFX	C++
Inhalt	HTML	TEX	Java	Java/XML	Java/FXML	C++
Design	CSS	CLS/STY	Java	XML	CSS	C++
Logik/Programmierung	JavaScript	TEX	Java	Java	Java	C++

HTML
Funktionsweise

Hierarchische Gliederung der Auszeichnungselemente, z.B.

- Auszeichnungselemente haben einen festen Erstreckungsraum
- Elemente können verschachtelt sein
- Verschachtelte Elemente müssen vollständig in den Erstreckungsraum der übergeordneten Elemente enthalten sein
- Darstellung als Baum möglich (DOM-Baum)

[Überschrift] *Text der Überschrift* **[Ende Überschrift]**
[Liste]
  [Listenpunkt] *Text des Listenpunkts* **[Ende Listenpunkt]**
  [Listenpunkt] *Text des Listenpunkts* **[Ende Listenpunkt]**
[Ende Liste]

HTML
Auszeichnungselemente: Tags

HTML-Elemente werden durch sogenannte „Tags“ markiert

- Tags werden in spitzen Klammern (<, >) notiert
- Die meisten HTML-Elemente werden durch ein einleitendes (<>) und ein abschließendes (</>) Tag markiert, außer der Erstreckungsraum ist auch ohne schließendem Tag klar
- Falls der Tag keinen schließenden Tag benötigt, so wird häufig der Tag selbst sofort geschlossen (Beispiel: <br /> oder <img />)
- Einige Tags unterstützen Attribute (Notation: name="wert"), die die Semantik präzisieren

```
<h1>Überschrift 1. Ordnung</h1>
<strong>Wichtiger Text in Fettschrift</strong>
Ein Zeilenumbruch wird durch den Tag <br>
eingeleitet

<h1>HTML - die Sprache des Web</h1>
<img src="katze.jpg" alt="Katze" width="100" height="100">
```

Vereinfachte Darstellung eines HTML-Dokumentes

```
<!DOCTYPE ...>
<html>

  <head>
     Grundsätzliches zum Dokument
     (z.B. <title>Fenstertitel</title>)
  </head>

  <body>

  Eigentliches Dokument mit Auszeichnungs-
  elementen

  </body>

</html>
```

HTML
Grundlegender Aufbau

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Titel der Webseite</title>
</head>
<body>

<h1>Überschrift</h1>
Text mit Zeilenumbruch am Ende<br>
An dieser Stelle erfolgt kein Umbruch:
Sag ich doch!

</body>
</html>
```

HTML
Darstellung als DOM-Baum

Darstellung des HTML-Dokumentes als DOM-Baum

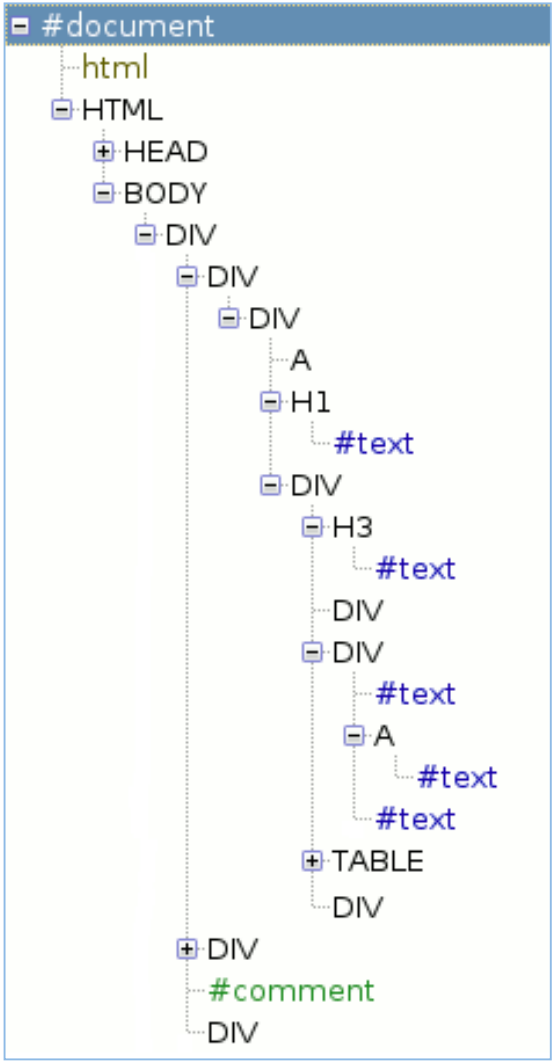
- Einfachere Darstellungsform
- Erlaubt Zugriff auf alle Tags und Attribute der Webseite
- Baumstruktur
- Tags und Texte werden durch Knoten repäsentiert

Beispiel (rechte Seite)

- Teilweise ausgeklappter DOM-Baum
- Text ist ein eingener DOM-Knoten (#text)

Tag-Attribute

- Zusätzlich zu den Kindknoten kann jeder Knoten Eigenschaften durch Attribute erhalten



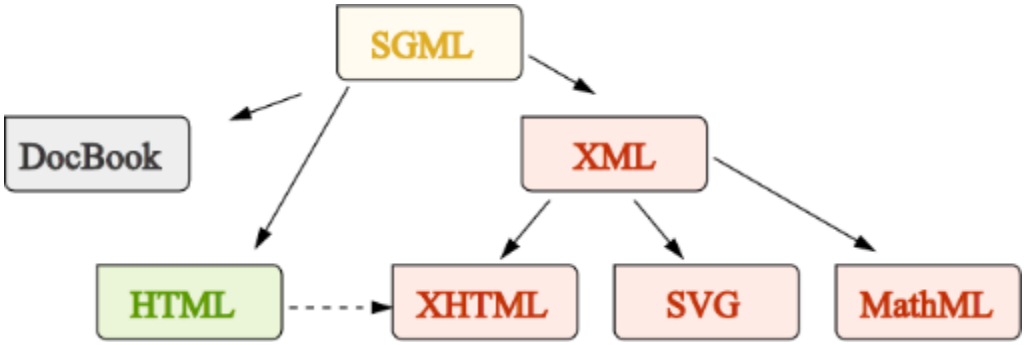
HTML
Historie HTML4/XHTML/HTML5

XML (Extensible Markup Language) und HTML

- HTML ursprünglich unabhängig von XML entwickelt
- Beide enstanden aus Standard Generalized Markup Language (SGML)
  - > Ähnlicher Aufbau (Tags, Baumstruktur)
- XML erlaubt beliebige Tags, HTML hat vorgegebene Tags

XHTML

- Vereinigung von HTML und XML
- Version 2.0 zu Gunsten von HTML 5 eingestellt



Quelle: https://wiki.selfhtml.org/wiki/Datei:Auszeichnungssprachen.svg

ISO 8879 (SGML): https://www.iso.org/standard/16387.html

HTML
Historie HTML4/XHTML/HTML5

HTML 4 (1997), HTML 4.01 (1999)

- Teils unspezifische Vorgaben
- Einfach HTML zu schreiben
- Schwierig browserübergreifend das gleiche Layout zu erzeugen

XHTML: Extensible HTML (~2000)

- Strikter Standard sollte die Fehler von HTML 4 beheben
- Durchsetzung von XHTML scheitert auf Grund der Striktheit
  - > Minimale Fehler im XML können zum kompletten Ausfall der Seite führen, der Browser zeigt einen XML-Parse-Error

HTML 5 (Entwicklung seit ca. 2008, Recommendation 2014)

- Ein browserübergreifender einheitlicher Parser soll die Unterschiede im Anzeigen verhindern und gleichzeitig Flexibilität erlauben
- Zahlreiche moderne HTML-Erweiterungen (video, audio)
- Viele „Altlasten" entfernt
- Oft als Sammelbegriff für moderne Technologien genutzt

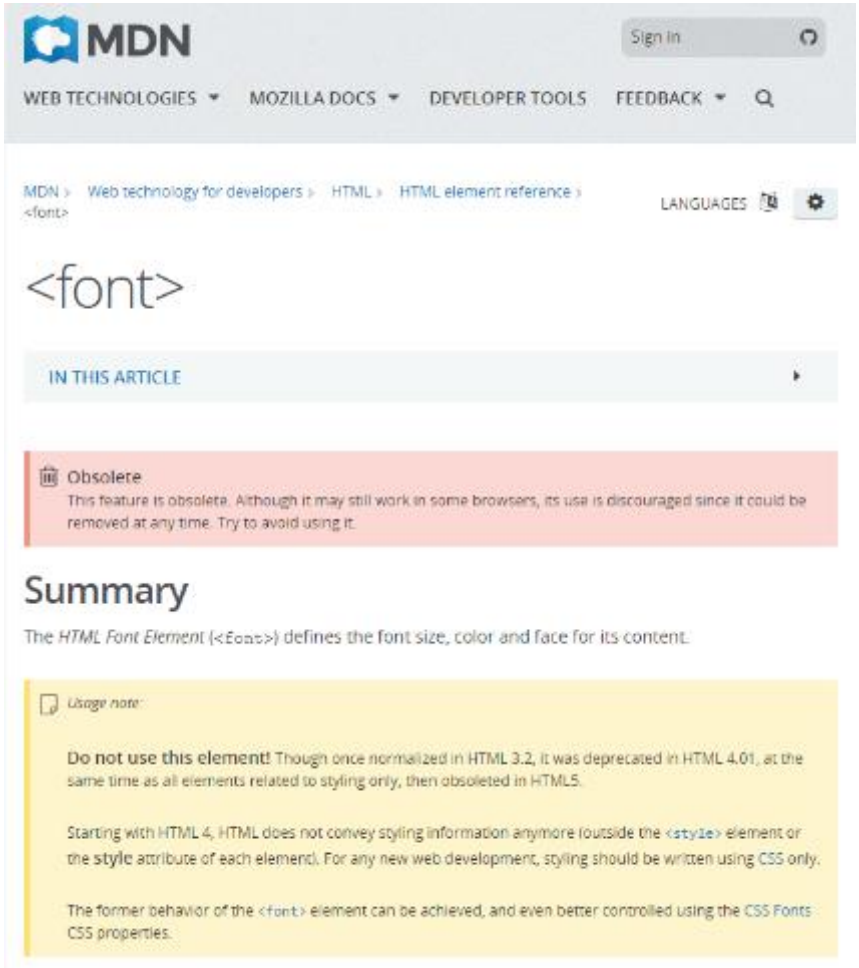
Unsere Beispiele sind HTML 5

Weitere Informationen: https://html.spec.whatwg.org/#history-2

HTML
Historie HTML4/XHTML/HTML5

HTML Altlasten

- Styling im HTML-Code
  - > Veränderung von Farben, Textgrößen, Abständen, usw. im HTML (z.B. mittels font-Tags)
  - > Zweckentfremdung von bold, italics, underline
  - > Sollte seit HTML 4 (1997) mittels CSS gemacht werden!
- Escaping von Umlauten (und anderen Zeichen)
  - > Bei der Wahl des richtigen Zeichenstatzes ist das nicht nötig
- JavaScript im HTML-Code
  - > HTML, JavaScript und CSS sollten getrennt sein



Quelle: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/font

Beispiele für schlechten Code

```
<body bgcolor="red">
<font size="+2" onclick="alert('Hi');">Gro&szlig;er Text</font>
<b>Das hier ist wichtig!</b>

<table align="center">
    <tr>
        <td height="50" width="50">Text</td>
        <td height="50" width="100">Mehr Text</td>
    <tr>
</table>
...
```

Browserunterstützung

- Trotz Auszeichung als „obsolete" unterstützen moderne Browser aus Gründen der Kompatibilität i.d.R. immer noch alle veralteten Tags und Attribute

Viele Internetseiten enthalten immer noch schlechte Codebeispiele

- Im Zweifel MSDN oder MDN aufsuchen

Weitere Informationen: https://www.w3.org/TR/html5/obsolete.html

Zeichensatz des Dokumentes

- Bestimmt welche Zeichen für die übermittelten Bytes angezeigt werden

- Beispiele:
  - > ASCII-Byte: 65 (alternativ: 0x41 oder 01000001)
    - Entspricht: A
  - > UTF-8 Byte: 65 (alternativ: 0x41 oder 01000001)
    - Entspricht: A

- Die ersten 128 Zeichen (0-127) des ASCII und UTF-8-Standards sind gleich!
  - > Tatsächlich besteht ein ASCII-Code nur aus 7 Bits

UTF-8 ist abwärtskompatibel zu ASCII

UTF-8 ist abwärtskompatibel zu ASCII

- Problem: Es fällt evtl. nicht auf, wenn ein Dokument mit einem falschen Zeichensatz ausgeliefert wurde.

Mögliche Lösung: Andere Zeichen kodieren

- Beispiele:
  - > € statt €
  - > ä statt ä
  - > © statt ©

- Unsaubere Lösung!
  - > Vor allem historisch bedingt, da Browser sich in manchen Fällen über den übermittelten Zeichensatz hinwegsetzten
  - > Besser: Korrekten Zeichensatz angeben!

When not to use escapes: https://www.w3.org/International/questions/qa-escapes#not

HTML
Zeichensatz

Nur wenige Zeichen müssen tatsächlich „escaped“ werden:

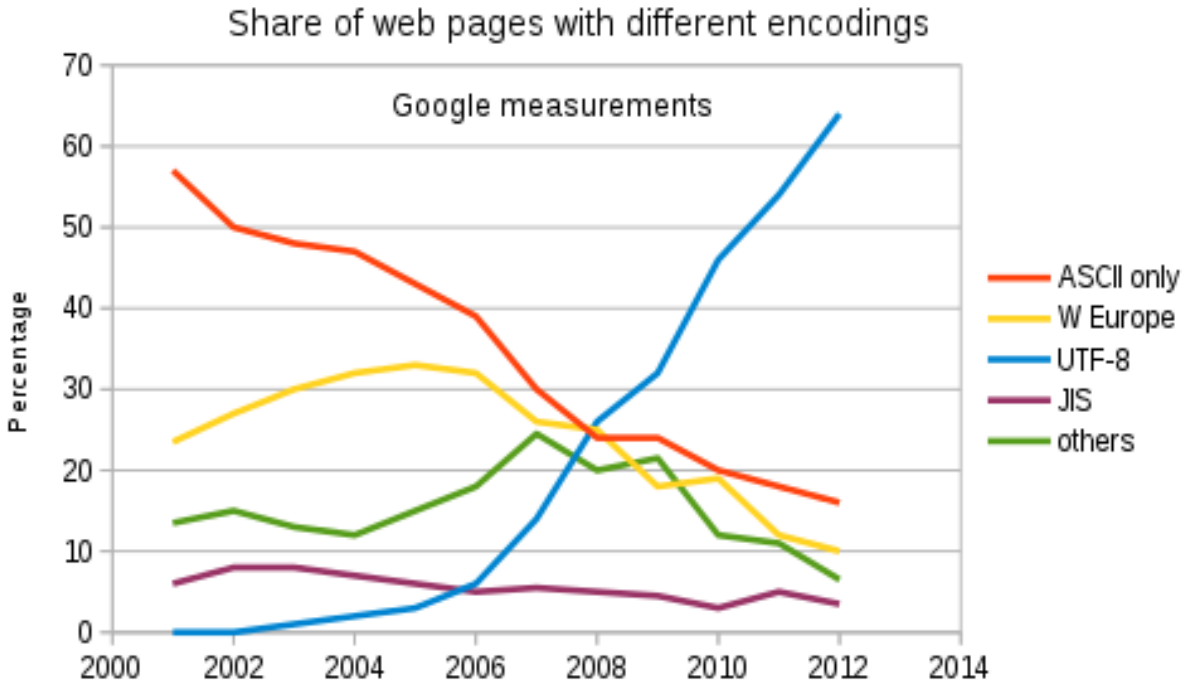
- Tags um eckige Klammern zu nutzen ohne, dass sie als Tag interpretiert werden
  - > Beispiel: `Ein h1-Tag: <h1>`
- „Ampersand“, das sonst als Einleitung eines Escape-Characters interpretiert wird
- Anführungszeichen innerhalb von Attributen
  - > Beispiel: `title="Er sagte &quot;Hallo&quot;."`

Zeichen	Bedeutung	HTML-Code
<	Öffnende Klammer (Auswertung als Tag verhindern)	`<`
>	Schließende Klammer	`>`
&	„Ampersand“, Käufmännisches Und	`&`
"	Doppelte Anführungszeichen	`&quot;`
'	Einfaches Anführungszeichen	`&#39;`

When to use escapes: https://www.w3.org/International/questions/qa-escapes#use

Den korrekten Zeichensatz wählen

- Immer UTF-8 wählen!
  - > Weitverbreitetste Implementierung von UNICODE-Zeichen



Quelle: https://commons.wikimedia.org/wiki/File:Utf8webgrowth.svg

Aktuellere Werte: https://w3techs.com/technologies/overview/character_encoding/all/

UTF-8 nutzen

1. Editor auf UTF-8 stellen
   > Code im richtigen Zeichensatz speichern

2. Encoding im HTML angeben

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

3. Sicherstellen, dass der Server das gleiche Encoding übermittelt
   > Response Header: `Content-Type: text/html;charset=utf-8`
   > Einstellung im Apache:
   – httpd.conf / .htaccess: `AddDefaultCharset UTF-8`
   > Bei wiedersprüchlichen Angaben nutzt der Browser das Server-Encoding!

Weitere Informationen: https://www.w3.org/International/questions/qa-html-encoding-declarations

HTML
<head></head>

Enthält Informationen über das Dokument

- Beispiel

```
<head>
    <meta charset="utf-8">
    <title>Titel der Webseite</title>
    <meta name="author" content="Autor der Seite">
    <meta name="description" content="Beschreibung der Webseite,
        die u.a. von Google verwendet wird">
    <meta name="keywords" content="Schüsselworte,HTML,lernen">
    <link rel="stylesheet" href="style.css" type="text/css">
</head>
```

- Die Informationen im <head> sind größtenteils für nicht menschliche Besucher der Webseite von Bedeutung (Browser, Bots)
- Teilweise wird dadurch die Nutzung eines Tags die Darstellung des body-Tags verändert (z.B. durch Einbindung eines Stylesheets)
- Manche Tags können sowohl im <head> als auch im <body> vorkommen (z.B. <script>)

HTML
<body></body>

Enthält den eigentlichen Teil des Dokuments

- Beispiel

```
<body>
<h1>Überschrift</h1>

<h2>Tabelle</h2>
<table>
    <tr>
        <th>Land</th>
        <th>BIP</th>
    </tr>
    <tr>
        <td>Deutschland</td>
        <td>3,73 Billionen USD</td>
    </tr>
    <tr>
        <td>UK</td>
        <td>2,81 Billionen USD</td>
    </tr>
</table>
</body>
```

HTML Elemente

Metadaten

- <meta>, <title>, <base>, <link>, <script>, <style>

Flow/Fluss-Elemente

- <a>, <article>, <br>, <button>, <form>, , <header>, <img>, <input>, <nav>, <ol>, <p>, <script>, <h[1-6]>, <section>,

Elemente zur Aufteilung

- <article>, <aside>, <nav>, <section>

Überschriften

- <h1>, <h2>, …

Stil-Elemente

- <b>, <button>, <code>, <input>, <label>, , <time>

Elemente zur Einbettung von Inhalten

- <audio>, <canvas>, <iframe>, <img>, <object>, <svg>, <video>

Interaktive Elemente

- <a>, <button>, <iframe>, <label>, <select>, <textarea>

Vollständige Liste: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Content_categories

HTML Elemente
Logisch vs. Physisch

Aussage von Elementen

- In HTML gibt es logische und physische Elemente zur Auszeichnung von Text.
- Physische Elemente definieren das Aussehen eines Elementes (z.B. unterstrichen, kursiv), treffen aber keine Aussage über die Bedeutung.
- Logische Elemente machen eine Aussage über die Bedeutung des Elementes (z.B. um was für Text es sich handelt), die Formatierung hängt vom Browser ab, kann aber auch mit Hilfe von CSS verändert werden.

Einsatz

- In den meisten Fällen ist es sinnvoll die logischen Elemente zu nutzen (wenn möglich). Mit CSS kann das Aussehen beliebig verändert werden.
- Physische Elemente sollten mit Bedacht eingesetzt werden.

HTML Elemente
Logisch vs. Physisch

Beispiele für logische Auszeichnungen

```
<strong>stark betont</strong>
<em>betont (emphasized)</em>
<code>Quellcode</code>
<samp>Beispiel</samp>
<cite>Zitat</cite>
<dfn>Definition</dfn>
<abbr>abgekürzte Schreibweise</abbr>
```

Beispiele für physische Auszeichnungen

```
<b>fett</b>
<i>kursiv (italic)</i>
<u>unterstrichen (underlined)</u>
<strike>durchgestrichen</strike>
<big>größer als normal</big>
<small>kleiner als normal</small>
<sup>hochgestellt</sup>
<sub>tiefgestellt</sub>
```

Weitere Informationen: https://developer.mozilla.org/de/docs/Web/HTML/Element/strong#Bold_vs._Strong

HTML Elemente
Überschriften

HTML unterscheidet 6 Überschriftenebenen

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

Nicht zu verwechseln mit: <header>…</header>

- Keine Überschrift!
- Element mit Kopf-/Einführungsinformationen darin
  - > Dies können z.B. Logo der Seite und Titel sein

- <footer>…</footer> entspricht dem Gegenstück dazu

Weitere Informationen: https://developer.mozilla.org/docs/Web/HTML/Element/h1-h6, https://developer.mozilla.org/docs/Web/HTML/Element/header

HTML Elemente
Textabsätze und Zeilenumbrüche

Textabsatz

- Repräsentiert einen Absatz

```
<p>Erster Absatz.</p>
<p>Hier beginnt der zweite Absatz.</p>
<p>Und hier ist noch ein Absatz.</p>
```


- Erzwingt einen Zeilenumbruch
- Dies sollte nicht als Ersatz für gesehen werden.

Weitere Informationen: https://developer.mozilla.org/docs/Web/HTML/Element/p, https://developer.mozilla.org/docs/Web/HTML/Element/br

HTML Elemente
Listen

Unsortierte Listen (Unordered List)

```
<ul>
    <li>Apfel</li>
    <li>Birne</li>
    <li>Kiwi</li>
</ul>
```

- Apfel
- Birne
- Kiwi

Sortierte Listen (Ordered List)

```
<ol>
    <li>Apfel</li>
    <li>Birne</li>
    <li>Kiwi</li>
</ol>
```

1. Apfel
2. Birne
3. Kiwi

Weitere Informationen: https://developer.mozilla.org/docs/Web/HTML/Element/ol, https://developer.mozilla.org/docs/Web/HTML/Element/ul

HTML Elemente
Tabellen

Formatierung

- <table> um die Tabelle einzuleiten
- <tr> beginnt eine Zeile (table row)
- <th> beginnt eine Header-Zelle
- <td> beginnt eine normale Zelle (table data)

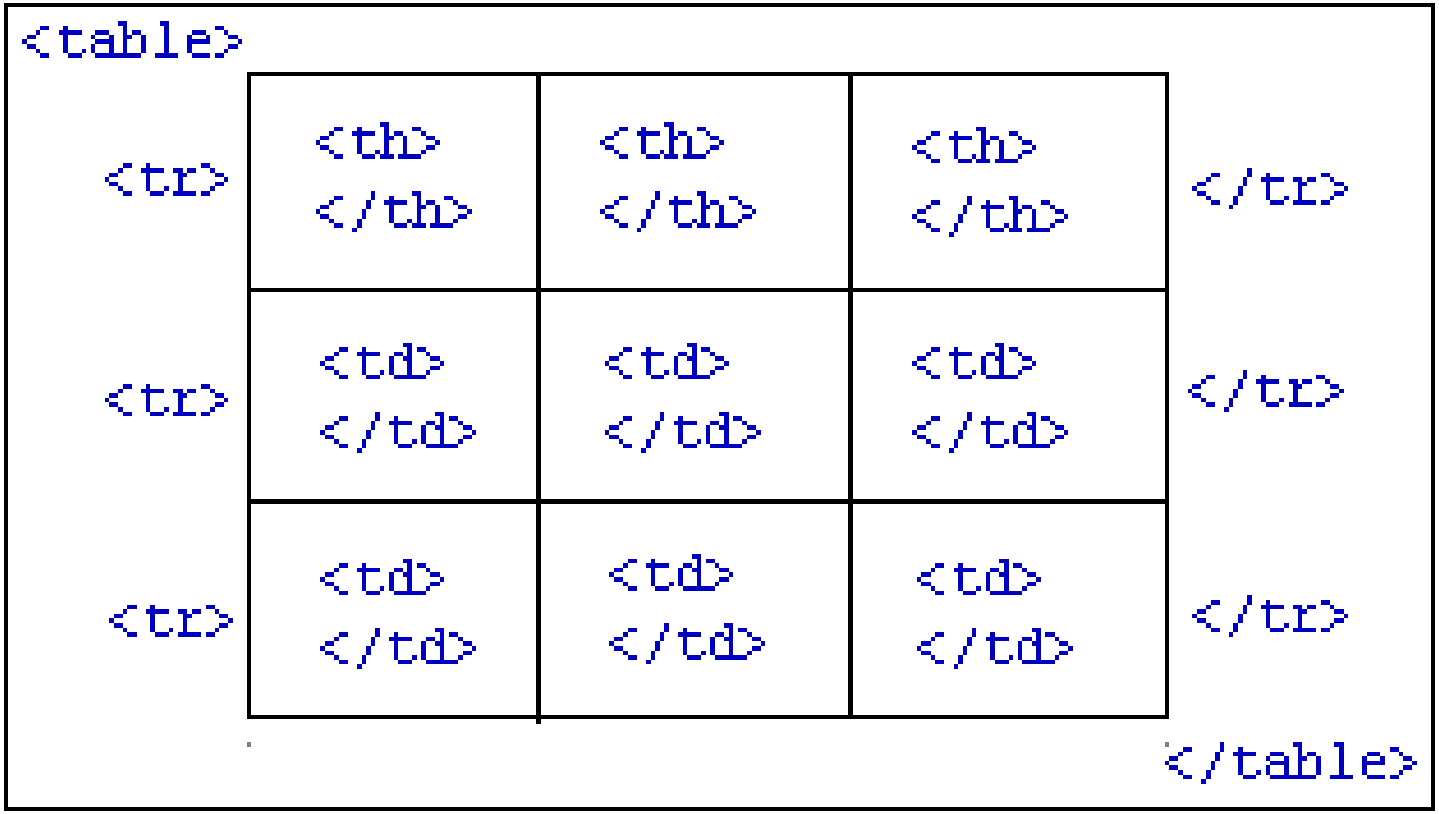
Beispiel

```
<table>
    <tr>
        <th>Land</th>
        <th>BIP (in Billionen USD)</th>
    </tr>
    <tr>
        <td>Deutschland</td>
        <td>3,73</td>
    </tr>
    <tr>
        <td>UK</td>
        <td>2,81</td>
    </tr>
</table>
```

Land	BIP (in Billionen USD)
Deutschland	3,73
UK	2,81

Weitere Informationen: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/table

HTML Elemente
Tabellen



HTML Elemente
Tabellen

Formatierung

- Es ist zusätzlich möglich, dass eine Zelle mehrere Spalten oder Zeilen umfasst.
- `colspan="2"` zum horizontalen Umfassen von zwei Zellen
- `rowspan="2"` zum vertikalen Umfassen von zwei Zellen

Beispiel (mit CSS-Klassen)

```
<table>
    <tr>
        <td colspan="2" class="red">1</td>
        <td rowspan="2" class="green">2</td>
    </tr>
    <tr>
        <td class="blue">3</td>
        <td class="yellow">4</td>
    </tr>
    <tr>
        <td colspan="3" class="red">5</td>
    </tr>
</table>
```

HTML Elemente
Bilder

Bilder einbinden

- <img>
  - > Das Attribut „src“ enthält den Pfad zur Bilddatei
  - > Das Attribut „alt“ enthält einen Alternativtext, der angezeigt wird, wenn das Bild nicht geladen werden kann oder von Screenreadern
- Die meist unterstützten Grafikformate sind
  - > JPEG, GIF, PNG, BMP
  - > https://en.wikipedia.org/wiki/Comparison_of_web_browsers#Image_format_support
  - > Empfehlungen:
    - JPEG für Fotos, Bilder mit vielen Farben oder Farbverläufen
    - GIF für Animationen
    - PNG für Logos, kleinere Bilder oder Bilder mit wenig Farben

Beispiel

```
<img src="itc-logo.png" alt="IT Center">
```

Weitere Informationen: https://developer.mozilla.org/docs/Web/HTML/Element/img

HTML Elemente
Hyperlink

Verbindung zwischen zwei Dokumenten per Link

- Relative und absolute Angaben sind erlaubt
- Beispiele

```
<a href="ziel.html">Verweistext</a>
<a href="https://example.com/datei.html">Ein absoluter Link</a>
<a href="mailto:test@example.com">Email versenden</a>
```

Verweise innerhalb des Dokumentes

- Ein Anker innerhalb des Dokumentes kann auch als Ziel eines Links angegeben werden
- Das Browserfenster spring dann an die jeweilige Stelle
- Beispiel

```
<h2 id="Anker">Überschrift</h2>
<a href="#Anker">Link zur Überschrift</a>
```

- Alternativ zu dem id-Attribut, kann auch das name-Attribut genutzt werden

Weiterführend: https://developer.mozilla.org/docs/Web/HTML/Element/a, http://w3c.github.io/html/browsers.html#navigating-to-a-fragment-identifier

HTML Elemente
Generische Elemente

…

- Generischer Container, der nichts repräsentiert
- Häufig genutzt um Elemente zu gruppieren oder zu stylen (per CSS)
- Wenn bessere semantische Alternativen verfügbar sind, sollten diese verwendet werden:
  - <nav>, <header>, <footer>, <article>, …
- „block"-Element, d.h. der Container beginnt in einer neuen Zeile, was i.d.R. zu einem Zeilenumbruch führt

…

- Generischer Container wie div
- „inline"-Element, d.h. der Container kann auch innerhalb von Text genutzt werden
- Beispiel (mit CSS-Klasse highlight)

```
<span class="highlight">Ich</span> bin hervorgehoben.
```

Weitere Informationen: https://developer.mozilla.org/docs/Web/HTML/Element/div, https://developer.mozilla.org/docs/Web/HTML/Element/span

HTML Elemente
Formulare

Übermittelung von Daten

- Ein Formular enthält Eingabefelder, die das Versenden von Informationen an einen Server erlauben

Beispiel

```
<form action="register_post.php" method="post">
  <label for="l-user">Name:</label>
  <input id="l-user" name="user" type="text"><br>
  <label for="l-pw">Passwort:</label>
  <input id="l-pw" name="pw" type="password"><br>
  <input type="submit" value="Absenden!">
</form>
```

<form>...</form>

- `action="…"` Wohin sollen die Daten gesendet werden
- `method="…"` - Methode zur Übermittlung der Daten
  - > POST oder GET

Weitere Informationen: https://developer.mozilla.org/docs/Web/HTML/Element/form

HTML Elemente
Formulare

Beispiel 2

```
<form action="do.php?q=login" method="post">
  <input type="text" name="username" />
  <input type="password" name="pw" />
  <input type="submit" value="Login" />
</form>
```

- Übermittelt Parameter „username“ und „pw“ per POST-Request
- „GET“-Parameter „q“ enthält den Wert „login“

- Wozu würde `method="get"` führen?
  - > `do.php?username=…&pw=…`
  - > `q=login` würde wegfallen

Weitere Informationen: https://developer.mozilla.org/docs/Web/HTML/Element/form

HTML Elemente
Formulare

- Auswahl an Eingabemöglichkeiten via <input>

<input type="...">	Bedeutung
text	Einfaches (einzeiliges) Eingabfeld
password	Eingabefeld, das die Eingabe verschleiert [*******]
submit	Button zum Übermitteln des Formulars, gute Alternative: <button>
radio	„Schaltknöpfe“ von denen genau ein Feld ausgewählt werden kann
checkbox	Mehrfachauswahl über Kästchen
file	Auswahl einer Datei zum Upload
hidden	Verstecktes Feld, das dem Nutzer nicht angezeigt wird
number	Eingabefeld für eine Zahl
date	Eingabefeld für Datum
datetime	Eingabefeld für Datum und Uhrzeiet
email	Eingabefeld für eine Emailadresse
range	Slider zur Auswahl in einem festgelegten Bereich
url	Eingabefeld für eine URL

Vollständige Liste mit weiteren Live-Beispielen: https://wiki.selfhtml.org/wiki/HTML/Formulare

HTML Elemente
Formulare

<textarea>...</textarea>

- Erlaubt die Eingabe von längerem und mehrzeiligem Text
- cols-Attribut bestimmt die Breite, rows die Höhe
  - > Besser über CSS festlegen
- Beispiel

```
<textarea name="textarea" rows="5" cols="30">Zeile 1
Zeile 2


Superviel Text</textarea>
```

Weitere Informationen: https://developer.mozilla.org/de/docs/Web/HTML/Element/textarea

HTML Elemente
Formulare

<select>...</select>

- Erlaubt die Auswahl eines Wertes (ähnlich zu Radiobuttons)
- <option>-Tags enthalten die Möglichkeiten
- Beispiel

```
<select name="select">
    <option value="value1">Value 1</option>
    <option value="value2" selected>Value 2</option>
    <option value="value3">Value 3</option>
</select>
```

Weitere Informationen: https://developer.mozilla.org/docs/Web/HTML/Element/select

HTML Elemente

Weitere nützliche HTML Elemente

- Seitenstrukturierung
  - > header, nav, aside, main, section, article, footer
- Textstrukturierung
  - > pre, figure, dt, dd, hr
- Textauszeichung
  - > cite, var, time, del, ins
- Multimedia
  - > canvas, svg, iframe, audio, video
- Formulare
  - > fieldset, legend, progress

Weitere Informationen: https://wiki.selfhtml.org/wiki/HTML

HTML Elemente

Dokumentstruktur

- html, head, body

Kopfdaten

- title, meta, link, style, base

Seitenstrukturierung

- body, header, nav, aside, main, section, article, footer, address
- h1, h2, h3, h4, h5, h6 (Überschriften)

Textstrukturierung

- h1, h2, h3, h4, h5, h6 (Überschriften)
- p, pre, blockquote, figure, figcaption
- ol, ul, dl, li, dt, dd (Listen)
- hr
- div

Textauszeichnung

- a (Verweise)
- b, em, i, kbd, mark, s, small, strong, sub, sup, u
- cite, q (Zitate)
- dfn, abbr
- code, var, samp
- time
- ruby, rt, rp
- bdi, bdo
- br, wbr (Zeilenumbrüche)
- del, ins (Änderungsmarkierungen)
- span

Links (Verweise)

- a
- map, area

Tabellen

- table
- caption
- col, colgroup
- thead, tbody, tfoot
- tr
- th, td

Multimedia und Grafiken

- img, picture, map, area,
- canvas, svg, math
- iframe, embed, object, param
- audio, video, source, track

Formulare

- form, fieldset, legend, label, datalist
- input, button, select, optgroup, option, textarea, keygen
- output, progress, meter

Skripte

- script, noscript,
- canvas
- content, decorator, element, shadow, template

Interaktive Elemente

- details, summary
- dialog
- menu, menuitem, command

Besondere Attribute

- Universalattribute
- Eventhandler

Quelle: https://wiki.selfhtml.org/wiki/HTML

CSS - Cascading Style Sheets

CSS

Formatierungssprache für HTML-Dokumente

- HTML zur Strukturierung
- CSS zum Formatieren/Stylen des Dokumentes (auch XML)
- Möglichkeit für verschiedene Ausgabemedien (Bildschirm, Papier) unterschiedliche Darstellungen vorzugeben

Trennung von Inhalt und Design

- Optimalfall: Austausch des Designs ohne Veränderung des Dokuments
- „inline"-style vermeiden und keine font-Tags, o.Ä.
- Klassen, Ids, Elemente mit Formatierungen versehen
- Wiederverwendbare Stil-Elemente schaffen

Beispiele

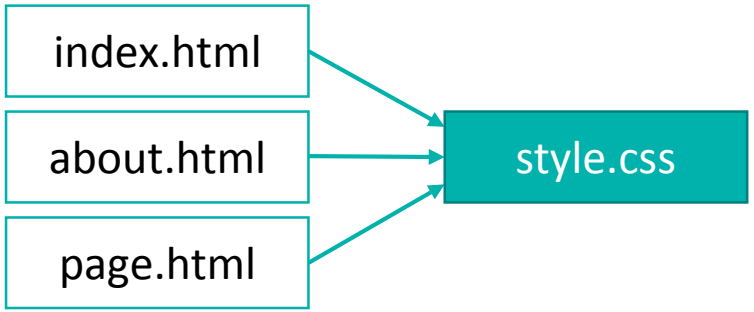
- Farben (Vordergrund, Hintergründe, Rahmen)
- Schriftgrößen, -farben, -arten, -stile
- Textformatierungen, Rahmen, Ränder
- Positionierung (pixelgenau!)
- …

CSS
Einbinden in das Dokument

1. **Einbinden im <head> (empfohlen)**

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

> Wiederverwenden der Formatierungen über mehrere Dokumente hinweg



2. **style-Element im <head>**

```
<style type="text/css">
  h1 { font-size: 200%; }
</style>
```

3. **style-Attribut (unschön, nicht wiederverwendbar)**

```
<h1 style="font-size:200%;">Überschrift</h1>
```

Aufbau eines CSS-Dokumentes

```
Selektor {
    Eigenschaft: Wert;
    Eigenschaft: Wert;
}
Selektor1, Selektor2 {
    Eigenschaft: Wert;
    /* Kommentar */
}
```

> Selektoren sprechen eine Gruppe von HTML-Elementen an
> Beinhaltete Eigenschaften werden auf diese HTML-Elemente angewendet
> Es gibt keine einfachen einzeiligen Kommentare

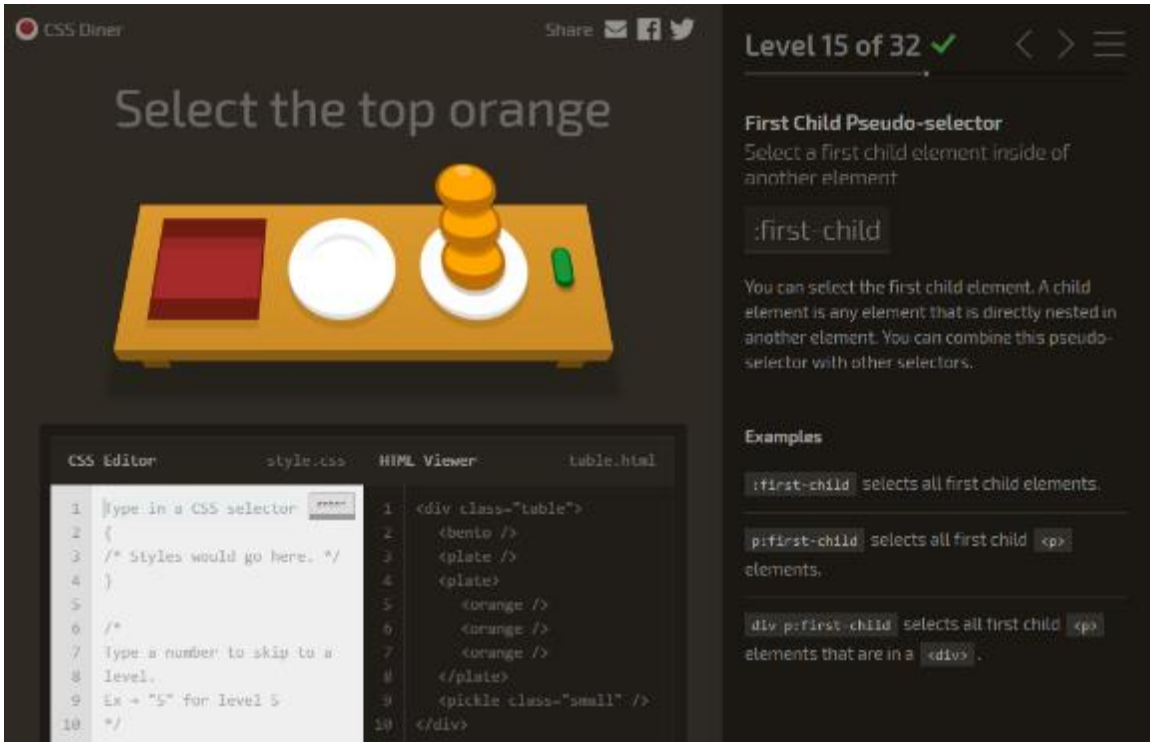
Beispiele

```
h1 { font-size:200%; }      /* <h1>...</h1> */
p strong { ... }             /* <p><strong>...</strong></p> */
p.first  { ... }             /* <p class="first">...</p> */
p, li { ... }                /* p- und li-Tags */
input[name=pw] { ... }       /* <input name="pw" .../> */
.hl { ... }                  /* Alle Tags mit class="hl" */
div#navi { ... }             /* <div id="navi">...</div> */
a:hover { ... }              /* a-Tag beim "Mouseover" */
```

- Es existieren noch viele weitere CSS-Selektoren
  - > https://wiki.selfhtml.org/wiki/CSS/Selektoren
  - > https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

Spiel zum Lernen der CSS-Selektoren

- https://flukeout.github.io/



CSS
Wertangaben

Werte und Maßeinheiten

- Meist besteht eine Angabe aus zwei Teilen:
  - > Wert und Einheit, Beispiel: 20px = (20, Pixel)
- Pixel (px) entspricht der absoluten Basiseinheit, oft gibt es aber sinnvollere Einheiten

Relative Längenangaben

Einheit	Beschreibung	Beispiel
%	Angabe relativ zum Elternelement (bzw. Referenzelement)	`width: 50%`
em	1em entspricht der Schriftgröße des Elementes auf das die Eigenschaft angewendet wird	`border-width: 1em;`
vw, vh	Viewport Units (*viewport width* und *viewport height*) sind relativ zur Größe des Browserfensters	`font-size: 3vw;` `height: 100vh;`

Weitere Einheiten:

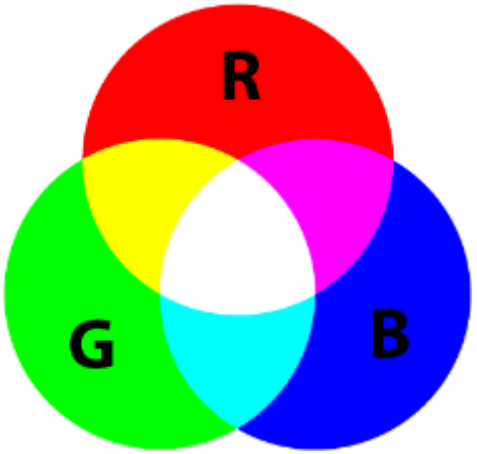
- Für Druckmedien: cm, mm, in
- Weitere relative Einheiten: ex, ch, rem, vmin, vmax

Weitere Informationen: https://wiki.selfhtml.org/wiki/CSS/Wertetypen/Zahlen,_Ma%C3%9Fe_und_Ma%C3%9Feinheiten

CSS
Wertangaben

Farben

- RGB-Farbmodell definiert eine Farbe durch die Angabe der Anteile an (rot, grün, blau)
  - > Werte im Bereich von 0 bis 255
  - > Beispiel: (255 rot, 255 grün, 0 blau) = gelb
  - > Hexadezimale Schreibweite: `#ffff00`
  - > Kurzschreibweise: `#ff0`



- RGBA erlaubt zusätzlich eine Deckkraftangabe
  - > Beispiel: rgba(255, 255, 0, 0.5)
    - – Gelb mit 50% Deckkraft

- Ergänzend gibt es einige fest vorgegebene Farbwerte
  - > Beispiele: red, blue, cyan, yellow

Farbbeispiele (auf weißem Hintergrund)

```
.a {
    color: red;
}
.b {
    color: cyan;
}
.c {
    color: #000;
}
.d {
    color: #ffffff;
}
.e {
    color: #f00;
}
.f {
    color: #32dfa2;
}
.g {
    color: rgba(255, 0, 0, 0.5);
}
```

Text mit Klasse a

Text mit Klasse b

Text mit Klasse c

Text mit Klasse e

Text mit Klasse f

Text mit Klasse g

CSS
Eigenschaften

Text- und Schriftformatierung

```
font-family: Arial, sans-serif;   /* Schriftart */
font-style: italic;               /* kursiv */
font-size: 20px;                  /* Schriftgröße */
font-weight: bold;                /* fett */
color: blue;                      /* Textfarbe (blau) */
text-align: center;               /* zentrieren */
font: 20px Arial, sans-serif;     /* zusammenfassend */
```

- Seltener benötigt:
  - > text-decoration, letter-spacing, font-strech, text-outline, text-stroke
  - > …
  - > https://wiki.selfhtml.org/wiki/CSS

CSS
Eigenschaften

Rahmenformatierung

```
border-width: 10px;     /* Breite */
border-style: solid;    /* Durchgezogene Linie */
border-color: red;      /* Farbe (rot) */
border: 10px solid red; /* Zusammengefasst */
border-radius: 5px;     /* Rundung der Ecken */
```

- Aufschlüsselung der Eigenschaften in (top, right, bottom, left) möglich
  - > border-top, border-right, border-bottom, border-left
- Zusätzlich ist auch eine „outer border“ möglich
  - > outline

CSS
Eigenschaften

Hintergrundformatierung

```
background-color: red;              /* Farbe (rot) */
background-image: url(logo.png);    /* Datei */
background-position: 10px 20px;     /* Verschiebung */
background-repeat: no-repeat;       /* Bild nicht "kacheln" */
```

- Weitere Eigenschaften:
  - -size, -attachment, -clip, -origin, …
  - https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Hintergrundfarben_und_-bilder

Absolute Positionierung

- Element exakt an der Stelle (100, 100) positionieren:

```
position: absolute;
left: 100px;
top: 100px;
```

> nur in seltenen Fällen notwendig

> besser:

```
float: left;
```

Formatierung einer Box

```
width: 100px;                    /* Breite */
height: 100px;                   /* Höhe */
margin: 10px;                    /* Außenabstand */
padding: 5px 10px 5px 10px;      /* Innenabstand */
display: none;                   /* "Display"-Typ */
```

CSS

CSS bietet noch weitere Möglichkeiten zur Formatierung

- Media Queries
- Transitions (animierter Übergang von einem Status in einen anderen)
- Animations (Animation mittels CSS)
- Flexbox
- Grafikfilter
- @-Regeln

- Zusätzlich wird bei Formatierungsschwierigkeiten empfohlen das Box-Modell zu studieren:
  > https://wiki.selfhtml.org/wiki/CSS/Box-Modell

Ergänzend gibt es auch Sprachen, die auf CSS aufsetzen und zusätzliche Features implementieren

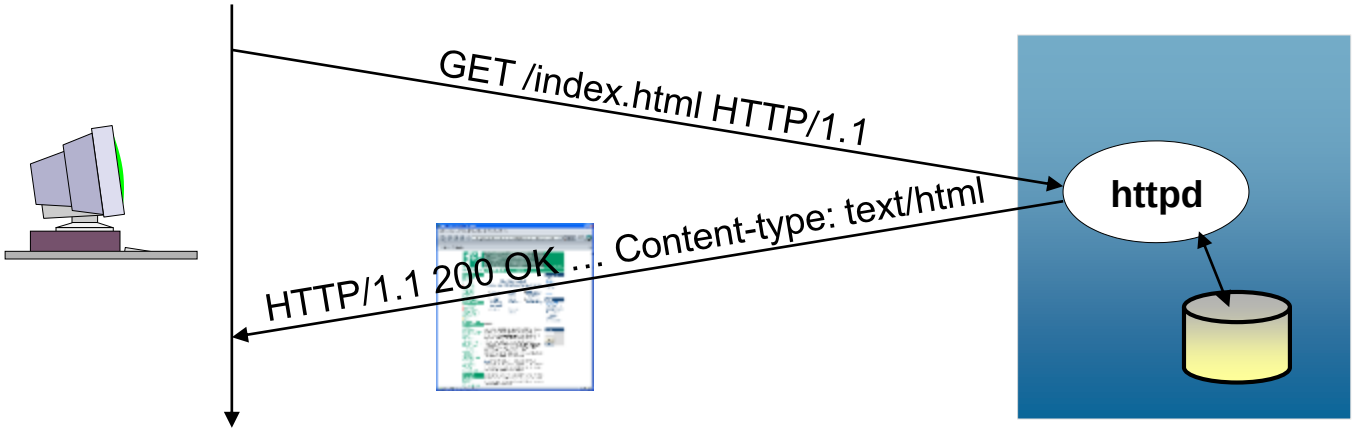
- SASS
- LESS

HTTP - Hypertext Transfer Protocol

Hypertext Transfer Protocol (HTTP)

Das Protokoll des WWW: HTTP

- Zustandsloses Anfrage-/Antwort-Protokoll
- ASCII-basiert (8-bit)
- Dient der Übertragung von hypermedialer Dokumente zwischen einem Server und einem Client
  - > Übertragung von Datenobjekten: Formatangaben, Inhalt, Server-Einstellungen
  - > Kodierungsregeln analog zu E-Mail
- Endpunkt wird über URL identifiziert:
  - > Protokoll://Host:Port/Ort?Parameter



HTTP/1.1 im RFC 2616 (1999): https://tools.ietf.org/html/rfc2616

Im ISO/OSI-Modell

	Schicht	Einheit	Protokoll
7	Application	Anwendung	**HTTP**
6	Presentation		
5	Session		
4	Transport	Transport	TCP
3	Network	Internet	IP
2	Data Link	Netzzugriff	Ethernet
1	Physical		

Client-Server-Modell

- Client sendet *Request*
- Server antwortet mit *Response*

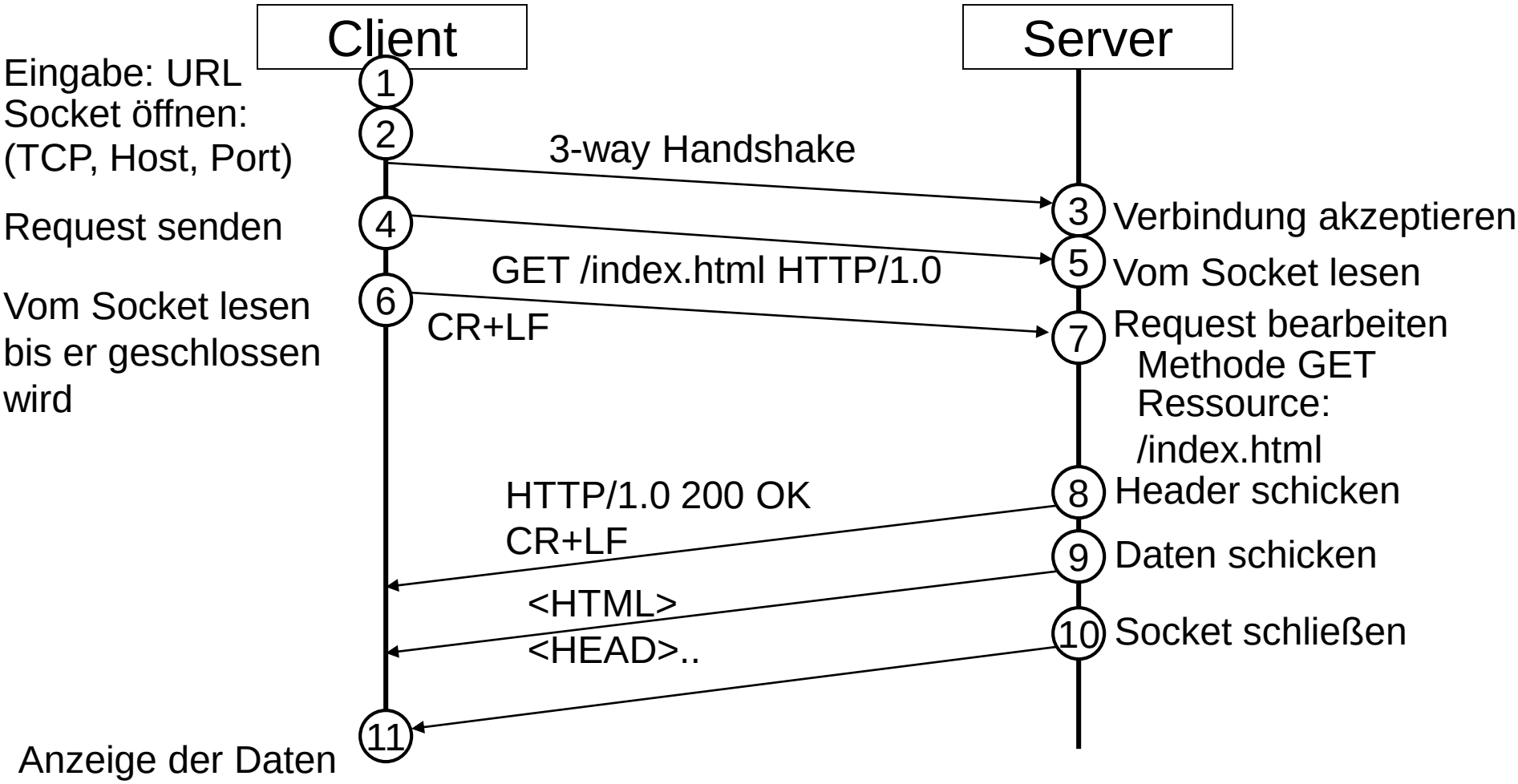
Eigenschaften

- Server stellt einen Dienst zur Verfügung
  - > Webseite: Browser (Client) erfragt Resource, Server antwortet (Response)
- Eine Einheit (z.B. ein Computer) kann gleichzeitig für eine Seite ein Server sein und für eine andere ein Client
  - > Beispiel: Proxy-Server
- Begriff „Server" wird häufig sowohl für den Dienst (z.B. Apache) als auch für die Maschine auf der der Dienst angeboten wird genutzt

Grundsätzlicher Ablauf

1. Verbindungsaufbau
   > Der Client baut eine TCP/IP-Verbindung zum Server auf.
2. Request
   > Der Web-Client sendet einen HTTP-Request, der die Spezifikation des gewünschten Dokumentes in Form von URL und die Protokoll-Version enthält, sowie optional einen Header dem noch eine Liste in Form von MIME-Headern (Multipurpose Internet Mail Extensions) folgen kann, die Client-Informationen und Namen der Zugriffsmethoden für das gewünschte Dokument enthalten.
3. Response
   > Der Server antwortet mit einem Header, der ebenfalls die Protokoll-Version, MIME-Felder und einen Statuscode enthält, der Erfolg bzw. Misserfolg der Anforderung mitteilt. Den Headern folgt das eigentliche Dokument in dem entsprechenden Datenformat (in der Regel HTML).
4. Schließen der Verbindung
   > Bei HTTP 1.0 wurde die Verbindung nun geschlossen. Seit HTTP/1.1 kann der Client mittels „keep-alive" alternativ mitteilen, dass die TCP/IP-Verbindung offen bleiben soll um zukünftige Anfragen zu beschleunigen. Die Schritte 4 und 1 entfallen dann bei zukünftigen Anfragen.

HTTP
Ablauf (Alternative Darstellung)

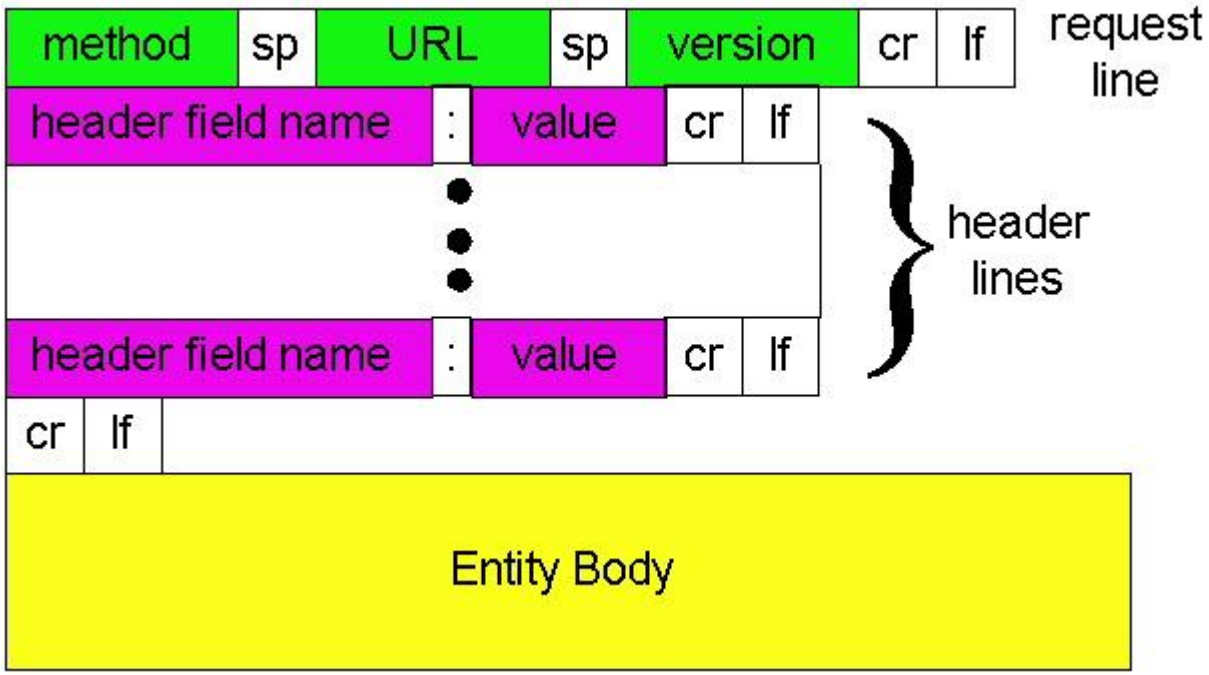


Formaler Aufbau von Request und Response

```
Request          = Request-Line
                   *((general-header | request-header | entity-header)) <CRLF>
                   [message-body]

Response         =  Status-Line
                   *((general-header | response-header | entity-header)) <CRLF>
                   [message-body]

Request-Line     = Method blank Request-URI blank HTTP-Version <CRLF>

Status-Line      =  HTTP-Version blank Status-Code blank Reason-Phrase <CRLF>
```

HTTP/1.1 im RFC 2616 (1999): https://tools.ietf.org/html/rfc2616

HTTP
Request

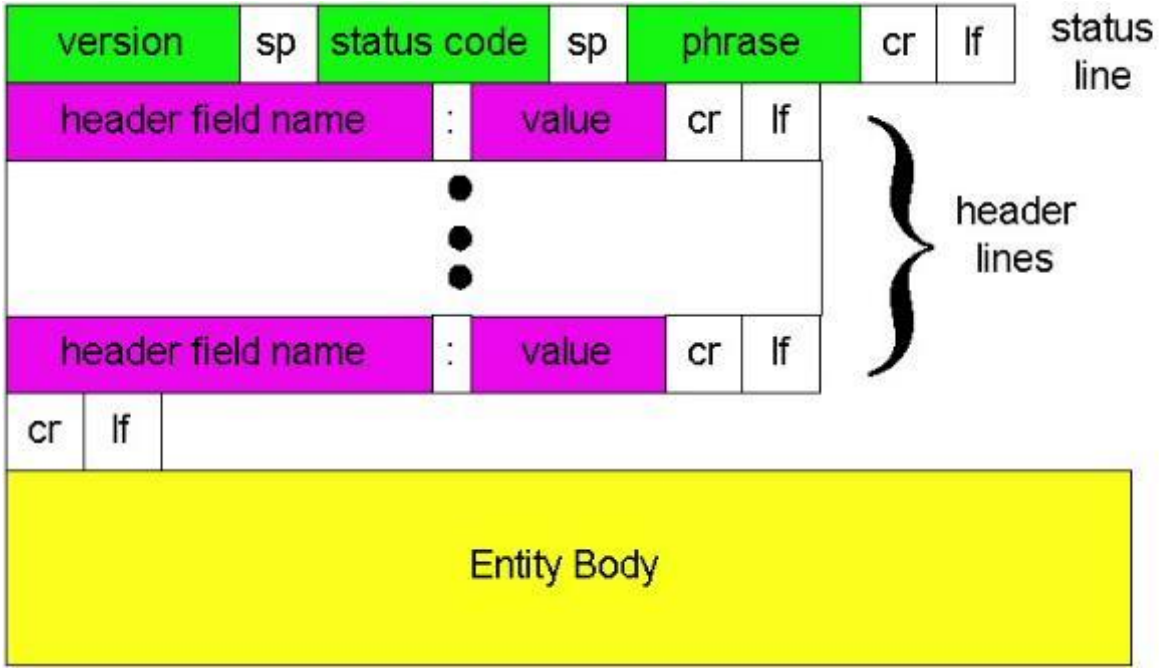


Quelle: http://userpages.umbc.edu/~dgorin1/451/OSI7/dcomm/http.htm

Request-Beispiel (GET):

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
            (KHTML, like Gecko) Chrome/27.0.1453.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-DE, de;q=0.8, en-US;q=0.6, en;q=0.4
Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1
Accept-Encoding: gzip, deflate, sdch
Connection: keep-alive
```

HTTP
Response



Quelle: http://userpages.umbc.edu/~dgorin1/451/OSI7/dcomm/http.htm

Response-Beispiel:

```
HTTP/1.1 200 OK
Date: Wed, 26 Jun 2013 16:36:27 GMT
Server: Apache
Content-Type: text/html; charset=UTF-8
Content-Length: 12313
Last-Modified: Mon, 16 Apr 2013 20:27:06 UTC
Connection: keep-alive

<!DOCTYPE html>
<html>
…
```

HTTP
Response

Statuscodes

Code	Bedeutung
1xx	**Informational** – Anforderung empfangen, Aktion folgt
2xx	**Success** - Erfolgreiche Abarbeitung der Anfrage
3xx	**Redirection** - Angefragtes Objekt befindet sich an anderer Stelle
4xx	**Client Error** - Falsche / Unerlaubte Anfrage
5xx	**Server Error** - Server kann Anfrage nicht ausführen

- Beispiele:
  - > 200 OK
  - > 301 Moved Permanently
  - > 302 Moved Temporarily
  - > 404 Not Found
  - > 500 Internal Server Error

Details im HTTP/1.1 RFC: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

HTTP
Request-Methoden

Methode	Bedeutung
GET	Anforderung, der im Request-URI angeforderten Ressource
POST	Wird zur Übertragung von Daten an den Server benutzt
HEAD	Anforderung des Headers einer Ressource (wie GET, lässt aber den Seiteninhalt weg)
PUT	Wird zur Übertragung einer neuen Ressource an den Server benutzt.
DELETE	Löschen von Dokumenten auf dem Server. Der Server muss entsprechende Rechte auf dem Server besitzen (Sicherheitsprobleme)
OPTIONS	Diese Methode erlaub, den Zugriffspfad zu einer Ressource zu ermitteln und die Methoden, die darüber möglich sind.
TRACE	Für Testzwecke: Die Anfrage wird vom Server zurück geschickt

- Hauptsächlich werden nur GET und POST genutzt, teilweise HEAD
- Andere Methoden werden meist nur von REST-Anwendungen genutzt (wenn überhaupt)

Details im HTTP/1.1 RFC: https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

HTTP
GET-Request

Eigenschaften

- Übertragung der Daten über die Adresszeile
- Parameter werden durch das Zeichen ? in der URL eingeleitet
- Nicht geeignet zur Übertragung großer Datenmengen
- Daten, die per GET übertragen werden, werden häufig geloggt (z.B. vom Server oder einem Proxy) und möglicherweise auch vom Browser gespeichert (im Cache)
  > Keine sensiblen Daten per GET übermitteln

Beispiel

```
GET /index.php?page=welcome&id=42 HTTP/1.1
Host: www.example.com
...
Connection: keep-alive
```

HTTP
POST-Request

Eigenschaften

- Übertragung der Daten im HTTP-Body (≠ HTML-Body!)
  - > Daten sind nicht in der URL sichtbar
- Geeignet für große Datenmengen
- Zusätzliche Übermittlung von Daten mittels GET-Methode möglich
- Daten, die per POST übertragen werden, werden i.d.R. nicht mitgeloggt und auch nicht gecached
  - > Zusätzliche Verschlüsselung ist trotzdem sinnvoll
- Leerzeile trennt HTTP-Header von body (wie bei der Response)

Beispiel

```
POST /edit.php HTTP/1.1
Host: www.example.com
...
Connection: keep-alive

username=thomas&pw=supersecret123
```

Beispiel POST

```
<form action="do.php?q=login" method="post">
  <input type="text" name="username" />
  <input type="password" name="pw" />
  <input type="submit" value="Login" />
</form>
```

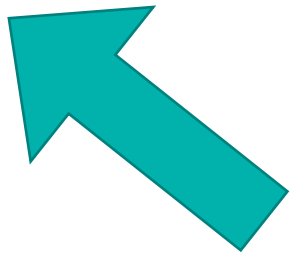
- Erzeugter HTTP-Request (bei entsprechender Eingabe):

```
POST /do.php?q=login HTTP/1.1
Host: www.example.com
...
Connection: keep-alive

username=thomas&pw=supersecret123
```

Weitere Informationen: https://developer.mozilla.org/docs/Web/HTML/Element/form

Beispiel GET

```
<form action="do.php?q=login" method="get">
  <input type="text" name="username" />
  <input type="password" name="pw" />
  <input type="submit" value="Login" />
</form>
```



- Erzeugter HTTP-Request:

```
GET /do.php?username=thomas&pw=supersecret123 HTTP/1.1
Host: www.example.com
...
Connection: keep-alive
```

Weitere Informationen: https://developer.mozilla.org/docs/Web/HTML/Element/form

HTTP 1.0

Ursprüngliches HTTP/1.0 Protokoll von 1996

- RFC 1945: https://tools.ietf.org/html/rfc1945
- Pragmatisch einfaches Protokoll
  - > Einfach zu implementieren
- ASCII-basiert (statt wie z.B. TCP binär)
- Jede einzelne Operation wird über eine separate TCP-Verbindung realisiert
  - > TCP CONNECT => GET index.html => TCP CLOSE
  - > TCP CONNECT => GET image.html => TCP CLOSE
  - > ...
  - > Je eingebundener Resource ist ein weiterer TCP Handshake notwendig
  - > Großen Aufwand beim sukzessiven Laden mehrerer Resourcen

HTTP 1.1

HTTP/1.1 wird 1999 standardisiert

- Ursprünglicher RFC 2616: https://tools.ietf.org/html/rfc2616
  - > Formulierungen in RFCs 7230-7235 verbessert

- Persistente Verbindungen
  - > Der Client kann nun mittels „Connection: keep-alive" angeben, dass er gerne die TCP/IP-Verbindung offen halten möchte
  - > Kompliziertere Implementierung, aber deutliche Verbesserung beim Laden

- Virtual Hosts
  - > Auf einem Rechner sollen verschiedene Domains und Dienste zur Verfügung stehen
  - > Vorherige Lösungen (HTTP/0.9) sahen verschiedene Serverports pro Dienst oder auch mehrere IP-Adressen für einen Rechner vor
  - > Non-IP-based Virtual Hosts lösten dieses Problem entgültig
    - – Im HTTP Request muss der Header *Host* enthalten sein
    - – Beliebig viele Domains können sich dann die gleiche IP-Adresse teilen

HTTP 2.0

HTTP/2 wurde 2015 veröffentlicht

- RFC 7540 (und 7541): https://tools.ietf.org/html/rfc7540

- Weitere Beschleunigung der Datenübertragung
- Abwärtskompatibel zu HTTP/1.1

- Wichtigste Neuerungen
  - > Mehrere Anfragen in einer Übertragung zusammenfassen
  - > PUSH-Verfahren, also Datenübertragungen, die vom Server initiiert werden können

URI: Uniform Resource Identifier

- Eine URI dient der eindeutigen Adressierung von abstrakten und physikalischen Ressourcen im Internet (Spezifikation in RFC2396)
- URI: URLs ∪ URNs

URI (**U**niform **R**esource **Identifier**)

URL (**U**niform **R**esource **L**ocator)
Adressierung von Informationsobjekten mit Festlegung des Zugangs-Protokolls (Ort der Ressource). RFC2141

URN (**U**niform **R**esource **N**ame)
Adressierung von Objekten ohne ein Protokoll festzulegen (Eindeutige und gleichbleibende Referenz – Name der Ressource). RFC1738

Weitere Informationen: https://www.ietf.org/rfc/rfc3986.txt

Unterschied URL/URI

- URI beschreibt allgemeinere Möglichkeit der Adressierung
- Jede URL ist auch eine URI
- Eine URL identifiziert Ressourcen mittels der Repräsentation ihres hauptsächlichen Zugriffsmechanismus und nicht über Namen oder andere Attribute der Ressource

Beispiele:

- URI: example.com
- URI (oder eine relative URL): datei.txt
- URL: http://www.example.com/datei.txt
- URL: mailto:test@example.com
- URL: ftp://127.0.0.1/dump.sql
- URN: urn:isbn:0596517742

> URLs und URNs sind auch gleichzeitig gültige URIs

HTTP
Adressierung

URL-Syntax

- <Schemata>:<Schemata-spezifischer-Teil>
- Viele URL-Schemata besitzen einen hierarchischen Aufbau.

Beispiel: HTTP URL

```
http://<user>:<password>@<host>:<port>/<url-path>?<searchpart>
```

```
http://user:pw@example.com:80/datei.html?q=123
```

Relative URLs

- Auch die Nutzung relativer URLs wurde definiert
- Beispiel: `../datei.html`
- Nicht relative URLs werden als „absolut“ bezeichnet

Weitere Informationen: https://www.ietf.org/rfc/rfc1738.txt, https://tools.ietf.org/html/rfc1808

Basis-URL

- Für das Verständnis relativer URLs ist ein Analogon zur „Working Directory“ wichtig, die Basis-URL
- Die „Basis-URL“ einer mittels einer absoluten URL-spezifizierten Internet-Ressource beinhaltet alle Zeichen bis (einschließlich) zum letzten Schrägstrich (/) im Pfadnamen

Absolute URL	Basis URL
http://www.fh-aachen.de/abt_juelich.html	http://www.fh-aachen.de/
http://www.example.com/dir/inhalt.html	http://www.example.com/dir/

Relative URL

- Eine partielle (relative) URL ist immer dann vorhanden, wenn Schemataname und ":" fehlen.
- Aus der Kombination von Basis-URL (die, der aktuell angezeigten Seite im Browser) und relativer URL lässt sich immer eine absolute URL ableiten

HTTP
Adressierung

Beispiel:

- Basis-URL: `http://www.example.com/html/`

Relative URL	Absolute URL
about.html	http://www.example.com/html/about.html
path/	http://www.example.com/html/path/
path/file.html	http://www.example.com/html/path/file.html
/	http://www.example.com/
//example.org/	http://example.org/
/config/	http://www.example.com/config/
../	http://www.example.com/
../dir/	http://www.example.com/dir/
../../../	http://www.example.com/
./	http://www.example.com/html/
./about.html	http://www.example.com/html/about.html

Dateinamen

- Um Konflikte zu vermeiden sollten nur ASCII-Zeichen als Dateiname genutzt werden, u.a.: a-z A-Z - _ . ! ~ * ( ) 0 -9
- Andere Zeichen können zwar verwendet werden, aber müssen kodiert werden (auch Leerzeichen)
- Diverse ASCII-Zeichen wie u.a. %+&=: sind bereits für andere Aufgaben vorgesehen und können ebenfalls so nicht direkt auftauchen.

- Für all diese Zeichen gibt es deshalb eine andere Darstellungsform
  - > Leerzeichen werden zu einem „+“ kodiert („%20“ funktioniert alternativ)
  - > Nationale Sonderzeichen werden durch ihre hexadezimale Darstellung entsprechend ASCII-Zeichensatz dargestellt. Zur Erkennung dieser Hex-Zeichen wird ein "%" vorangestellt.
    - Beispiele: ß wird zu %DF,] zu %5D, } zu %7D
  - > Browser zeigen heutzutage häufig die dekodierten Werte an, was dazu führt, dass der Nutzer häufig gar nichts von der Umwandlung merkt

Weitere Informationen: https://tools.ietf.org/html/rfc3986