Webgrundtechniken

Hochschule Ravensburg-Weingarten
Technik | Wirtschaft | Sozialwesen

Wintersemester 2011/2012

LV 4051

Bachelorstudiengang: Wirtschaftsinformatik / WI Plus

Karl Glatz, BSc.

karl.glatz@hs-weingarten.de

PDF Version | HTML5 Version

Teile der Präsentation basieren auf Arbeiten von Dr. Stefan Müller

INTRO:

Lerziele und Organisation

Lernziele

- Vermittlung elementarer Techniken des WWW
- Beherrschung der grundlegenden Elemente einer statischen Webseite
- Grundsätze zur Erstellung einer modernen Web-Präsentation
- Vertiefung der Konzepte und Grundsätze im Rahmen einer praktischen Arbeit
- Erstellung einer Webseite auf Basis von (X)HTML und CSS
- Prinzip von modernen Webanwendungen verstehen

Organisation

- Vorlesung Montags 14:15 in H061
- Übungen, Montags 14:15 in V206/V208 (zwei Räume paralell!)
  - Übungsaufgaben sind Pflicht! (mindestens 80%)
  - Testate durch Tutoren
- Prüfungsleistung: Praktische Arbeit
  - Durchführung eines umfangreicheren **Web-Projekts** (in 2-3er Gruppen)
    - Erstellung einer Web-Präsentation für eine Firma, ein Projekt, Verein etc. (fiktiv oder real)
    - Beginn: 28.11.2011 (spätestens)
    - Meilenstein 1: 18.12.2011 / Meilenstein 2: 8.01.2012
    - Finale Präsentation des Projekts (mündliche Prüfungsleistung): 23.01.2012 / 30.01.2012

Kalender Download

Kalender als iCal-Link

E-Learning Plattform

Link: elearning.hs-weingarten.de

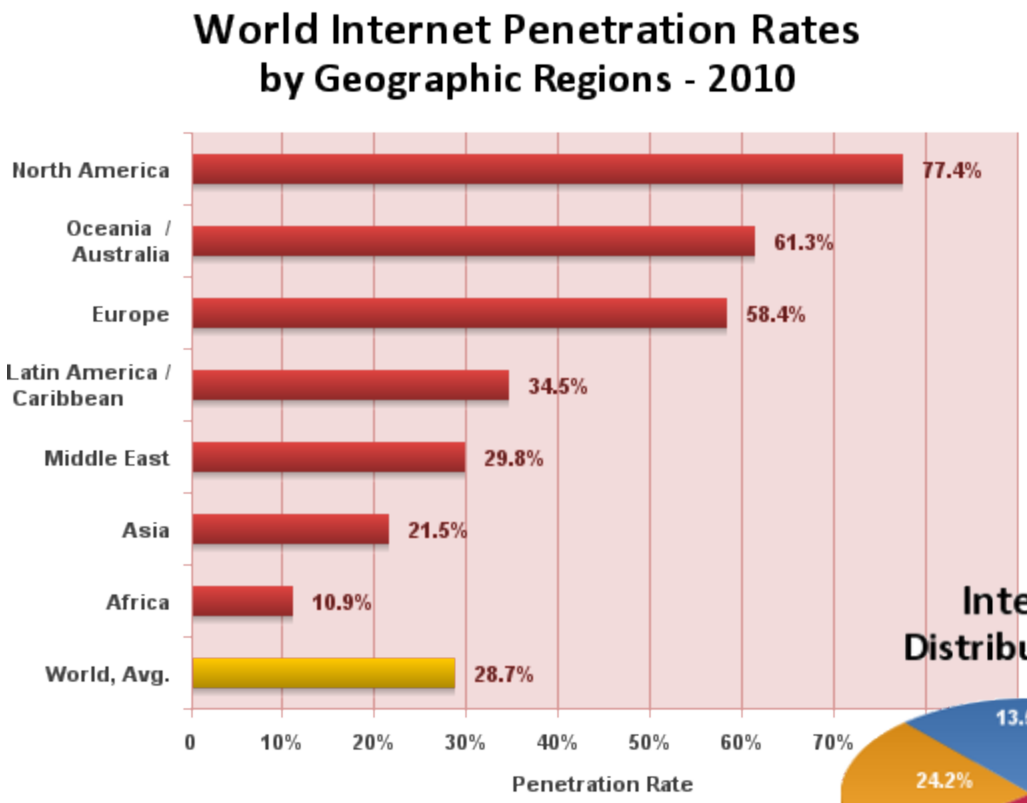
INTRO:

Internet und WWW

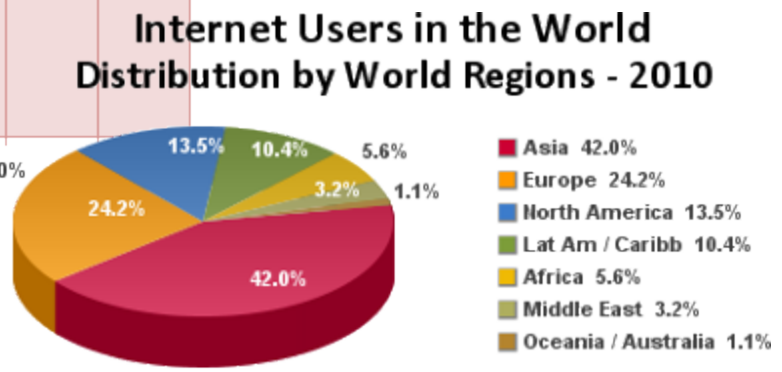
Entstehung des Internet

- Vom ARPANET zum Internet
- 1969: ARPANET als erstes paketorientiertes Netzwerk
- 1977: Anschluss weiterer Netzwerke an das ARPANET via TCP/IP
  - TCP (Transmission Control Protocol)
    Paketorientiertes Datenübertragungsprotokoll (auf der 4. Schicht des OSI-Referenzmodells)
  - IP (Internet Protocol)
    Vermittlungsprotokoll (auf der 3. Schicht des OSI-Referenzmodells)
    Vermittlung zwischen Subnetzen mit unterschiedlicher Netzwerktechnologie (auf der 1. und 2. OSI-Schicht)
- ⇒ Geburt des Internet als Netz aus verschiedenen Teilnetzen (interconnected networks)

Verbreitung des Internet

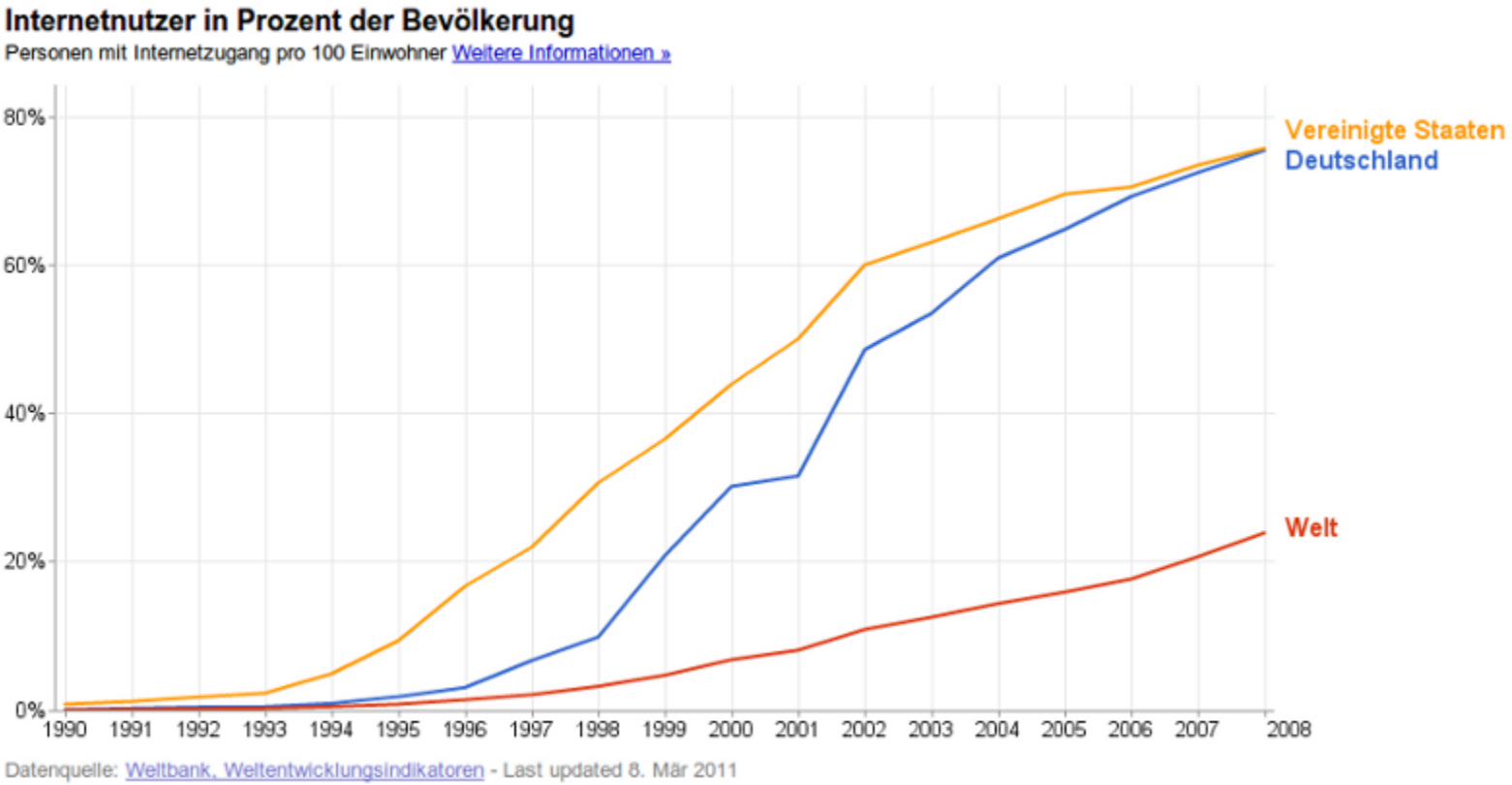


Source: Internet World Stats - www.internetworldststs.com/stats.htm
Penetration Rates are based on a world population of 6,845,609,960 and 1,966,514,816 estimated Internet users on June 30, 2010.
Copyright © 2010, Miniwatts Marketing Group



Source: Internet World Stats - www.internetworldstats.com/stats.htm
Basis: 1,966,514,816 Internet users on June 30, 2010
Copyright © 2010, Miniwatts Marketing Group

Verbreitung des Internet #2



Die Dienste des Internet

- World Wide Web (WWW)
  Standard zur Übermittlung von Multimedia-Dokumenten im HTML-Format.
  Protokoll: HTTP (Hypertext Transfer Protocol)

- File Transfer (Übertragung von Dateien)
  Standard zur Übertragung von Dateien

- Protokoll: FTP (File Transfer Protocol)

- E-Mail (elektronische Post)
  Standard für den Versand von E-Mails über das Internet
  Protokoll: SMTP (Simple Mail Transfer Protocol), POP3 (Post Office Protocol), IMAP

- Newsgroups (auch Usenet)
  Übertragung von Netzwerk-Nachrichten
  Protokoll: NNTP für Net News Transfer Protocol

Die Dienste des Internet

- WAP (Wireless Applications)
  Internet-basierter Dienst zur Übertragung von WML-Dokumenten für Mobilfunkendgeräte
  Protokoll: WAP (Wireless Application Protocol)

- TELNET / SSH
  Terminal-Emulator zum Einloggen und Arbeiten auf entfernten Rechnern

- Voice over IP (VoIP)
  Telefonieren über das Internet
  Protokoll: SIP (Session Initiation Protocol)

- Chat / Instant Messaging
  IRC (Internet Relay Chat)
  XMPP (Jabber)

Entstehung des World Wide Web

- Das WWW als weltweites Hypertextsystem
  - 1990: Entwicklung eines weltweiten Hypertextsystems auf Basis des Internets (Tim Berners-Lee)
- Die Säulen des WWW
  - **HTML**: Spezifikation einer Auszeichnungssprache für Web-Dokumente
  - **URIs** (Universal Resource Identifiers): Spezifikation für die Adressierung beliebiger Datenquellen im Internet
  - **HTTP-Protokoll**: Spezifikation für die Kommunikation zwischen Web-Clients und Web-Servern

Geschichte der Web-Browser

- Mosaic - Erster graphischer Browser
- Netscape 4.x
- Microsoft Internet Explorer 6.0

⇒ Konkurrenzkampf gekennzeichnet durch die Entwicklung proprietärer Features

W3C WORLD WIDE WEB consortium

Initiative zur (kontrollierten) Weiterentwicklung des WWW Definition weltweiter Standards für das WWW HTML, CSS, XML, …

Web-Browser Marktanteile

Datum	Firefox	IE	Chrome	Safari	Opera
Q4 2011	47,3 %	30,6 %	8,3 %	5,2 %	2,4 %
Q3 2011	48,5 %	31,2 %	7,4 %	5,1 %	2,4 %
Q2 2011	49,1 %	32,5 %	6,2 %	5,1 %	2,6 %
Q1 2011	49,6 %	34,6 %	5,1 %	4,5 %	2,6 %
Q4 2010	50,4 %	35,7 %	4,1 %	4,2 %	2,7 %
Q3 2010	50,7 %	37,1 %	3,3 %	3,9 %	2,7 %
Q2 2010	51,3 %	37,5 %	2,8 %	3,7 %	2,7 %
Q1 2010	50,0 %	40,1 %	2,1 %	3,5 %	2,7 %
Q4 2009	48,0 %	43,4 %	1,5 %	3,1 %	2,6 %
Q3 2009	46,1 %	46,1 %	1,1 %	2,9 %	2,5 %

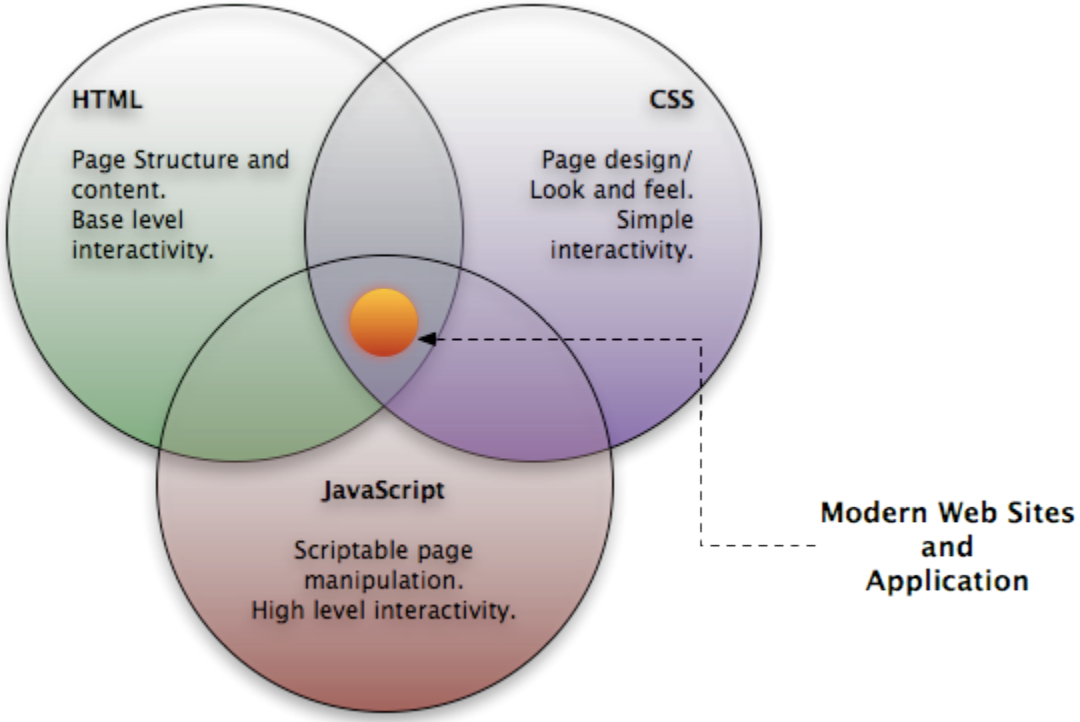
Hypertext

- Hypertext als nicht-lineares Medium
  - **Pro**
    - Flexibler Zugang zu Wissen (Nachschlagewerke)
    - Vernetztes Wissen kann leichter aufgenommen werden (wissenschaftlich nicht eindeutig belegt)
  - **Contra**
    - Steigende Komplexität
    - Gefahr des “Lost in Hyperspace”
  - Kohäsive Geschlossenheit
    - Verständlichkeit der Informationseinheiten unabhängig vom Verweiskontext
    - Einordnung von Informationseinheiten in ihren Kontext durch Verlinkung

HTML und XHTML

- HTML (Hypertext Markup Language)
  - Sprache zur Auszeichnung von Texten bzw. deren Elementen (Überschriften, Absätze, Listen, Tabellen, usw.)
  - Definiert mittels SGML (Standard Generalized Markup Language)
  - Aktuell in der Version 4.01 (5.0 ist im „Working Draft“ State)
    - Rückbesinnung auf Kernaufgaben
    - Auslagerung von Layoutangaben (CSS)
    - Sprachvarianten Strict, Transitional und Frameset
- XHTML
  - HTML definiert mittels XML (XML Parser sind einfacher als SGML Parser)
  - XHTML 1.0 entspricht HTML 4.01
  - XHTML 1.1 reduziert auf Variante Strict

Web-Technologien



HTML5

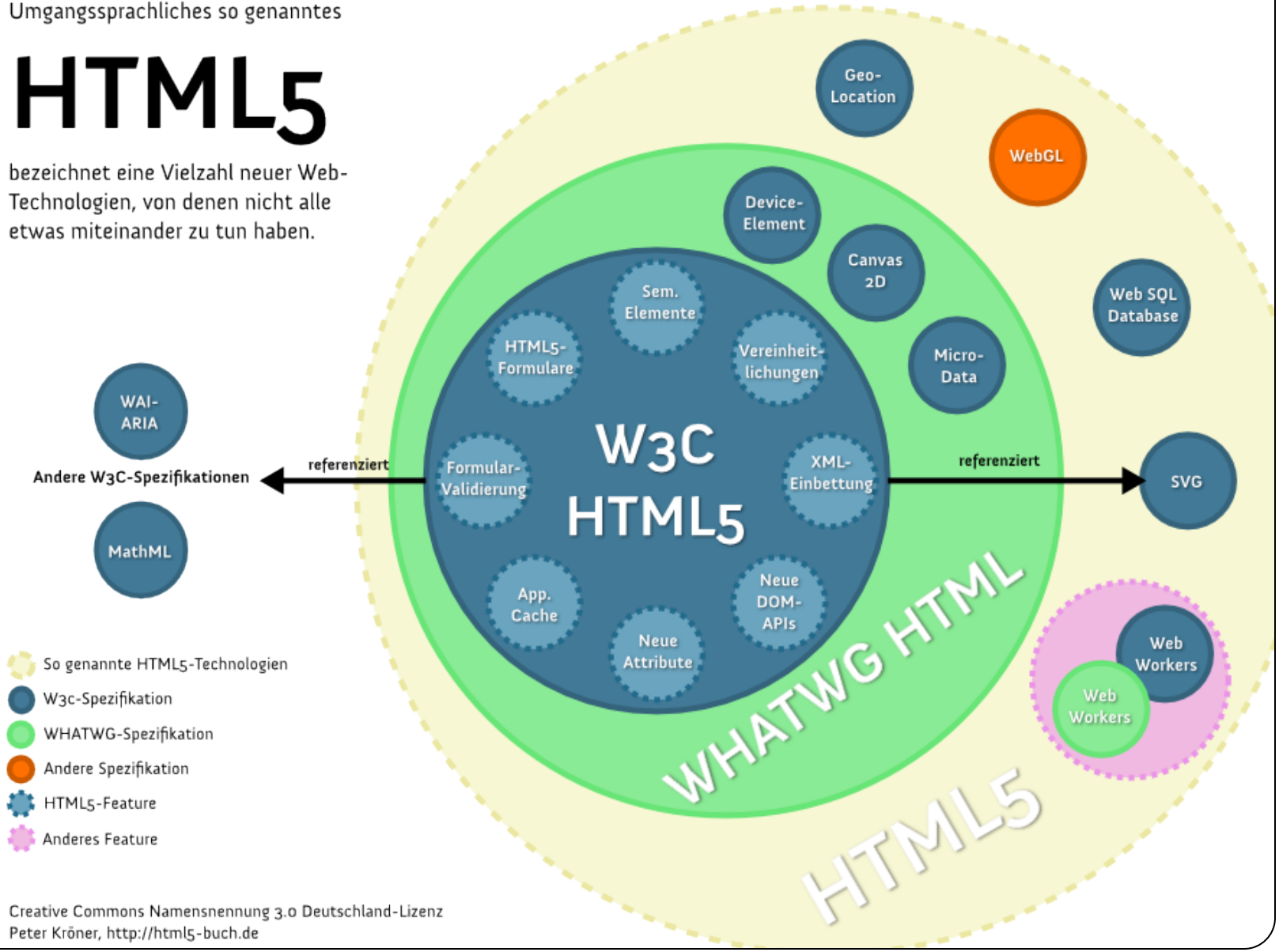
- Neue Tags
  - Audio/Video
  - <article>, <section>, <header>
- SVG: Vektor Grafik
- CSS3
  - Schatten
  - Runde Ecken
  - Animation
  - Mehr-Spaltiger Text
- Javascript APIs -Canvas - WebGL - Offline Storage
- Test unter: www.html5test.com

HTML5 #2

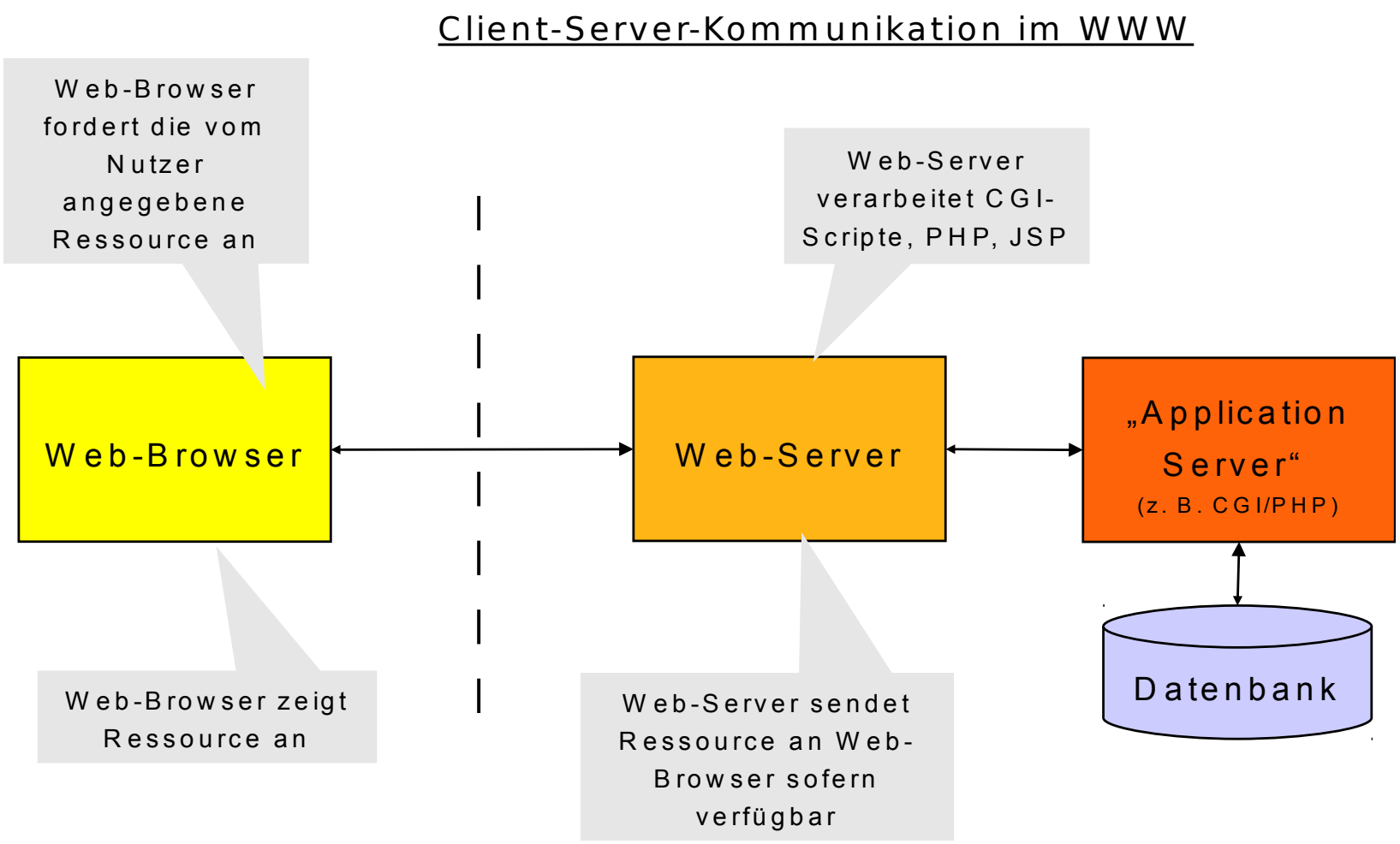
Umgangssprachliches so genanntes

HTML5

bezeichnet eine Vielzahl neuer Web-Technologien, von denen nicht alle etwas miteinander zu tun haben.



Web-Browser und Web-Server



Client Web-Technologien
JavaScript & DOM

Dynamisierung von HTML-Dokumenten im Web-Browser Verarbeitung von Maus- oder Tastatureingaben und Ausgaben am Bildschirm bzw. Veränderung des HTML-Dokuments DOM (Document Object Model): Schema für den Zugriff auf HTML-Dokumente Javascript Libraries (jQuery, MooTools, Dojo etc.) Erleichtern die Programmierung von dynamischen Seiten

GWT

Google Web Toolkit, RIAs mit Java programmieren, ohne Browser Plugin ActiveX & Java Applets (Plugins) Veraltete Microsoft-Technologie für dynamische „Webseiten“ Veraltete Methode um Java Code im Browser auszuführen

Flash & Silverlight (Plugins)

Modernere Technologien für RIA (Rich Internet Applications) Flash dient oft als „lückenschließer“ für fehlende HTML/JS Features

Server Web-Technologien
CGI & Perl

CGI (Common Gateway Interface): Web-Server-Schnittstelle für den Programmzugriff

Perl (Practical Extraction and Report Language)

Script-Sprache zur Automatisierung von Datenzugriffen und -weiterverarbeitung

PHP (PHP Hypertext Preprocessor)

Programmiersprache zur server-seitigen Dynamisierung von HTML Web-Server verarbeitet PHP-Code und modifiziert HTML-Dokument

JSP (Java Server Pages)

Aufruf von Java-Applikationen durch den Web-Server

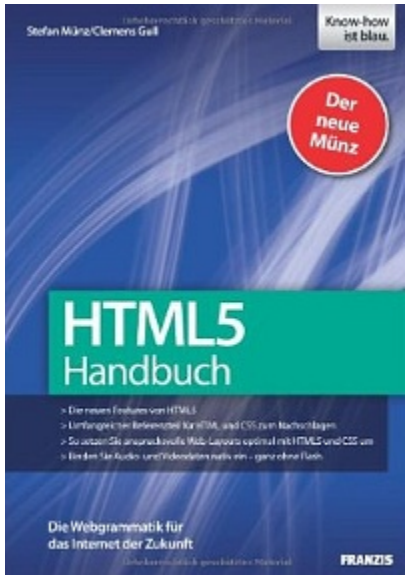
ASP/.NET

ASP (Active Server Pages): Microsoft-Gegenstück zu JSP .NET: Web-Applikationsframework (incl. C#)

Python/Django, JSF uvm.

(Online-)Literatur
Buch

- HTML 5 Handbuch
  Kostenlos Online verfügbar
  http://webkompetenz.wikidot.com/docs:html-handbuch
  Vom Self-HTML Gründer Stefan Münz (2010)
- HTML5 Buch
  - http://html5-buch.de/
  - Viele Demos und Aufgaben



Nachschlagewerke Online

- SELFHTML. www.de.selfhtml.org
- Online-Internet-Kurs. www.www-kurs.de

INTRO:

Planung und Durchführung eines Web-Projekts

Vorgehensweise für Web-Design #1

Strategische Positionierung

- Zielsetzung & **Zielgruppe** der Webseite
  - Kommunikation und Erreichbarkeit, Information, Interaktion, Verkauf, Service- und Kundenbindungsinstrument
  - Festlegung des beabsichtigten Mehrwerts (z.B. neue Kundensegmente, bessere Kundenbindung, etc.)
- **Wettbewerb**: Vergleich mit Wettbewerbern
- **Marketing**: Integration in Marketing-Strategie

Vorgehensweise für Web-Design #2

Realisierung einer Webseite

- Festlegung der Inhalte und Struktur (Navigation)
- Festlegung der Funktionen (Value-Added Services)
- Graphisches Design
- Technologie
- Design Guidelines, z.B. www.useit.com (Jakob Nielsen)

Vorgehensweise für Web-Design #3

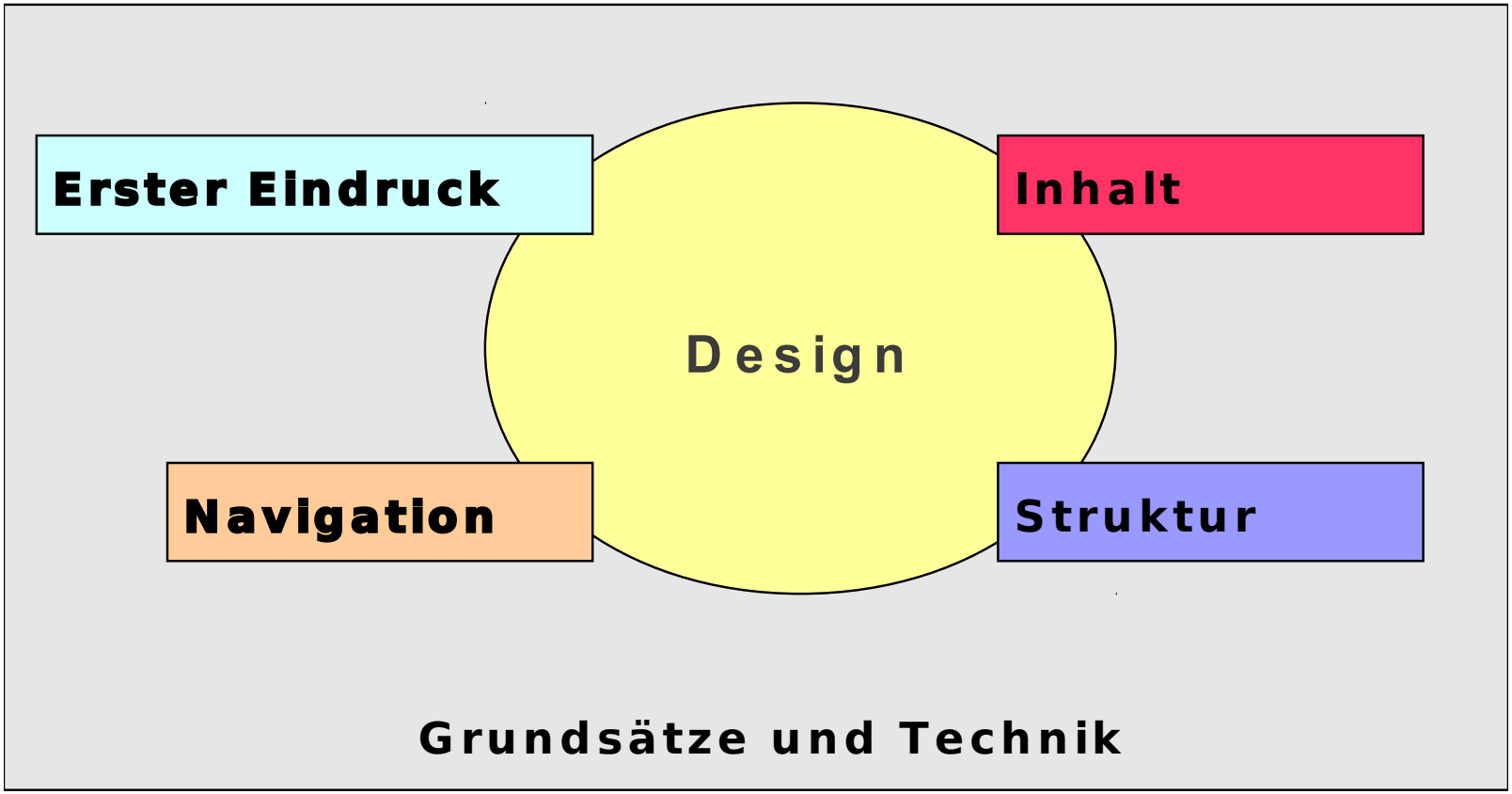
Integration

- Organisatorische Integration
  - Pflege und Aktualisierung der Inhalte
  - Wartung und Betrieb
- Technische Integration
  - Einbindung in bestehende Infrastruktur

Monitoring

- Kundenfeedback
- Kosten-/Nutzenanalyse
- Wettbewerbsanalyse (Benchmarking)
- Technologische Entwicklung

Kriterienkatalog für Webseiten



Beurteilungskriterien #1

Erster Eindruck

Ist die Präsentation übersichtlich?
Ist das Grundthema klar erkennbar?
Wirkt ein einheitlicher angemessener Gesamteindruck?

Inhalt

Erhält man auf der Startseite einen Überblick über den Inhalt?
Ist der Informationsgehalt ausreichend?
Ist die Zielsetzung klar ersichtlich?
Sind die für die Nutzung erforderlichen Funktionen vorhanden?
Ist der Inhalt nach Themen geordnet?

Beurteilungskriterien #2

Struktur

Ist die Struktur auf den ersten Blick durchschaubar und leicht nachvollziehbar?
Sind die einzelnen Themenblöcke gut strukturiert?
Sind Navigationselemente und Links erkennbar? Ist die Verlinkung sinnvoll und umfangreich?

Navigation

Ist die Navigation bzw. Hierarchie nachvollziehbar?
Ist der Standort innerhalb der Web-Site jederzeit erkennbar?
Ist die Navigation durchgehend und prägnant?
Ist der Schritt zur Startseite jederzeit möglich?

Beurteilungskriterien #3

Design

Steht der Inhalt durch das Design weiterhin im Vordergrund?
Spiegelt das Design die Zielsetzung wider?
Spricht das Design die Zielgruppe an?
Ist genügend Farbkontrast vorhanden?
Sind die benutzten Farbtöne harmonisch?
Ist das Design durchgehend?
Ist der Text lesbar, die Schriftart und -größe angemessen?
Sind besuchte Links markiert?

Beurteilungskriterien #4

Grundsätze und Technik

Ist die Präsentation unabhängig vom Browser (IE, Firefox, Chrome, Safari)?
Passt sich die Darstellung den Anforderungen des Nutzers an (Auflösung, Schriftgrößen)?
Sind Inhalt und Design strikt getrennt (HTML und CSS)?
Ist die Präsentation bezüglich Inhalt, Struktur und Design ausreichend komplex?
Sind die verwendeten HTML- und CSS-Elemente sinnvoll eingesetzt?
Ist ein alternatives Design für den Ausdruck vorhanden?

HTML Basics

HTML Bascis: Inhalt / Ziel

- Was ist HTML?
- HTML Grundgerüst
- Textformatierung
- Links
- Listen
- Weitere Textauszeichnungen (physisch und logisch)
- Allgemeine HTML Elemente
- Bilder einbinden

=> Vertiefung der Kentnisse in Übung 1 (nächste Woche).

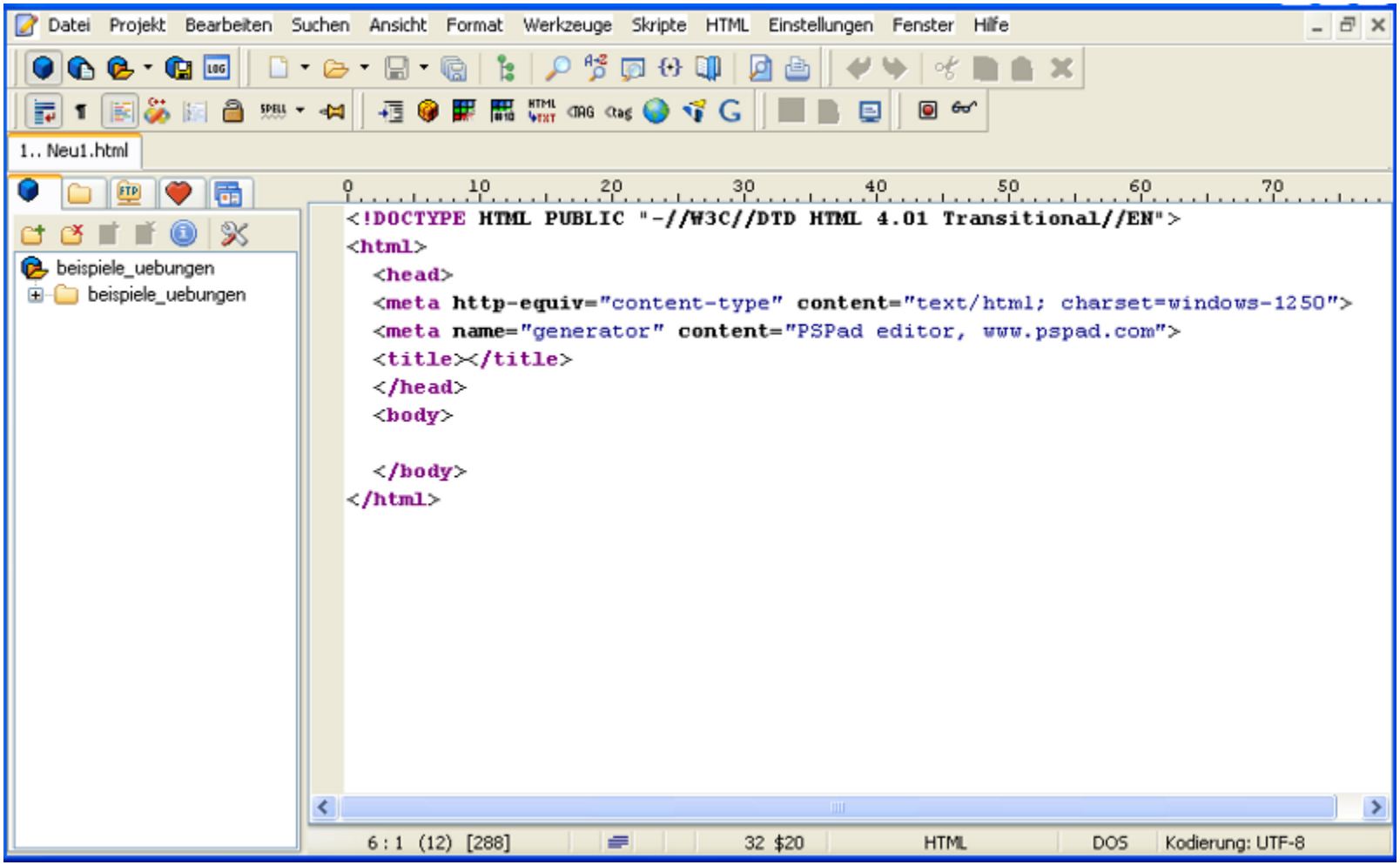
Das HTML Format

- HTML Dateien sind Text-Dateien
- Bearbeitbar mit einfachem Texteditor (z. B. Notepad, gedit, TextMate)
- Spezialisierte HTML Editoren
  - Windows: PSPad / Notepad++
  - Linux: gEdit / Kate / Geany
  - OSX: TextMate
- Dateiendung *.html oder* .htm
- Programm zum Anzeigen einer HTML-Datei: Browser
  - Chrome, Firefox, Safari, Opera, IE

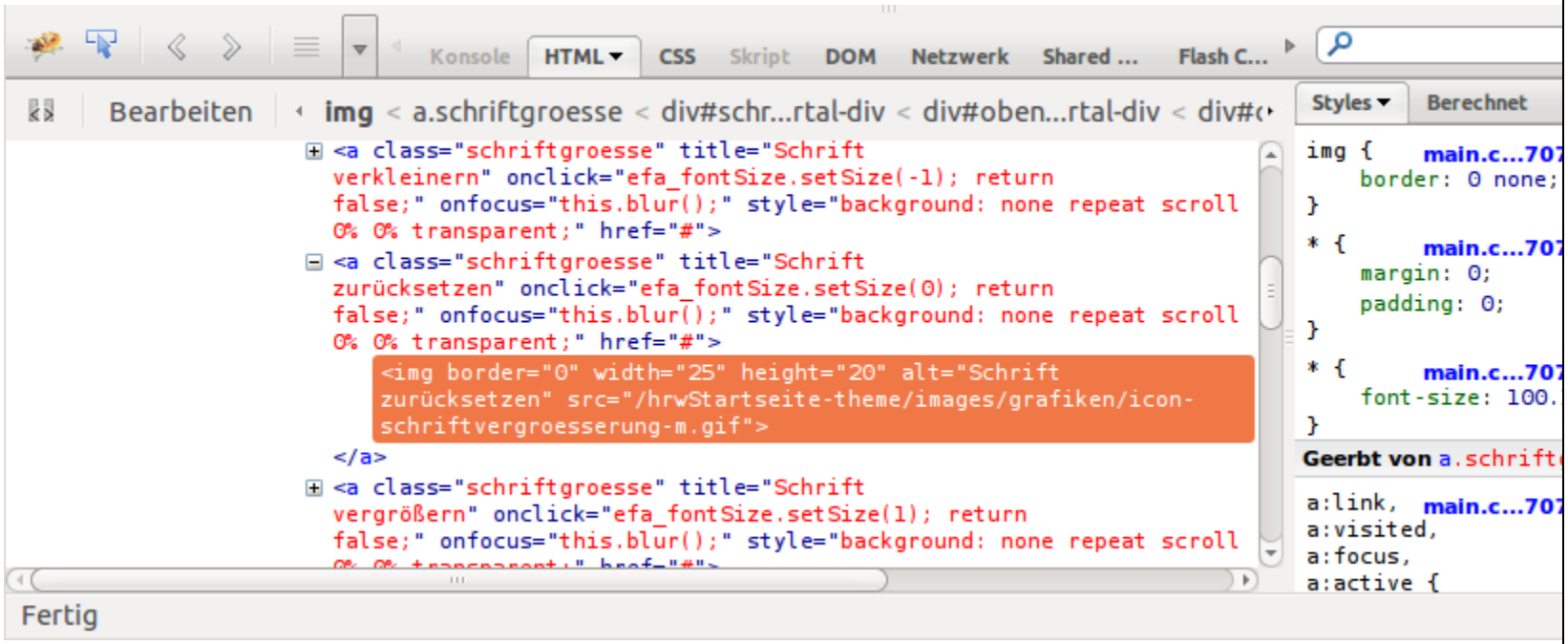
Aufgabe von HTML

- Idealisiert:
  - Textauszeichnung
    - Bedeutung von Textelementen festlegen (Semantik)
  - NICHT Textpräsentation, Layout, Design
    - CSS
- Realität:
  - Bau von Webseiten im Zusammenspiel mit CSS und JavaScript

Der HTML-Editor PSPad



Entwicklertools im Browser



- Firefox: Addon Firebug
- Chrome: Schraubenschlüssel => Tools => Entwicklertools | UMSCHALT+STRG+I | F12
- Safari: ALT+CONTROL+I
- Opera: Tools => Advanced => Opera Dragonfly

Aktuelle Versionen vom Internet Explorer bieten ebenfalls Entwicklertools.

HTML4-Datei Aufbau

```
 1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 2 "http://www.w3.org/TR/html4/loose.dtd">
 3 <html>
 4   <head>
 5     <title>Titel der Seite</title>
 6   </head>
 7   <body>
 8     <h1>Überschrift der Seite</h1>
 9     <p>Ein Absatz mit Etwas Text</p>
10   </body>
11 </html>
```

- Dokumenttyp-Deklaration: Deklariert die verwendete Auszeichnungssprache, d.h. das vereinbarte „Regelwerk“
- HTML 4.01 Sprachvarianten
  - Strict: Keine Verwendung unerwünschter Elemente
  - Transitional: Verwendung aller Elemente
  - Frameset: Definition von Framesets

HTML5-Datei Aufbau

```
1 <!DOCTYPE HTML>
2 <html>
3   <head>
4     <title>Titel der Seite</title>
5   </head>
6   <body>
7   </body>
8 </html>
```

- HTML5 kennt keine verschiedenen Varianten mehr
  - Keine Frames mehr!
  - Kein DTD nötig, da HTML5 formell keine SMGL Sprache mehr ist.
  - Im nicht-HTML5 Browser: Standards Mode

HTML Tags

- Tags sind immer in spitzen Klammern eingeschlosse

```
<tagname>
```

- Tags werden (fast) immer geschlossen

```
<tagname></tagname>
```

- Zwischen den Tags kann (meist) Text und/oder weitere Tags stehen

```
<b>Text</b>
```

- Tags können (beliebig viele) Attriube haben

```
<b class="test">Text</b>
```

Text formatierung

```
1 <p>Das ist etwas Text</p>
```

Das ist etwas Text

P steht für Paragraph und bildet einen Absatz

```
1 <p>Das ist etwas Text<br>mit einem
2 Umbruch im Text</p>
```

Das ist etwas Text
mit einem Umbruch im Text

br erzeugt (weiche) Umbrüche im Text (in Word/Writer: STRG+ENTER)

- Überschriften

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

Überschrift 1
Überschrift 2
Überschrift 3
Überschrift 4
Überschrift 5
Überschrift 6

- Überschriften sind nicht nur ein visuelles Mittel!
    - Semantische Auszeichnung
    - Wichtig für maschnielle Verarbeitung, z. B. Suchmaschinenen und Sehbehinderte

Zeilenumbruch

- Zeilenumbruch erzwingen
  -
 (Standalone-Element)
- Zeilenumbruch verhindern
  -   erzeugt ein geschütztes Leerzeichen
  - An einer solchen Stelle erfolgt kein Umbruch
  - Alternative Schreibweise:
- Bedingter Zeilenumbruch
  - markiert eine Stelle an der getrennt werden darf Browserunterstützung mangelhaft, daher vermeiden

Sonderzeichen

` ` geschütztes leerzeichen
`>` >
`<` <
`&` &

`&auml;` ä
`&uuml;` ü
`&Ouml;` Ö
`&szlig;` ß

Beispiel:

```
1 <b> 5 &gt; 3 </b>
```

5 > 3

Hyperlinks

```
1 <a href="http://scooter.de" target="_blank">HYPER HYPER</a>
2 <a href="[URI]" target="[ZIEL]">[SICHTBARER TEXT]</a>
```

HYPER HYPER

- Herzstück des WWW: Ermöglichen Kontext von Informationen
- Prizip: Weiterentwicklung von Literaturverzeichnissen aus Wissenschaftlichen Publikationen
- Vordefinierte optionen für `target`
  - `_blank`: Anzeige des Verweisziels in neuem Fenster
  - `_self`: Anzeige des Verweisziels in aktuellem Fenster
  - `_parent`: Anzeige in Vaterfenster bei verschachteltem Frameset
  - `_top`: Anzeige in Hauptfenster bei verschachteltem Frameset

URI: Uniform Resource Identifier

Ein Uniform Resource Identifier (URI) (engl. „einheitlicher Bezeichner für Ressourcen“) ist ein Identifikator und besteht aus einer Zeichenfolge, die zur Identifizierung einer abstrakten oder physischen Ressource dient. URIs werden zur Bezeichnung von Ressourcen (wie Webseiten, sonstigen Dateien, Aufruf von Webservices, aber auch z. B. E-Mail-Empfängern) im Internet und dort vor allem im WWW eingesetzt.

Verweise innerhalb einer Datei

```
1 <a href="[URI]#[Ankername]">Verweistext</a>
```

- Regeln für Ankernamen (bzw. IDs):
    - sollten nur Buchstaben, Ziffern und die Sonderzeichen Unterstrich, Bindestrich, Punkt und Doppelpunkt enthalten
    - Ankernamen und ID-Attribute müssen dateiweit eindeutig sein

Beispiel: Wikipedia Artikel HTML, Abschnitt Syntax

Verweise zu Email-Adressen

```
1 <a href="mailto:[email-Adresse]">Verweistext</a>
2 <a href="mailto:karl.glatz@hs-weingarten.de">Karl</a>
3 <a href="mailto:karl.glatz@hs-weingarten.de?cc=karl.glatz@gmail.com">
4 Karl Glatz</a>
```

- Optionen bei Email-Verweisen
  - cc, bcc, subject und body
- Optionen werden als Parameter des href-Attributs übergeben, eingeleitet durch ein ?
- Problem: Spam-Bots durchsuchen internet nach mailto: Adressen!
  - Lösungen: oft wird die Adresse verunstaltet oder mit JS encodiert

Listen: Aufzählungslisten (unordered list)

```
 1 <h2>Einkaufsliste</h2>
 2 <ul>
 3     <li>Milch</li>
 4     <li>Eier</li>
 5     <ul>
 6         <li>3x Bio</li>
 7         <li>2x Freiland</li>
 8     </ul>
 9     <li>Spagetti</li>
10 </ul>
```

Einkaufsliste

- Milch
- Eier
  - 3x Bio
  - 2x Freiland
- Spagetti

ul: unorderd list | **li**: list item

Nummerierte Listen (ordered list)

```
1 <h2>Anleitung</h1>
2 <ol>
3     <li>Download</li>
4     <li>Entpacken</li>
5     <li>Starten ...</li>
6 </ol>
```

Anleitung

1. **Download**
2. **Entpacken**
3. **Starten ...**

Achtung: Verschachtelung nummerierter Listen bewirkt keine Nummerierungshierarchie (Mit CSS möglich)

Präformatierter Text

```
1 <pre>präformatierter
2     <b>Text</b> mit ein paar
3 Umbrüchen und
4         Einrückungen!</pre>
```

```
präformatierter
        Text mit ein paar
    Umbrüchen und
            Einrückungen!
```

Anzeige mit Formatierungen in dichtengleicher Schrift
Aber: HTML-Zeichen werden interpretiert

- Verwendung
  - Anzeige von Quellcode
  - vordefinierten Tabellen, etc.

Zitate und Adressen

```
1 <blockquote><p>there are only two hard problems in computing: caching,
2 concurrency and off-by-one errors</p></blockquote>
```

there are only two hard problems in computing: caching, concurrency and off-by-one errors

Zitate

- Attribut cite: URI der zitierten Quelle (ohne Visualisierung)
- `<blockquote cite="http://www.hs-weingarten.de/">` ...

Adressen

```
1 <address>Hochschule Ravensburg-Weingarten<br>
2 Doggenried Str.<br>88250 Weingarten</address>
```

Logische Textauszeichnung

- Logische Auszeichnungen im Text
  - Elemente definieren logische Bedeutung unabhängig von einer konkreten Darstellung
  - Logische Auszeichnungen sind Inline-Elemente
- Elemente zur logischen Textauszeichnung
  - em – empathisch, betont
  - strong – stark betont
  - code – Quelltext
  - samp – Beispiel
  - kbd – Benutzereingaben
  - var – Variable
  - cite – Quelle oder Autor
  - dfn – Definition
  - abbr – Abkürzung
  - acronym – Akronym
  - q - Zitat

- del - gelöschter Text
  ins - eingefügter Text
  - Attribut datetime: Zeitpunkt der Änderung
  - Attribut cite: URI als Grund für Änderung

Physische Textauszeichnung

- HTML4: Elemente definieren direkt eine gewünschte Darstellung
- HTML5: Jeweils "schwache" semantische Bedeutung zugeordnet => Styling CSS
- Elemente zur physischen Textauszeichnung
  - b – fett (bold)
  - i – kursiv (italic)
  - ~~tt - dichtengleich (teletyper)~~*
  - ~~big - größer als normal~~*
  - ~~center - zentriert~~*
  - ~~strike - durchgestrichen~~*
  - small – kleiner als normal
  - sup – hochgestellt (superior)
  - sub – tiefgestellt (subordinate)
- Sonstige Elemente: hr – trennlinie

*nicht in HTML5: W3C: HTML5 vs HTML4

Allgemeine Elemente für Textbereiche

- Allgemeines Block-Element
  - div
  - Kind-Elemente: Block-Elemente und Inline-Elemente
- Allgemeines Inline-Element
  - span
  - Kind-Elemente: Inline-Elemente
- Formatierung allgemeiner Elemente mit CSS
  - Allgemeine Elemente ermöglichen die logische Auszeichnung von Abschnitten oder Blöcken
  - Formatierung mit CSS

Allgemeine Elemente HTML5

- HTML5 biete neue allgemeine Elemente
  - article – Artikel z. B. in einem Blog
  - section – Abschnitt eines Textes
  - nav – Navigation, Menü
  - header – Kopf einer Seite
  - footer – Fuß einer Seite
  - aside – z.B. Sidebar bei einem Blog
- Alle Elemente verhalten sich wie das div, bieten jedoch die möglichkeit das HTML-Dokument besser zu strukturieren.

Grafikformate: Vektor und Pixel

- Pixel:
  - **JPEG**: Joint Photographic Experts Group
    - Verlustbehaftetes Format
    - Einsatzzweck: Fotos von Personen, Landschaften etc.
  - **PNG**: Portable Network Graphics
    - Verlustfreies Format
    - Echte Transparenz: Alpha Kanal
    - Animierte Varianten: APNG, MNG (fehlender Browser Support)
    - Einsatz: Alle Grafiken wie Verläufe, Comics, Zeichnungen, Buttons etc.
  - **GIF**: Graphics Interchange Format
    - Verlustfreies Format
    - Limitierter Farbraum
    - Einfache Transparenz (1-Bit)
    - Einsatzzweck: Animation
    - **Veraltetes Format!**

- Vektor:
  - **SVG**: Scalable Vector Graphics
    - Einsatzzweck: Grafiken wie Karten, Logos, Wappen, Zeichnungen, usw.
  - **Canvas**: Generierte Vektorgrafik
    - Einsatzzweck: Interaktive Spiele, Animationen etc.
    - Grafik muss programmiert werden in JavaScript

Einbinden von Bildern

```
1 <img src="[URI]" alt="[Alternativtext]">
```

- Referenz auf eine Graphik
  - img-Element ist Standalone-Element
- Attribut `src` bestimmt die Graphikdatei Beachten Sie die Möglichkeiten zur Referenzierung von Dateien in HTML
- Attribut `alt` definiert alternativen Text
  - IE: Tooltip
    - Tooltips normalerweise via Universalattribute `title`
  - Alle: Falls Bild nicht angezeigt werden kann; Screenreader

(Veraltete) Attribute für Bilder

- Größe der Grafik: `width`, `height`
  - Bei langsamen Verbindungen führt angabe zur schnelleren Seitendarstellung
  - Abweichende Größen zum Original sind nicht empfehlenswert:
    - Schlechte Skalierungsalgorithmen der Browser
    - Höhere Ladezeiten bei verkleinerung der Grafiken
- Ausrichtung:
  - `align`, `vspace`, `hspace`
- Für beides sollte heute CSS verwendet werden! (HTML4 Strict/HTML5)
- Image Maps: Verweissensitive Grafiken

CSS Basics

Stylesheets

Grundidee von Stylesheets

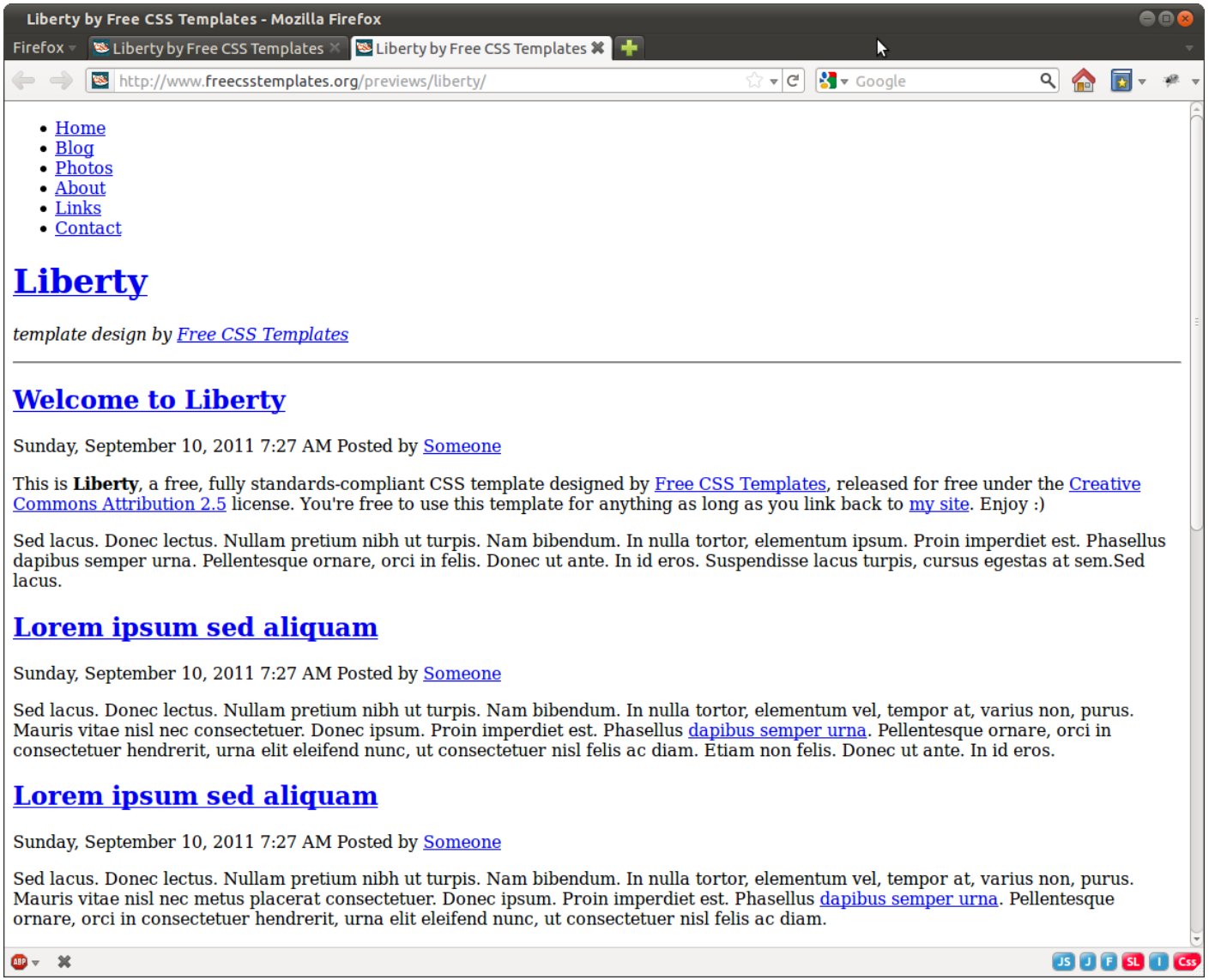
- Definition von *Formateigenschaften* für HTML-Elemente
- Strikte Trennung von Inhalt und Layout/Design
- Zentralisierung/Externalisierung von Formatdefinitionen
- Bereitstellung unterschiedlicher Layouts für ein Web-Projekt (adaptive Webseiten)

Stylesheet-Sprachen

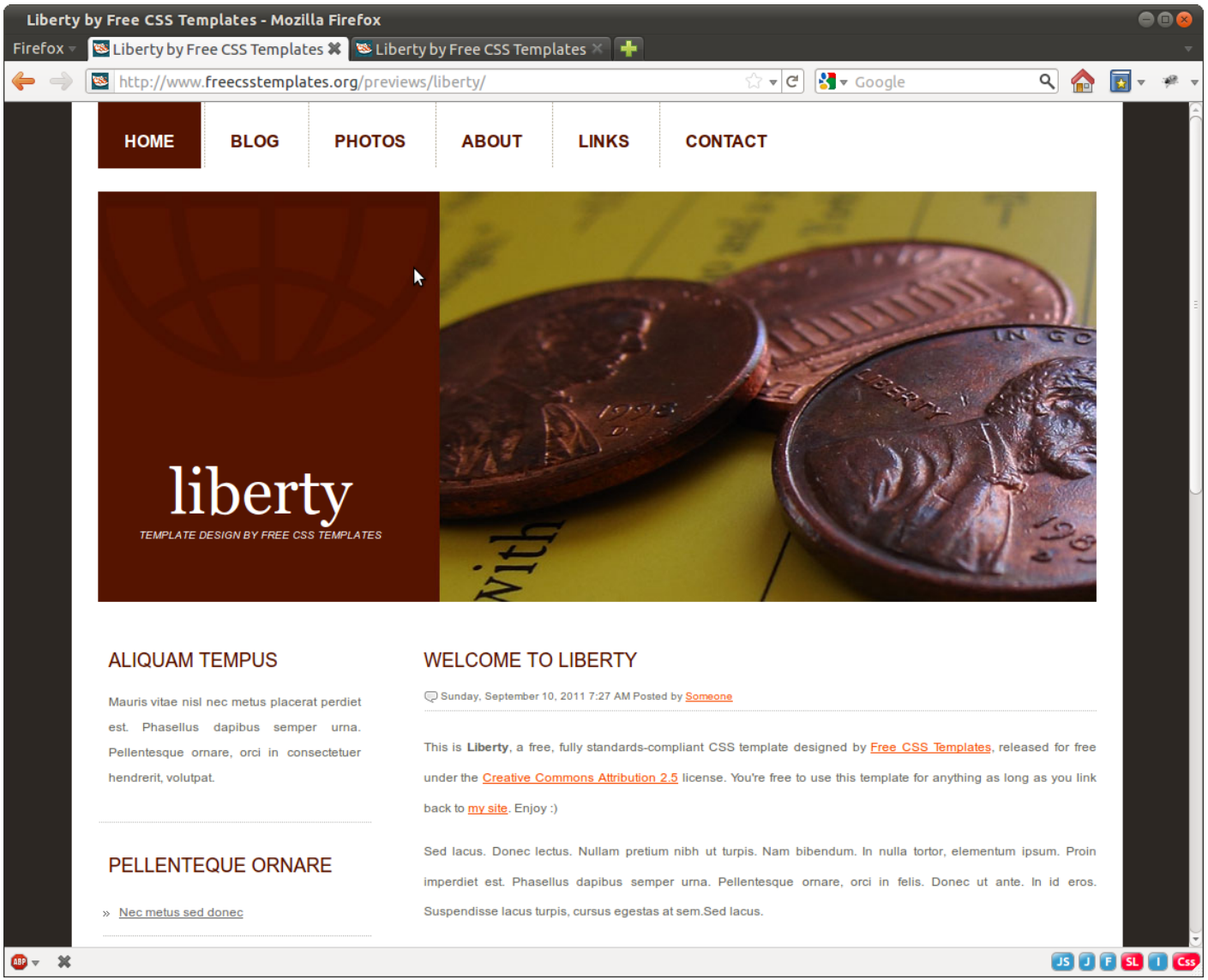
- CSS (Cascading Style Sheets): Formatierung von HTML-Dokumenten (W3C-Standard)
- XSL (Extensible Stylesheet Language): Formatierung von XML-Dokumenten (W3C-Standard)

Instacss: Interaktive CSS Dokumentation

Was kann CSS?


Liberty by Free CSS Templates - Mozilla Firefox
Firefox
Liberty by Free CSS Templates
Liberty by Free CSS Templates
http://www.freecsstemplates.org/previews/liberty/
Google
Home
Blog
Photos
About
Links
Contact
Liberty
template design by Free CSS Templates
Welcome to Liberty
Sunday, September 10, 2011 7:27 AM Posted by Someone
This is Liberty, a free, fully standards-compliant CSS template designed by Free CSS Templates, released for free under the Creative Commons Attribution 2.5 license. You're free to use this template for anything as long as you link back to my site. Enjoy :)
Sed lacus. Donec lectus. Nullam pretium nibh ut turpis. Nam bibendum. In nulla tortor, elementum ipsum. Proin imperdiet est. Phasellus dapibus semper urna. Pellentesque ornare, orci in felis. Donec ut ante. In id eros. Suspendisse lacus turpis, cursus egestas at sem.Sed lacus.
Lorem ipsum sed aliquam
Sunday, September 10, 2011 7:27 AM Posted by Someone
Sed lacus. Donec lectus. Nullam pretium nibh ut turpis. Nam bibendum. In nulla tortor, elementum vel, tempor at, varius non, purus. Mauris vitae nisl nec consectetuer. Donec ipsum. Proin imperdiet est. Phasellus dapibus semper urna. Pellentesque ornare, orci in consectetuer hendrerit, urna elit eleifend nunc, ut consectetuer nisl felis ac diam. Etiam non felis. Donec ut ante. In id eros.
Lorem ipsum sed aliquam
Sunday, September 10, 2011 7:27 AM Posted by Someone
Sed lacus. Donec lectus. Nullam pretium nibh ut turpis. Nam bibendum. In nulla tortor, elementum vel, tempor at, varius non, purus. Mauris vitae nisl nec metus placerat consectetuer. Donec ipsum. Proin imperdiet est. Phasellus dapibus semper urna. Pellentesque ornare, orci in consectetuer hendrerit, urna elit eleifend nunc, ut consectetuer nisl felis ac diam.

Was kann CSS?


Liberty by Free CSS Templates - Mozilla Firefox
Firefox
Liberty by Free CSS Templates
Liberty by Free CSS Templates
http://www.freecsstemplates.org/previews/liberty/
Google
HOME
BLOG
PHOTOS
ABOUT
LINKS
CONTACT
liberty
TEMPLATE DESIGN BY FREE CSS TEMPLATES
ALIQUAM TEMPUS
Mauris vitae nisl nec metus placerat perdiet est. Phasellus dapibus semper urna. Pellentesque ornare, orci in consectetuer hendrerit, volutpat.
PELLENTEQUE ORNARE
» Nec metus sed donec
WELCOME TO LIBERTY
Sunday, September 10, 2011 7:27 AM Posted by Someone
This is Liberty, a free, fully standards-compliant CSS template designed by Free CSS Templates, released for free under the Creative Commons Attribution 2.5 license. You're free to use this template for anything as long as you link back to my site. Enjoy :)
Sed lacus. Donec lectus. Nullam pretium nibh ut turpis. Nam bibendum. In nulla tortor, elementum ipsum. Proin imperdiet est. Phasellus dapibus semper urna. Pellentesque ornare, orci in felis. Donec ut ante. In id eros. Suspendisse lacus turpis, cursus egestas at sem.Sed lacus.

// TODO:

Image Zoom

CSS3

CSS3 Beispiele

CSS: Gefahren/Fallstricke

- Mangelnde Browserunterstützung (IE < 9.0)
- Farbdarstellung oder Schriftgrößen nicht zwingend einheitlich (Rendering Bugs)
- Umfassende Formatierungsmöglichkeiten verführen oft zu übertriebenen Layouts

Grundregeln

- Ordentliche Farbkontraste
- Keine kritischen Farbkombinationen (rot/grün)
- Normale Schriftarten, Größen, Zeilenhöhen, usw.
- Durchgängiges Layout/Design (CI/CD)

CSS-Definition: HTML Kopf

Definition zentral für eine HTML-Datei

```
1 <head>
2 <style type="text/css">
3 [selector] { [Eigenschaft]:[Wert]; ...}
4 </style>
5 </head>
```

Beispiel

```
1 <head>
2 <style type="text/css">
3     h1 { color:green; font-size:24px; }
4 </style>
5 </head>
```

CSS-Definition: Datei

Definition in separater CSS-Datei

```
1 <head>
2 <link rel="stylesheet" type="text/css" href="formatierung.css"></link>
3 </head>
```

CSS-Datei

- Textdatei mit der Endung `.css`
- Zeichenkodierung `@charset "UTF-8";`
- Kommentare: `/* ... */`
- Stylesheet-Definitionen
    - `[selector] { [Eigenschaft]:[Wert]; ...}`

CSS-Definition: HTML-Element

Definition innerhalb eines HTML-Elements

```
1 <h1 style="color:blue;">...</h1>
```

- Universalattribut `style="[Eigenschaft]:[Wert];"'
- Kein Selektor (bezieht sich nur auf aktuelles Element)
- Verschachtelte Definitionen
    - Definitionen in externen CSS-Dateien, im HTML-Kopf sowie innerhalb eines HTML-Elements können kombiniert werden
    - Die inneren Definitionen überschreiben die äußeren

`CSSDateien -> HTMLKopf -> HTMLElement`

Ausgabemedien-spezifisches Design

- Separate Stylesheet-Dateien für unterschiedliche Ausgabemedien
- Einbindung unterschiedlicher CSS-Dateien für unterschiedliche
- Ausgabemedien
  - Mögliche Medien: `all, aural, handheld, print, screen, ...`
  - Begrenzte Unterstützung durch Browser

```
1 <link rel="stylesheet" type="text/css" href="[CSS-Datei]" media="[Medium, ...]">
2 </link>
```

Ausgabemedien-spezifisches Design

Spezifische Stylesheet-Definitionen für unterschiedliche Ausgabemedien

```
1 <style type="text/css">
2 @media [medium], ...
3   { [CSS-Definitionen] }
4 </style>
```

Einbindung mittels CSS-@import

```
1 <style type="text/css">
2 @import url("[URL]") [Medium], ...;
3 </style>
```

In einer CSS-Datei müssen `@import`-Anweisungen am Anfang der Datei stehen

CSS-Regeln festlegen

- Aufbau zentraler CSS-Definitionen

  `[Selector] { [CSS-Eigenschaft] : [Wert]; }`

- **Selektor** wählt aus, für welche Elemente die folgenden Definitionen gelten
- CSS-Definitionen für HTML-Elemente

Definition der Formatierung von HTML-Elementtypen (h1, b, etc.)

```
1 h1 { color:green; font-size:24px }
2 h1, h2 { color:green; font-size:24px }
3 * { color:green; font-size:24px }
```

Universalselektor: *

CSS-Regeln: Verschachtelte Elemente

CSS-Definitionen für verschachtelte Elemente

Nachfahren-Selektoren

```
1 h1 b { font-style:italic }
```

Mindestens eine Ebene zwischen h1 und b

```
1 h1 * b { font-style:italic }
```

b direktes Kindelement von h1 Nachbar-Selektor

```
1 h1 > b { font-style:italic }
```

p unmittelbarer Nachfolger von h1

```
1 h1 + p { font-style:italic }
```

CSS-Regeln: Attribute und Klassen

Formatdefinitionen für Elemente mit bestimmten Attributen oder Attributzuweisungen

```
1 h1[align] { font-style:italic }
2 h1[align=center] { font-style:italic }
3 *[align=center] { font-style:italic }
```

CSS-Definitionen für Klassen

Formatdefinitionen für Elemente einer bestimmten Klasse

```
1 h2.neu { color:red }
2 *.neu { color:red }
3 /* oder */
4 .neu { }
```

CSS-Regeln: class + span/div

- CSS-Definitionen für Klassen mit div und span
- Die Elemente div und span ermöglichen die Zuweisung (semantischer) Klassennamen an beliebige Bereiche oder Gruppen von Elementen
- Diese Bereiche können dann mit CSS formatiert werden

CSS Code

```
1 .neu { color:red; }
2 .anmerkung { font-style:italic; font-size:10px }
3 .quelle { font-style:italic; font-size:8px }
```

HTML Code

```
1 <span class="neu">...</span>
```

CSS-Regeln: id / Pseudoklassen

CSS-Definitionen für einzelne Elemente

Formatdefinitionen für Elemente mit einer bestimmten ID

```
1 #titel { color:red; }
2 h1#titel { color:red; }
```

Pseudoelemente und Pseudoklassen

Formatdefinitionen für HTML-Bestandteile, die sich nicht durch ein HTML-Element ausdrücken lassen

```
1 a:link { } /* noch nicht besuchter Verweis */
2 a:visited { } /* besuchter Verweis  */
3 a:hover { } /* Verweis auf den die Maus zeigt  */
```

CSS-Regeln: Pseudoklassen
Before / After

CSS

```
1 h4:before { content: ">> " }
2 h4:after { content: url('http://up.frubar.net/1170/css_icon.png') }
```

HTML

```
1 <h4>Testueberschrift</h4>
```

>> Testueberschrift 

Beispiel: Linkadressen werden beim Drucken mit ausgedruckt

Verhalten von HTML-Elementen: Block

```
1 <div>
2     <h1>Überschrift</h1>
3     <p>Etwas Text</p>
4     Nur <b>Text</b> ohne p...
5 </div>
```

Überschrift

Etwas Text

Nur **Text** ohne p...

- Block-Elemente
  - Erzeugen einen eigenen Absatz im Textfluss
  - Können i.d.R. enthalten
    - Text (#PCDATA)
    - Block-Elemente **und** Inline-Elemente
    - Beispiele: <h1>, , <table>

HTML: Inline / CSS: display

- Inline-Elemente
  - Erzeugen keinen Absatz im Textfluss
  - Können i.d.R. enthalten -Text (`#PCDATA`)
    - Inline-Elemente
  - Beispiele: <br>, <i>, , ...
- Unsichtbare-Elemente
  - Beispiele: <meta>, <style>, <script>
- Steuerbar mit CSS

```
1 display: none | block | inline | ...
```

- Demo
- W3C: Default Style HTML4

Kaskadierung von Stylesheets

Unterschiedliche Stylesheets (aufst. Priorität)

- **Browser-Stylesheet** (Vorgegeben vom W3C)
  - Grundlegende Formatierung für alle HTML-Elemente
- **Benutzer-Stylesheet** (Im Browser einstellbar)
  - Spezifische Formatierungen des Benutzers
- **Autoren-Stylesheet** (Mit HTML Datei zusammen ausgeliefert)
  - Formatierungen in HTML-Seite oder importierter CSS-Datei
- **Autoren-Stylesheet mit** `!important`
  - Als wichtig gekennzeichnete Formatierungen im Autoren-Stylesheet
- **Benutzer-Stylesheet mit** `!important`
  - Als wichtig gekennzeichnete Formatierungen im Autoren-Stylesheet

Kaskadierung von Stylesheets

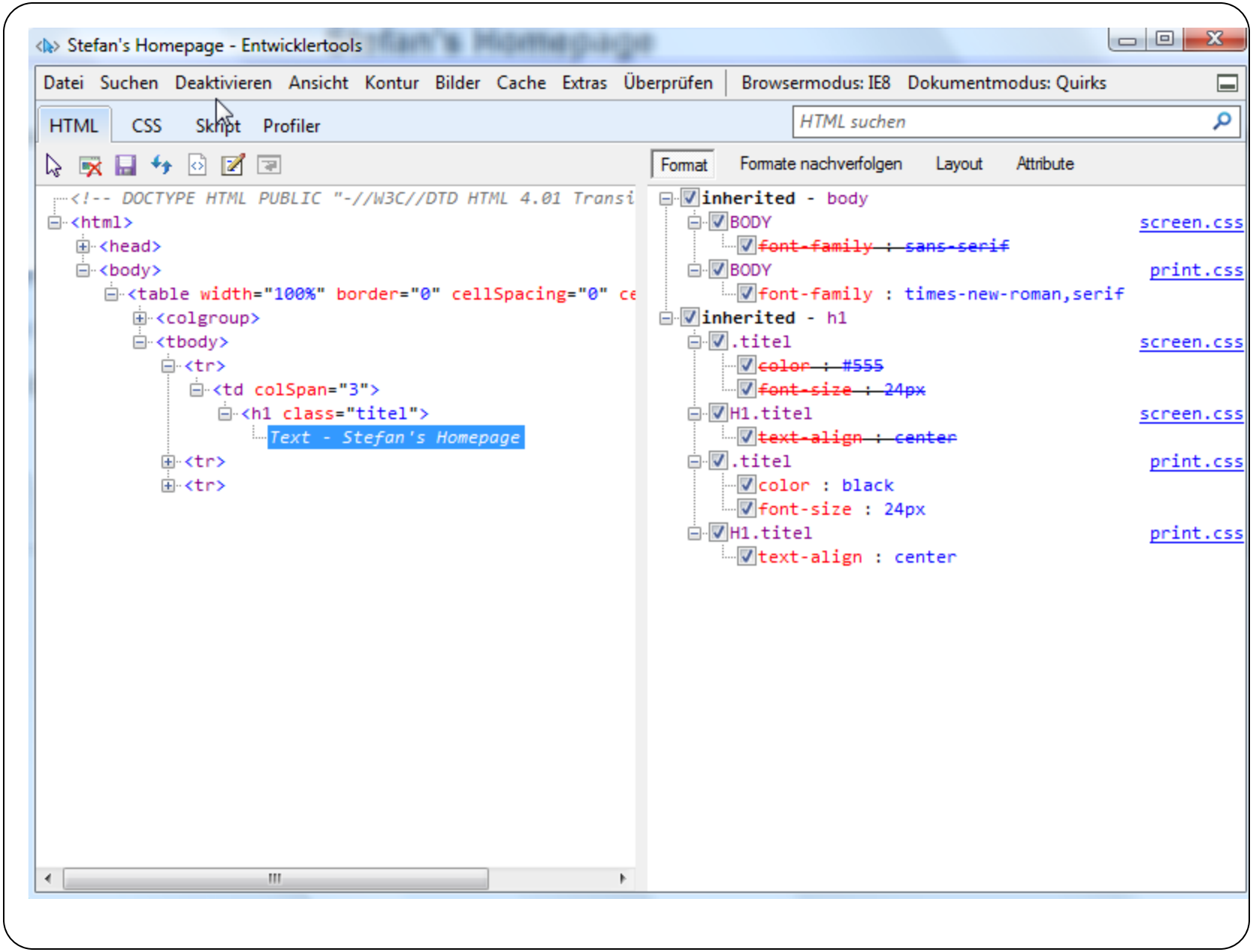
- Spezifität der Selektoren
  - Grundidee: *spezifischere* CSS-Definitionen *haben Vorrang* vor unspezifischeren
  - Beispiel: `h2.myheading` überschreibt `h2`
- Sortierung nach Spezifität
  - Nach dem *Ort* der Definition: CSS-Definitionen per `style`-Attribute haben Vorrang
  - Nach der *Zahl* der selektieren *ID-Attribute*
  - Nach der *Zahl* der selektieren sonstigen Attribute (z.B. Klassen oder Pseudoklassen)
  - Nach der *Zahl* der selektierten *Elementnamen* und *Pseudoelemente*
  - *Reihenfolge* des Vorkommens (spätere Selektoren überschreiben vorherige)

Analyse von Styles: FF

Analyse der Darstellung von CSS-formatierten Seiten im Firefox mit dem Add-on Firebug

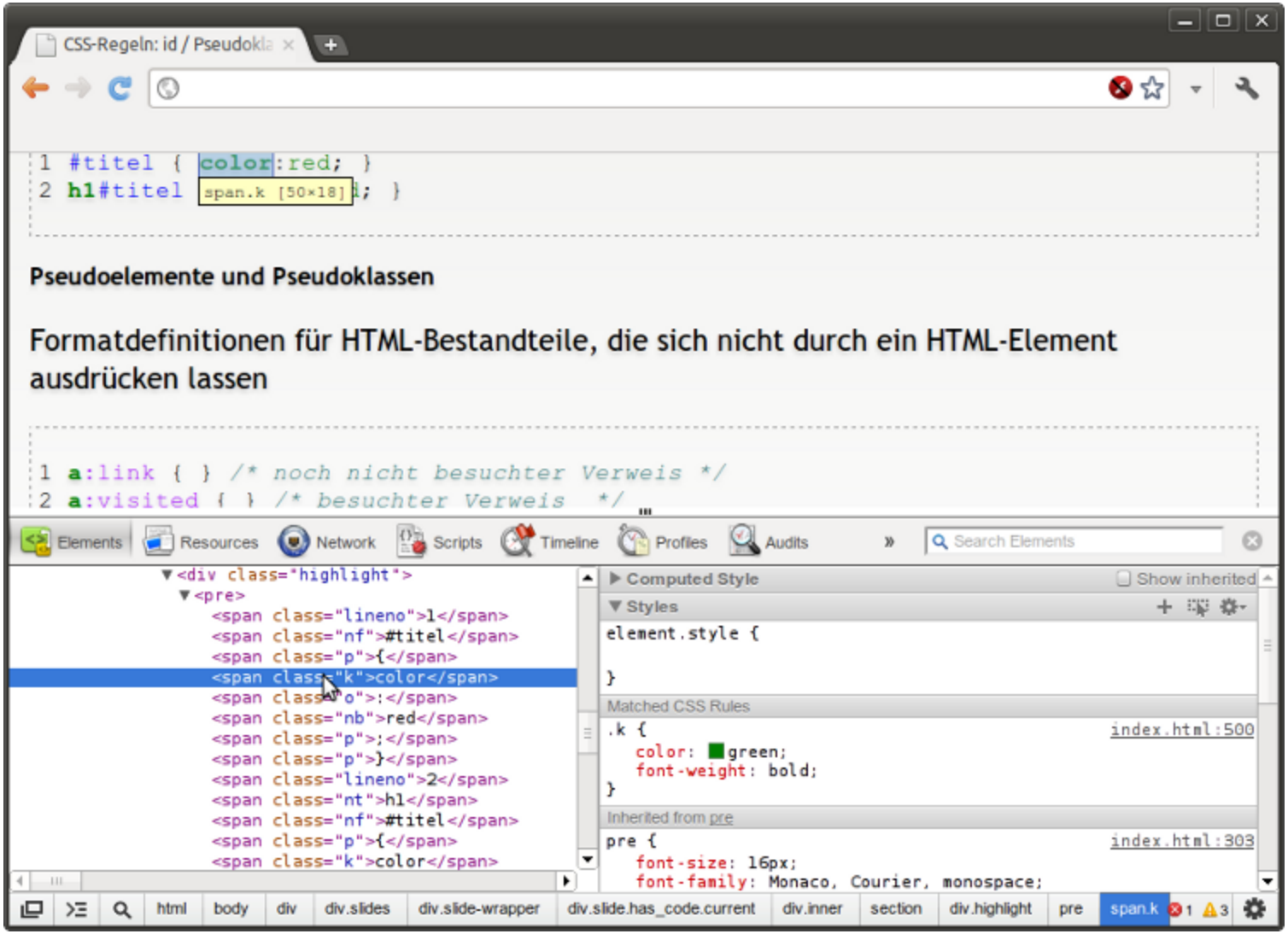
Analyse von Styles: IE

Analyse der Darstellung von CSS-formatierten Seiten im IE mit den eingebauten Entwicklertools (F12)


Stefan's Homepage - Entwicklertools
Datei Suchen Deaktivieren Ansicht Kontur Bilder Cache Extras Überprüfen Browsermodus: IE8 Dokumentmodus: Quirks
HTML CSS Skript Profiler
HTML suchen
Format Formate nachverfolgen Layout Attribute
<!-- DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi
<html>
<head>
<body>
<table width="100%" border="0" cellSpacing="0" ce
<colgroup>
<tbody>
<tr>
<td colSpan="3">
<h1 class="titel">
Text - Stefan's Homepage
<tr>
<tr>
inherited - body
BODY screen.css
font-family : sans-serif
BODY print.css
font-family : times-new-roman,serif
inherited - h1
.titel screen.css
color : #555
font-size : 24px
H1.titel screen.css
text-align : center
.titel print.css
color : black
font-size : 24px
H1.titel print.css
text-align : center

Analyse von Styles: Chrome

Analyse der Darstellung von CSS-formatierten Seiten im Chrome mit den eingebauten Entwicklertools (F12)


CSS-Regeln: id / Pseudokla
1 #titel { color:red; }
2 h1#titel span.k [50×18]; }
Pseudoelemente und Pseudoklassen
Formatdefinitionen für HTML-Bestandteile, die sich nicht durch ein HTML-Element ausdrücken lassen
1 a:link { } /* noch nicht besuchter Verweis */
2 a:visited { } /* besuchter Verweis */
Elements
Resources
Network
Scripts
Timeline
Profiles
Audits
Search Elements

<pre>
1
#titel
{
color
:
red
;
}
2
h1
#titel
{
color
Computed Style
Show inherited
Styles
element.style {
}
Matched CSS Rules
.k {
color: green;
font-weight: bold;
}
index.html:500
Inherited from pre
pre {
font-size: 16px;
font-family: Monaco, Courier, monospace;
index.html:303
html
body
div
div.slides
div.slide-wrapper
div.slide.has_code.current
div.inner
section
div.highlight
pre
span.k

CSS-Eigenschaften: Schriftformat

Schriftart

```
1 font-family:[Schriftart/Schriftfamilie],[Schriftart/Schriftfamilie],...;
```

- Schriftarten werden in angegebener Reihenfolge verwendet
- Schriftarten mit Leerzeichen werden in Hochkomma gesetzt
- Schriftfamilien: serif, sans-serif, cursive, fantasy, monospace (oft als letzte Angabe)

Beispiel:

```
1 font-family:Verdana,sans-serif
```

Schriftstil

```
1 font-style:[italic|oblique|normal]
```

CSS-Eigenschaften: Schriftformat #2

Schriftvariante (1) / Schriftgröße (2)

```
1 font-variant:[small-caps|normal]
2 font-size:[Schriftgroesse]
```

- Numerische Angabe
  - Absolut: `px` (pixel), `pt` (point), `mm`, `cm`
  - Relativ: `em` (Elementgröße), `ex` (Höhe x), `%`
- Schlüsselwörter
  - Absolut: `xx/x-small, small, medium, large, x/xx-large`
  - Relativ: `smaller, larger`

Absolute numerische Angaben für Drucklayouts
Relative Angaben oder Schlüsselwörter für Screen-Layouts

CSS-Eigenschaften: Schriftformat #3

Schriftgewicht

```
1 font-weight:[bold|bolder|lighter|normal|100..900]
```

Schrift gesamt

```
1 font:[font-style
2 font-variant
3 font-weight
4 font-size/line-height
5 font-family]
```

Reihenfolge ist einzuhalten `font-size` und `font-family` sind obligatorisch `font` setzt zunächst alle Angaben auf Standardwerte

```
1 font:italic bold 1.2em/1.4em Verdana
```

CSS: Farben

```
1 p {
2     color: red;
3     background-color: blue;
4 }
```

- Vordergrundfarbe: `color`
- Hintergrundfarbe: `background-color`
- Farbangaben:
  - `#RGB` bzw. `#RRGGBB` (Hexadezimal)
  - `rgb(0-255, 0-255, 0-255)`
  - Keywords: `black, white, red, blue, green, lightblue`
- Beispiel ROT `#FF0000` oder `rgb(255,0,0)` oder `red`

W3C Schools: CSS Colors

Web Tool: HTML Color Codes

CSS: Hintergrund

Hintergrundbilder

```
1     p { background: red url(images/bg.png) fixed repeat-y; }
```

Werte: `background-color, background-image, background-attachment, background-repeat, background-position`

Bild-Adressen mit `url('pfad/zum/bild/Datei.png')`

CSS: Absatzformatierung

Texteinrückung

```
1 text-indent:[Einrueckung f. erste Zeile]
2 text-indent:1em;
```

Zeilenhöhe

```
1 line-height:[Zeilenhoehe]
```

Vertikale Ausrichtung

```
1 vertical-align:[top|middle|bottom|
2 baseline|sub|super|text-top|text-bottom]
```

CSS: Absatzformatierung

Horizontale Ausrichtung

```
1 text-align:[left|center|right|justify]
```

Zeilenumbruch

```
1 white-space:[normal|pre|nowrap]
```

Listenformatierung

Darstellungstyp

```
1 list-style-type:[ decimal | lower-roman | upper-roman | lower-alpha |
2 upper-alpha | disc | circle | square | none ]
```

Listeneinrückung

```
1 list-style-position:[inside|outside]
```

Bullet-Graphik

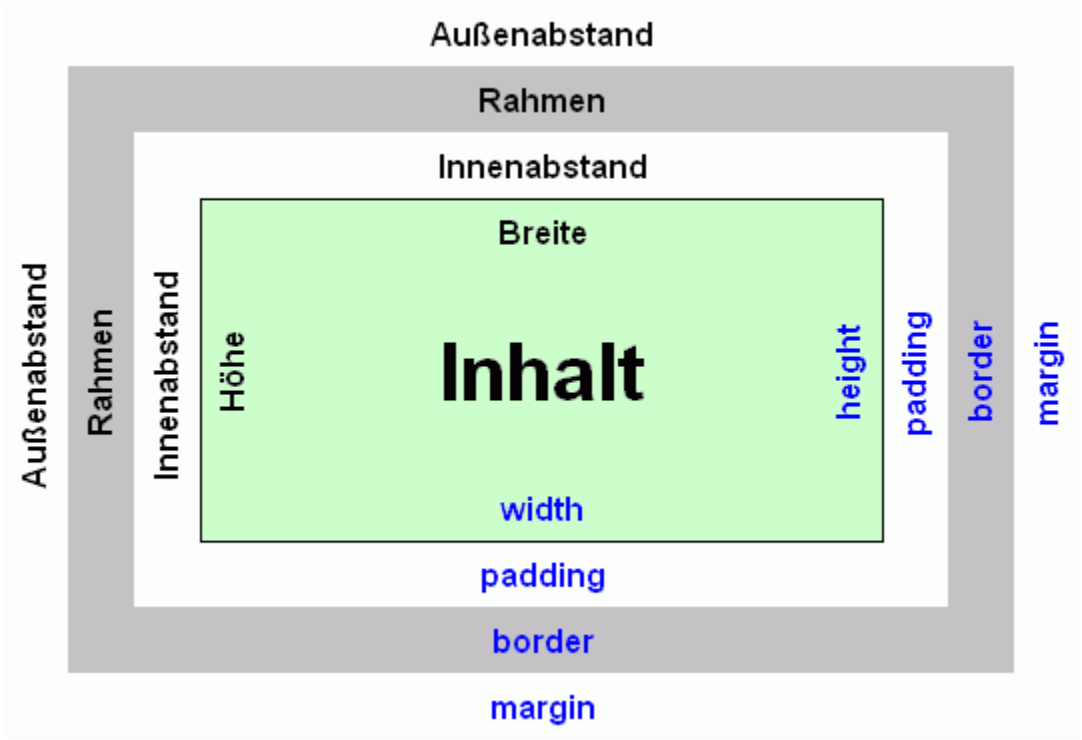
```
1 list-style-image:url([URL|none])
```

Listenformatierung gesamt

```
1 list-style:[type position image]
```

Layout

Das Box Modell



Quelle: SelfHTML

Aussen-Abstand: Margin

Aussen-Abstand oben/rechts/unten/links

```
margin-[top|right|bottom|left]: [Abstandsangabe|auto|inherit]
```

Abstände rechts/links aneinandergrenzender Elemente werden addiert Abstände oben/unten aneinandergrenzender Elemente werden nicht addiert sondern der größere Abstand gilt

`auto` oben/unten bewirkt Abstand `0` `auto` links bewirkt rechtsbündige Position `auto` rechts und links bewirkt Zentrierung

```
margin-top:10px; margin-bottom:10px
```

Aussen-Abstand: Margin

```
1 margin:10px
2 margin:10px 5px 10px 5px
```

Außenrand/Abstand oben/rechts/unten/links

`margin:[Angabe 1] ... [Angabe 4]`

- Eine Angabe für alle 4 Ränder
- Angabe für oben/unten und rechts/links
- Angabe für oben, rechts/links und unten
- Angabe für oben, rechts, unten und links

Innenabstand: Padding

Innenabstand oben/rechts/unten/links

```
1 padding-[top|right|bottom|left]:[Abstand]
2 padding-top:10px; padding-bottom:10px
3 padding:[Abstand 1] ... [Abstand 4]
```

- Eine Angabe für alle 4 Ränder
- Angabe für oben/unten und rechts/links
- Angabe für oben, rechts/links und unten
- Angabe für oben, rechts, unten und links

Rahmen: Border

Rahmentyp

```
1 border-[top|right|bottom|left]-style: [none|hidden|dotted|dashed|solid|double| groove|ridge
2 border-style:[Angabe 1] ... [Angabe 4]
3 border-style:solid
4 border-top-style:solid
```

Rahmenbreite

```
1 border-[top|right|bottom|left]-width: [Rahmenbreite|thin|medium|thick]
2 border-width:5px; border-top-width:5px
```

Rahmen: Border

Rahmenfarbe

```
1 border-[top|right|bottom|left]-color: [Rahmenfarbe]
2 border-color:blue; border-top-color:blue
```

Rahmen gesamt

```
1 border-[top|right|bottom|left]: [border-width border-style border-color]
```

Reihenfolge der Angaben ist egal

```
1 border:5px solid red
2 border-top:5px solid red
```

Rahmen: Border

Outline (Kontur)

Rahmen für nicht rechteckige Bereiche (Textfluss) Unterschied zu `border`: Konturen werdenüber dem Element gezeichnet, daher keinen eigenen Platz Outline-Rahmen können nur vollständig geschlossen gezeichnet werden

```
1 outline-width: [Rahmenbreite]
2 outline-style: [Rahmentyp]
3 outline-color: [Rahmenfarbe]
4 outline: [width style color]
```

Hintergrundfarben und -bilder

Hintergrundfarbe

```
1 background-color:[Farbe]
```

Hintergrundbild

```
1 background-image:url([URI])
```

Wiederholung Hintergrundbild

```
1 background-repeat:[repeat|repeat-x|repeat-y|no-repeat]
```

Wasserzeichen-Effekt

```
1 background-attachment:[scroll|fixed]
```

Hintergrundfarben und -bilder

Hintergrundposition

```
1 background-position:[left|center|right] [top|center|bottom]
2 background-position: left top;
```

Hintergrund gesamt

```
1 background:[background-color    background-image    background-repeat
2    background-attachment    background-position]
3
4 background:url(wi-team-1.jpg) no-repeat fixed center;
```

Größe von Elementen

Angabe von Breite und Höhe

`width:[Breite|auto]` Breite des Elements
`min-width:[Breite]` Mindestbreite des Elements
`max-width:[Breite]` Maximalbreite des Elements

`height:[Höhe|auto]` Höhe des Elements
`min-height:[Breite]` Mindesthöhe des Elements
`max-height:[Breite]` Maximalhöhe des Elements

Position von Elementen

Art der Positionierung

```
1 position: [static|relative|absolute|fixed]
```

`static` normaler Elementfluss (Normalposition)
`relative` relativ zur Normalposition (Verschieb.)
`absolute` absolute Position (zum nächsthöheren *nicht static* positionierten Element)
`fixed` absolute Position (zum Browserfenster) bleibt beim Scrollen stehen

Mit `absolute` oder `fixed` positionierte Elemente sind nicht Teil des Elementflusses

Angabe der Position

`top:[Abstand|auto]` Position von oben
`left:[Abstand|auto]` Position von links
`bottom:[Abstand|auto]` Position von unten
`right:[Abstand|auto]` Position von rechts

Anzeige von Elementen

Anzeige von übergroßem Inhalt

```
1 overflow:[visible|hidden|scroll|auto]
```

`visible` Inhalt ragt aus Element heraus
`hidden` Inhalt wird abgeschnitten
`scroll` abschneiden + scrollen
`auto` Web-Browser entscheidet

Textfluss

```
1 float:[left|right|none]
```

Element steht links/rechts und wird von nachfolgendem Text umflossen

```
1 clear:[left|right|both|none]
```

Textumfluss wird abgebrochen und nachfolgender Text beginnt unterhalb des Elements

Anzeige von Elementen

Ebene bei Überlappung

```
1 z-index:[Ebene];
2 z-index:3;
```

Art der Anzeige

```
1 display:[block|inline|inline-block|list-item|run-in|none]
```

`block` Anzeige als Block
`inline` Anzeige im Textfluss
`inline-block` Block im Textfluss
`list-item` Block mit Bullet
`run-in` kontextabhängig
`none` Keine Anzeige

Anzeige von Elementen

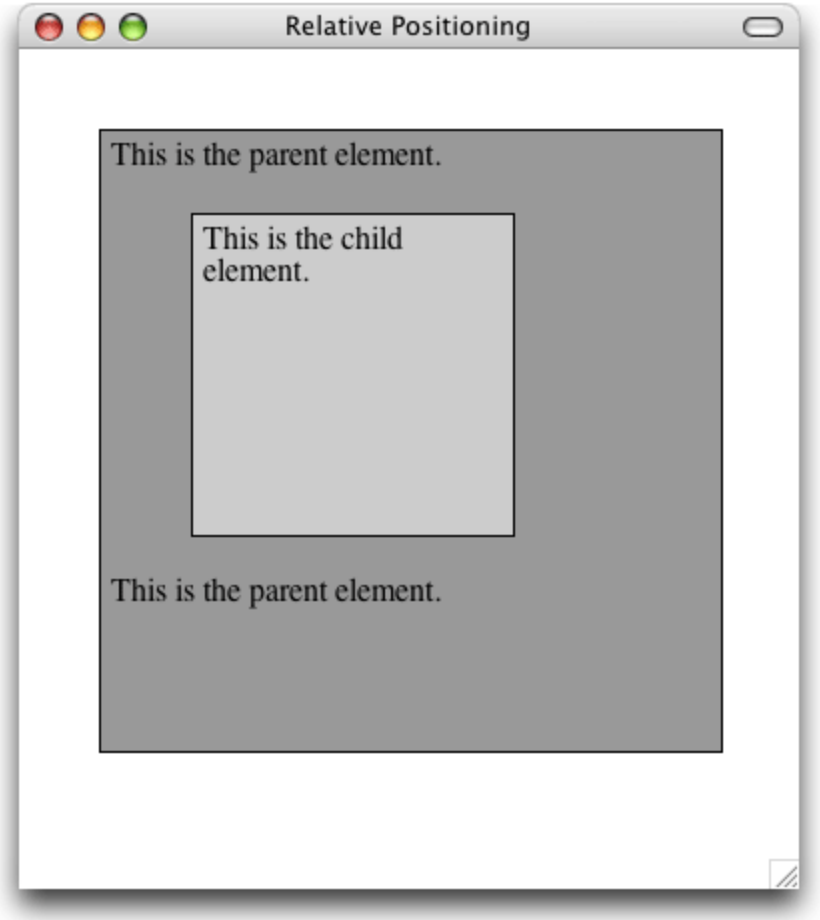
```
1 visibility:[visible|hidden|collapse]
```

`visible` sichtbar
`hidden` versteckt (Platzhalter)
`collapse` Tabelle gibt Platz frei

Unterschied zwischen `visibility: hidden` und `display: none`: Berechnung der Position anderer Elemente!

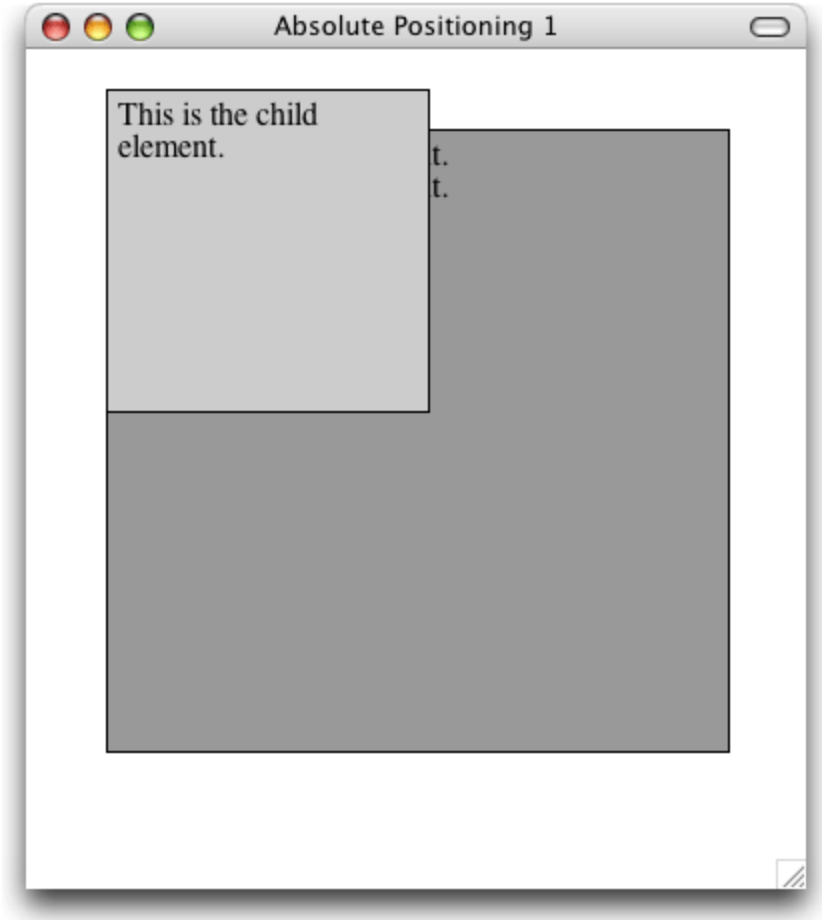
Positionierung: Relativ

```
 1 <html>
 2 <head>
 3 <title>Relative Positioning</title>
 4 <style>
 5 .parent {
 6 background: #999;
 7 }
 8 .child {
 9 position: relative;
10 top: 20px;
11 left: 40px;
12 background: #ccc;
13 }
14 </style>
15 </head>
16 <body>
17 <div class="parent">
18 <p>This is the parent element.</p>
19 <div class="child">
20 <p>This is the child element.</p>
21 </div>
22 <p>This is the parent element.</p>
23 </div>
24 </body>
25 </html>
```


Relative Positioning
This is the parent element.
This is the child element.
This is the parent element.

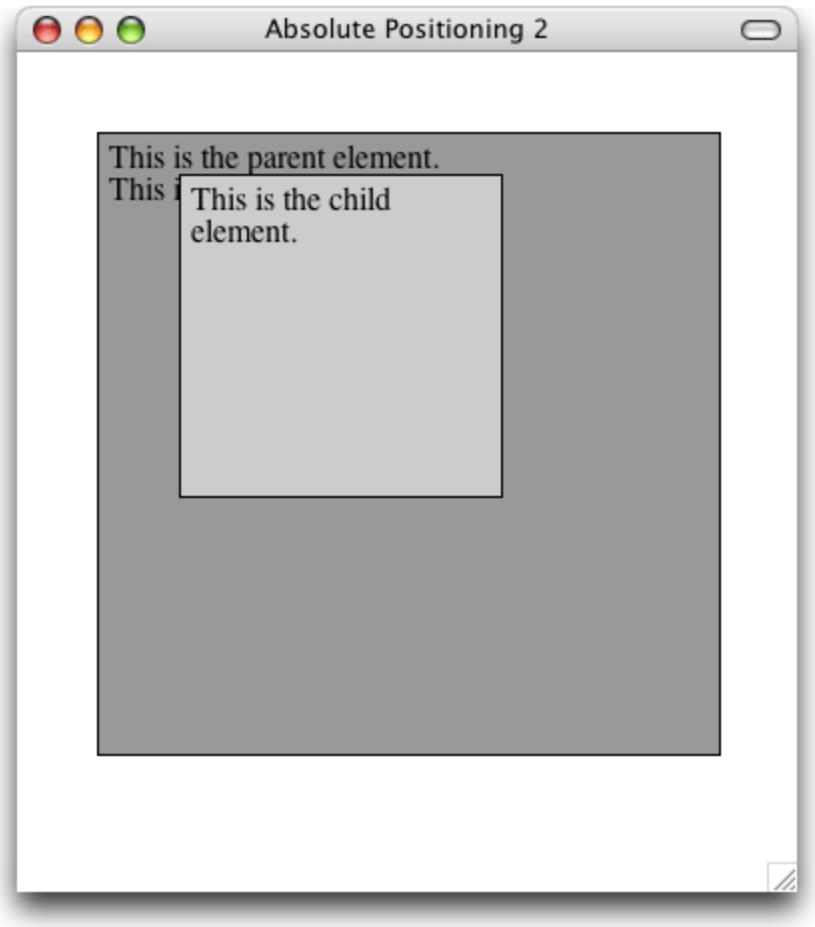
Positionierung: Absolut

```
 1 <html>
 2 <head>
 3 <title>Absolute Positioning 1</title>
 4 <style>
 5 .parent {
 6 background: #999;
 7 }
 8 .child {
 9 position: absolute;
10 top: 20px;
11 left: 40px;
12 background: #ccc;
13 }
14 </style>
15 </head>
16 <body>
17 <div class="parent">
18 <p>This is the parent element.</p>
19 <div class="child">
20 <p>This is the child element.</p>
21 </div>
22 <p>This is the parent element.</p>
23 </div>
24 </body>
25 </html>
```


Absolute Positioning 1
This is the child element.
t.
t.

Positionierung: Absolut #2

```
 1 <html>
 2 <head>
 3 <title>Absolute Positioning 2</title>
 4 <style>
 5 .parent {
 6 position: relative;
 7 background: #999;
 8 }
 9 .child {
10 position: absolute;
11 top: 20px;
12 left: 40px;
13 background: #ccc;
14 }
15 </style>
16 </head>
17 <body>
18 <div class="parent">
19 <p>This is the parent element.</p>
20 <div class="child">
21 <p>This is the child element.</p>
22 </div>
23 <p>This is the parent element.</p>
24 </div>
25 </body>
26 </html>
```


Absolute Positioning 2
This is the parent element.
This i
This is the child element.

Layout Methoden

Veraltete Methoden des Seitenlayouts

- Frames
  - „Deprecated“: kein gültiges HTML in der Variante Strict
- Tabellenbasiertes Seitenlayout
  - Verletzt die Trennung zwischen Inhalt und Layout

Beide Varianten ermöglichen keine Anpassung an Smartphones/Tablets!!

CSS-basiertes Seitenlayout

➔ Flexible Definition der Anordnung der Elemente einer Seite
➔ Erleichtert die Pflege der Seiten
➔ Ermöglicht alternative Layouts
➔ Reduziert den Umfang des HTML-Codes

Layout: Elemente nebeneinander
Simples 2 Spalten Layout

Block-Elemente werden mittels **width**, **float** und **margin** als unabhängige Boxen untereinander oder nebeneinander angeordnet.

Beispiel: Zwei Elemente nebeneinander:

```
1 <div>Dies ist eine Box mit Text</div>
2 <div>Eine zweite Box mit Text</div>
3 <hr>
4 <div style="float: left;  border: 2px solid;">Dies ist eine Box mit Text</div>
5 <div style="float: left;  border:2px solid;">Eine zweite Box mit Text</div>
```

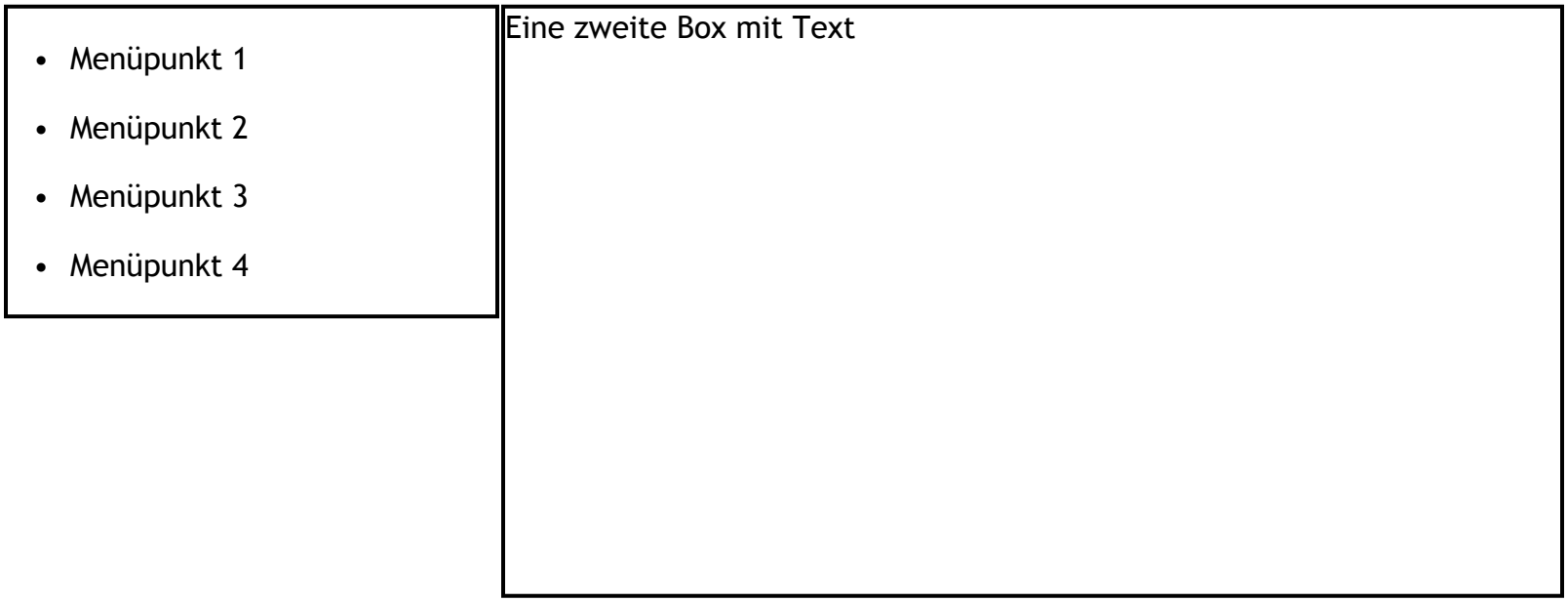
Dies ist eine Box mit Text
Eine zweite Box mit Text

Dies ist eine Box mit Text Eine zweite Box mit Text

Layout: 2 Spalten

Problem: Eine Box feste größe, andere Box rest des Bildschirms

```
1 <div style="float: left;  width: 250px; border: 2px solid;">
2 Dies ist eine Box mit Text</div>
3 <div style="margin-left: 252px; border:2px solid; min-height: 300px;">
4 Eine zweite Box mit Text</div>
```



Achtung: Margin wirkt nur bei nicht-float!

3 Spalten Layout

```
1 <h1>Überschrift</h1>
2 <ul id="navigation">...</ul>
3 <div id="info">...</div>
4 <div id="inhalt">...</div>
5 <div id="footer">Fußzeile</div>
```

[4] (Reihenfolge von außen nach innen)

CSS:

```
 1 #navigation {
 2     float:left;
 3     width:150px;
 4 }
 5 #info {
 6     float:right;
 7     width:150px;
 8 }
 9 #inhalt {
10     margin:0px 160px; /* link und rechts 160px */
11 }
12 #footer {
13     clear:both;
14 }
```

Dreispaltiges Layout

Überschrift

- Menüpunkt 1
- Menüpunkt 2
- Menüpunkt 3
- Menüpunkt 4

Etwas Inhalt
lala

Infobox

Fußzeile

Probleme

- Keine gleichmäßige Höhe
- Feste Größenangaben

Layout-Templates

- 9 Timeless 3 Column Layout Techniques
- The Perfect 3 Column Liquid Layout
  No CSS hacks. SEO friendly. No Images. No JavaScript. Cross-browser & iPhone compatible.

Layout Frameworks:

- 960 Grid System
- Blueprint
- YAML: Yet Another Multicolumn Layout Ein (X)HTML/CSS Framework (ACHTUNG: Lizenz)
- 5 Popular CSS Frameworks - Getting Started

Was Sie über CSS-Frameworks wissen sollten!

ACHTUNG: Die Frameworks dürfen im Projekt nicht verwendet werden! Simple Templates hingegen schon, ihr solltet allerdings verstehen was diese bewirken! (Wird u. U. abgeprüft in der mündlichen Prüfung)!

Navigationsleisten

```
1 <ul id="navigation">
2     <li>Menüpunkt
3         <ul><li>Unterpunkt</li></ul>
4     </li>
5 </ul>
```

CSS:

```
 1 #navigation li {
 2     list-style:none;
 3 }
 4
 5 #navigation li ul {
 6 margin:0px 0px 0px 20px;
 7 font-size:0.8em;
 8 display:none;  /* Unterpunkte werden zunächst nicht angezeigt */
 9 }
10
11 #navigation li:hover ul {
12 display:block; /* Unterpunkte werden angezeigt */
13 }
```

Navigationsleisten: Horizontal

Horizontale Navigationsleisten

```
 1 #navigation li {
 2     float:left;          /* bewirkt horizontale Anordnung */
 3     position: relative; /*ermöglicht Positionierung der Unterpunkte*/
 4 }
 5
 6 #navigation +div {
 7     clear:left; /* Aufhebung des Umflusses am Ende der Navigationsleiste*/
 8 }
 9
10 #navigation li ul {
11     position:absolute;  /* Positionierung der Unterpunkte relativ zum Oberpunkt */
12     top:10px;
13     left:0px;
14 }
```

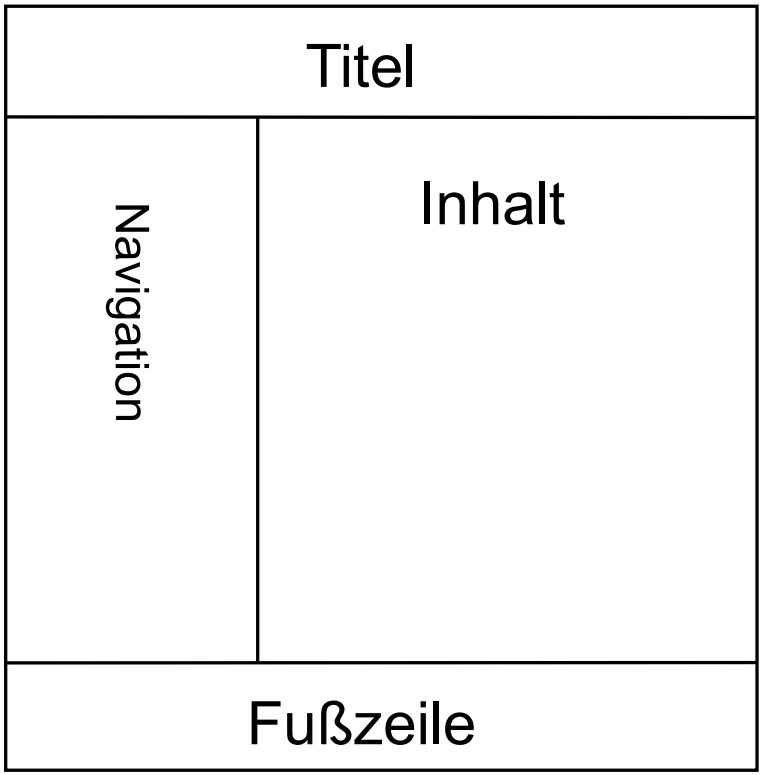
HTML "Advanced"

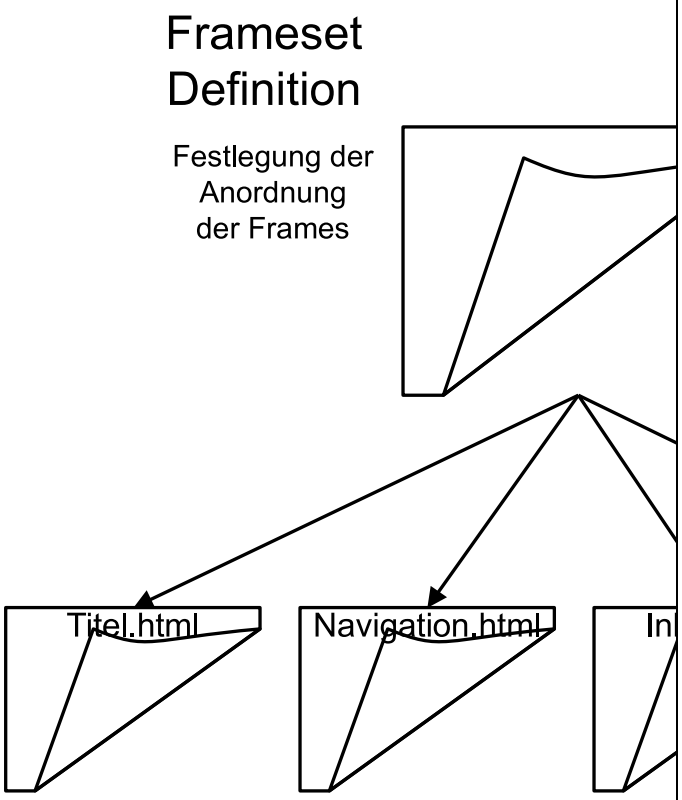
Inhalt

- (I)-Frames
- Tabellen
- Formulare

Frames

- **Veraltete Technik** die für das Layout eingesetzt wurde.
- Mehrere HTML Seiten gleichzeitig darstellen





Frames: Nachteile

- Problem der Unterstützung nicht framefähiger Browser (Heute: vernachlässigbar!)
- Problem bei der Verlinkung auf untergeordnete Seiten (deep links, Lesezeichen)
- Verletzung des URI Prinzips (Jede Information besitzt eine unabhängige URI)
- Frames sehr problematisch als Mittel zum Seitenlayout
    - Limitierte Gesaltungsmöglichkeiten
- **Kein modernes Layout setzt auf Frames!**

I-Frames

Anzeigen fremder Quellen (HTML-Dateien)

```
1 <iframe src="[URI]" width="[Breite]" height="[Höhe]"
2 name="[Name]" align="[left|right]"  scrolling="[auto|yes|no]"
3 frameborder="[1|0]" longdesc="[URI]">
4 [Alternativtext]</iframe>
```

Der Inhalt eingebetteter Frames kann durch Verweise geändert werden

```
1 <a href="[URI]" target="[Name des iframes]">
```

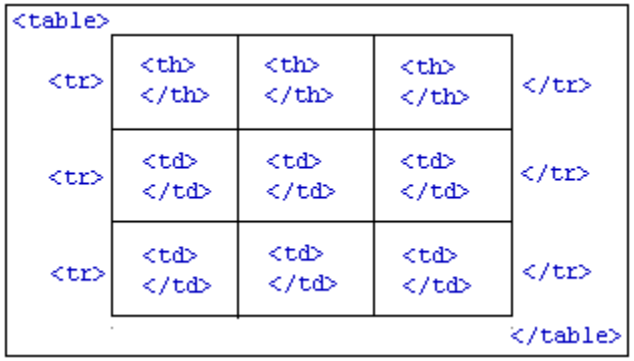
Einsatzzwecke

- Facbeook integration
- Youtube Videos (HTML5)
- Verschiedenste Widgets von Drittanbietern

HTML Tabellen

- Tabellarische Darstellung von Daten
- *KEIN* Grundgestaltungsmittel für Seiten-Layouts
- Anordnung von Informationen ist sowohl ein Aspekt der Strukturierung als auch der Gestaltung
    - Verstößt eigentlich gegen Trennung von Inhalt/Struktur und Gestaltung, aus praktischen Gründen akzeptiert

HTML-Elemente zum Aufbau einer Tabelle



HTML Tabelle

Berlin	Hamburg	München
Miljöh	Kiez	Bierdampf
Buletten	Frikadellen	Fleischpflanzerl

```
 1 <table border="1">
 2   <tr>
 3     <th>Berlin</th>
 4     <th>Hamburg</th>
 5     <th>M&uuml;nchen</th>
 6   </tr>
 7   <tr>
 8     <td>Milj&ouml;h</td>
 9     <td>Kiez</td>
10     <td>Bierdampf</td>
11   </tr>
12   <tr>
13     <td>Buletten</td>
14     <td>Frikadellen</td>
15     <td>Fleischpflanzerl</td>
16   </tr>
17 </table>
```

Tabellen Spalten definieren

Spalten vordefinieren

```
 1 <table>
 2 <colgroup>
 3 <col width="100">
 4 <col width="200">
 5 <col width="50" span="3">
 6 </colgroup>
 7 <tr> ...
 8 <table>
 9 <colgroup width="50%" span="2"></colgroup>
10 <tr> ...
```

- colgroup kann mehrfach vorkommen
- Angaben in col überschreiben Angaben in colgroup

Logische Bereiche einer Tabelle

Kopf, Körper und Fuß einer Tabelle

```
1 <table>
2 <thead> <th> ... </th> ... </thead>
3 <tfoot> <td> ... </td> ... </tfoot>
4 <tbody> <td> ... </td> ... </tbody>
5 </table>
```

- Kopf und Fuß können maximal einmal vorkommen und werden üblicherweise bei umgebrochenen Tabellen auf jeder Seite wiederholt
- Der Körper kann beliebig oft vorkommen Logische Bereiche einer Tabelle können anschließend mittels CSS formatiert werden

HTML-Elemente einer Tabelle

- Kindelemente `table`: `col, colgroup, thead, tfoot, tbody, tr`
- Attribute `table` (Besser: CSS!)
  - border: Rahmenbreite (border="0")
  - width: Breite der Tabelle (width="100%") [Auch in tr für Spaltenbreite]
- Kindelemente th/td: Block- oder Inline-Elemente
- Tabellen können beliebig verschachtelt werden
- Tabellen sollten **nicht** zur Strukturierung ganzer Webseiten verwendet werden

CSS Eigenschaften von Tabellen

```
1 .staetetab {
2     border: 1px solid #000;
3     border-collapse: collapse;
4     border-spacing: 10px; /* Keine Wirkung mit border-collapse: collapse */
5 }
6 .staetetab td { border: 1px solid green; }
7 .staetetab tr:first-child { height: 100px; }
8 .staetetab tr th:first-child {  width: 150px; }
9 .staetetab td, .staetetab th {  padding: 10px;  }
```

Berlin	Hamburg	München
Miljöh	Kiez	Bierdampf
Buletten	Frikadellen	Fleischpflanzerl

The Styleworks: Tabellenformatierung mit CSS

Zellen verbinden

- Zellen einer Zeile verbinden
  - `colspan="[Spaltenzahl]"` in erster Zelle angeben Zahl der Zellen der Zeile reduziert sich um Spaltenzahl-1
- Zellen einer Spalte verbinden
  - `rowspan="[Zeilenzahl]"` in erster Zelle angeben Zahl der Zellen der folgenden Zeilenzahl-1 Zeilen reduziert sich jeweils um eins
- Zellen spalten- und zeilenweise verbinden: Kombination der Attribute `colspan` und `rowspan`
- Zellen verbingen = Strukturbeinflussung, daher in HTML; nicht in CSS

Zellen verbinden #2

```
 1 <table border="1">
 2   <tr>
 3     <td rowspan="2">Basiszahlen</td>
 4     <td>200 &euro;</td>
 5   </tr>
 6   <tr>
 7     <td>400 &euro;</td>
 8   </tr>
 9   <tr>
10     <td colspan="2">Sonstige Informationen</td>
11   </tr>
12 </table>
```

<table>
<tr><td rowspan="2">Basiszahlen</td><td>200 €</td></tr>
<tr><td>400 €</td></tr>
<tr><td colspan="2">Sonstige Informationen</td></tr>
</table>

Weitere Attribute für Tabellen

- Überschrift:
  `Überschrift`
  - `<caption align="[left, right]">...</caption>`
  - `<caption align="bottom">...</caption>`
- Zusammenfassung des Inhalts `<table summary="[Zusammenfassung]">`
- Kurzbeschreibung für Zellen `<th/td abbr="[Kurzbeschreibung]">`
- Referenzen auf Kopfzellen
  - Identifikation der Kopfzellen mittels `<th id="[ID]">`
  - Bezugnahme auf Kopfzellen mittels `<td headers="[ID]">`
- Kategoriennamen für Tabellenzellen `<th/td axis="[Kategorie]">`
- Verwendung von Zusatzinformationen
  - Ausgabe auf nicht-visuellen Medien (z.B. Sprachausgabe)
  - Formatierung mittels CSS

Wiederholung der Kopfinformationen in Datenzellen

Formulare in HTML

- Formulare ermöglichen
  - Die strukturierte Erfassung von Daten durch den Nutzer
    - Eingabefelder/Textfelder, Auswahlelemente, etc.
  - Das Senden der Daten an die Web-Applikation bzw. den Betreiber
  - Die Interaktion des Nutzers mit der Web-Applikation
- Formularbereiche
  - `<form>...</form>`
  - Innerhalb des form-Elements
    - Eingabefelder des Formulars
    - beliebige Block-Elemente zur Anordnung und Beschriftung der Eingabefelder

Verarbeitung von Formularen

Verarbeitung der Formularinhalte

- Attribut action definiert ein Programm zur Weiterverarbeitung der Daten des Formulars
  - `<form action="[URI]">`
  - Verarbeitung mittels der CGI-Schnittstelle (z.B. Perl / PHP Skript)
- Attribut method definiert Übertragungsmethode
  - `method="get"`
    - Formulardaten werden beim Aufruf als Parameter angehängt http://www.homepage.de/cgi-bin/test.cgi?Name=Mustermann&Vorname=Max
  - `method="post"`
    - Übergabe der Daten an das verarbeitende Programm über die Standardeingabe (z.B. bei umfangreichen Formulardaten)
  - W3C-Empfehlung: get für temporäre Daten, post für persistente Daten

Kodierung der Formulardaten

- Attribut accept-charset definiert die verwendete Kodierung für Formulardaten
- Kodierung sollte übereinstimmen mit Angaben im HTTP-Header (Serverkonfiguration) und Meta-Tag der HTML-Datei

Beispiel

```
1 <form action="mailto:..." enctype="text/plain"
2 method="post" accept-charset="UTF-8">
```

Anmerkung: `action="mailto:mail@domain.de"` ruft auf dem Rechner das E-Mail Programm auf und ermöglicht so das vorherige ausfüllen von E-Mails auf der Webseite. Nicht mehr gebräuchlich, die E-Mails werden heute vom Webserver gesendet (z. B. PHP mail()).

Formularfelder: Eingabe

Einzeilige Eingabefelder

```
1 <input type="text" name="[Name]" size="[Size]"
2 maxlength="[maxlength]" value="[Vorbelegung]">
3 <!-- Beispiel Beruf: -->
4 <input name="Beruf" type="text" size="20" maxlength="30">
```

Informatiker

Eingabefelder für Passwörter

```
1 <input type="password" name="[Name]" size="[Size]"
2 maxlength="[maxlength]" value="[Vorbelegung]">
```

Formularfelder: Eingabe

Mehrzeilige Eingabefelder

```
1 <textarea name="[Name]" rows="[Zeilen]"
2 cols="[Spalten]">[Vorbelegung]</textarea>
```

Das ist ein
wenig Text.

- Text als Vorbelegung innerhalb des textarea-Elements wird so dargestellt wie er eingegeben wurde HTML-Sonderzeichen sind zu maskieren (`<` `>` `&`)
  - Es sind keine weiteren HTML-Elemente erlaubt
- Zeilenumbruch verhindern: `wrap="off"`
- Readonly Eingabefelder: `readonly`

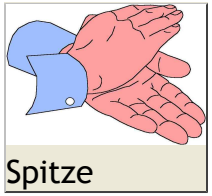
Formularfelder: Buttons

Buttons

```
1 <input type="button" name="[Name]" value="[Beschriftung]" [Event-Handler]>
2 <button type="button" name="[Name]" value="[Beschriftung]" [Event-Handler]>
3 [beliebiger HTML-Code als Beschriftung (keine Verweiselemente)]</button>
```

Beispiel: Button mit Graphik

```
1 <button type="button" name="Button" value="hier klicken" onclick="alert('Bravo')">
2 <img src="http://blog.radiobob.de/wp-content/uploads/2008/08/applaus.jpg"
3 width="100" height="70" alt="Applaus"><br>Spitze</button>
```


Spitze

Formularfelder: Absenden

Buttons zum Absenden oder Abbrechen

```
1 <input type="[submit/reset]" name="[Name]" value="[Beschriftung]">
```

Beispiel

```
1 <input type="submit" name="Button" value="Absenden">
2 <input type="image" src="wi-team-1.jpg" width="100"
3 height="70" name="Button" alt="Absenden">
```

Neben dem value werden bei `button` auch die Koordinaten des Klicks innerhalb des Bildes übertragen

```
1 <button type="submit" name="Button" value="Absenden">Absenden</button>
```

Formularfelder: Auswahliste

Auswahllisten

```
1 <select name="[Name]" size="[Zahl Einträge]">
2 <option>[Eintrag]</option>
3 ...
4 </select>
```

`size="1"` ergibt eine Dropdown-Liste Beispiel

```
1 <select name="Auswahlliste" size="1">
2     <option>Eintrag 1</option>
3     <option>Eintrag 2</option>
4     <option>Eintrag 3</option>
5     <option>Eintrag 4</option>
6 </select>
```

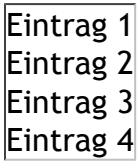
Eintrag 1

Formularfelder: Auswahliste

- Attribut multiple ermöglicht eine Mehrfachauswahl
- Attribut selected im option-Element ermöglicht Vorauswahl
- Attribut value="[Absendewert]" ermöglicht die Definition eines speziellen Absendewerts

Beispiel

```
1 <select name="Auswahlliste" size="3" multiple>
2     <option value="1" selected>Eintrag 1</option>
3     <option value="2">Eintrag 2</option>
4     <option value="3">Eintrag 3</option>
5     <option value="4">Eintrag 4</option>
6 </select>
```



Formulare: Verschachtelte Auswahllisten

`optgroup` erlaubt die Gruppierung von Elementen einer Auswahlliste

Beispiel

```
 1 <select name="Verschachtelte_Auswahlliste" size="1">
 2     <optgroup label="Gruppe 1">
 3         <option label="Eintrag_1">Eintrag 1</option>
 4         <option label="Eintrag_2">Eintrag 2</option>
 5     </optgroup>
 6     <optgroup label="Gruppe 2">
 7         <option label="Eintrag_3">Eintrag 3</option>
 8         <option label="Eintrag_4">Eintrag 4</option>
 9     </optgroup>
10 </select>
```

Eintrag 1
Eintrag 3

Formularfelder: Radio Buttons

```
1 <input type="radio" name="[Name]" value="[Wert]">
```

- Radio-Buttons mit dem gleichen Namen gehören zu einer Gruppe
- Attribut `checked` ermöglicht die Vorauswahl eines Buttons

Beispiel

```
1 <input type="radio" name="Geschlecht" value="m" checked>männlich<br>
2 <input type="radio" name="Geschlecht" value="w">weiblich<br>
```


männlich

weiblich

Formularfelder: Checkboxen

```
1 <input type="checkbox" name="[Name]"value="[Wert]">
```

Checkboxen mit dem gleichen Namen gehören zu einer Gruppe Attribut checked ermöglicht die Vorauswahl einer Checkbox

```
1 <input type="checkbox" name="Hobbies" value="lesen" checked> lesen<br>
2 <input type="checkbox" name="Hobbies" value="kochen"> kochen<br>
3 <input type="checkbox" name="Hobbies" value="studieren"> studieren<br>
```

☑ lesen
☐ kochen
☐ studieren

Datei-Upload / Versteckte Felder

Datei-Upload

```
1 <input type="file" name="[Name]" size="[Feldlänge]">
```

Funktioniert nur in Verbindung mit method="post" und enctype="multipart/form-data"

Versteckte Felder

```
1 <input type="hidden" name="[Name]">
```

- Können zur Speicherung und Übertragung von Daten an die Server-Applikation verwendet werden
- Der Inhalt versteckter Felder kann z.B. mittels Java-Skript verändert werden

Formular: Elementgruppen

Elementgruppen

```
1 <fieldset>
2 <legend>[Überschrift]</legend>
3 [Felder der Gruppe]
4 </fieldset>
```

Beispiel

```
1 <fieldset>
2 <legend>Überschrift</legend>
3 <input type="checkbox" name="Checkbox"
4 value="Wert_1"> Wert 1<br>
5 <input type="checkbox" name="Checkbox"
6 value="Wert_2"> Wert 2<br>
7 <input type="checkbox" name="Checkbox"
8 value="Wert_3"> Wert 3<br>
9 </fieldset>
```

Überschrift ☐ Wert 1
☐ Wert 2
☐ Wert 3

Formular: Labels

Label für Felder

```
1 <label for="[ID]">[Beschriftung]</label>
2 <input id="[ID]" ... >
```

Beschriftung wird einem Feld eindeutig über dessen ID zugeordnet

```
1 <label for="Vorname">Vorname: </label>
2 <input id="Vorname" name="Vorname" type="text"
3 size="20" maxlength="30"><br>
4 <label for="Nachname">Nachname: </label>
5 <input id="Nachname" name="Nachname"
6 type="text" size="20" maxlength="30">
```

Vorname:
Nachname:

Formular Eigenschaften

- Tabulator-Reihenfolge
  - Das Attribut tabindex in den Fomularelementen `input`, `textarea`, `select` oder `button` erlaubt die Definition der Tabulator-Reihenfolge
  - Die Tabulator-Reihenfolge sollte *konsistent* für alle auswählbaren Elemente einer HTML-Seite definiert werden (Verweise, Graphiken, Objekte, Formularelemente)
- Tastaturkürzel
  - Das Attribute `accesskey` in den Formularelementen `input, textarea, label, legend` und `button` definiert ein Zeichen zum Auswählen eines Formularelementes
- Disabled Elemente
  - Das Attribut `disabled` ermöglich das disablen von Elementen (erlaubt in `input, textarea, select, option, optgoup, button`)

Formular Demo

Persönliches Vorname:
Nachname:
Strasse:
Hausnummer: 150
Hobbys ☑ Radfahren
☐ Schwimmen
☐ Lesen
Geschlecht ◉ männlich
○ weiblich
Absenden Reset

Formular: Demo Source

```
 1 <form action="http://lookthis.de/test.php" method="POST" target="_blank">
 2     <input type="hidden" name="unsichtbare_id" value="3" />
 3     <fieldset>
 4     <legend>Persönliches</legend>
 5     <label for="Vorname">Vorname: </label>
 6     <input id="Vorname" name="Vorname" type="text" size="20" maxlength="30"><br>
 7     <label for="Nachname" tabindex="1">Nachname: </label>
 8     <input id="Nachname" name="Nachname" type="text" size="20" maxlength="30"><br>
 9     <label for="Strasse">Strasse: </label>
10     <input id="Strasse" name="Strasse" type="text" size="20" maxlength="30"><br>
11     <label for="Hausnummer">Hausnummer: </label>
12     <input id="Hausnummer" value="150" name="Hausnummer" type="text" size="20" maxlength="3
13     </fieldset>
14     <fieldset>
15     <legend>Hobbys</legend>
16     <input type="checkbox" name="Hobbys[]" value="radfahren" checked> Radfahren<br>
17     <input type="checkbox" name="Hobbys[]" value="schwimmen"> Schwimmen<br>
18     <input type="checkbox" name="Hobbys[]" value="lesen"> Lesen<br>
19     </fieldset>
20     <fieldset>
21     <legend>Geschlecht</legend>
22     <input type="radio" name="Geschlecht" value="m" checked>männlich<br>
23     <input type="radio" name="Geschlecht" value="w">weiblich<br>
24     </fieldset>
25     <input type="submit" name="submit" value="Absenden">
26     <input type="reset" name="reset" value="Reset">
27 </form>
```

HTML und XHTML

- SGML- Standard Generalized Markup Language
  - Standardisierte (Meta)-Auszeichnungssprache
  - Erlaubt die Definition spezifischer Auszeichnungssprachen mittels
  - DTDs (Document Type Definitions)
    - Eine DTD legt fest, aus welchen Elementen/Tags und Attributen eine konkrete Auszeichnunssprache besteht
    - HTML ist eine SGML-basierte Auszeichnungssprache
- XML – Extensible Markup Language
  - Vereinfachte Variante von SGML
  - Auch XML erlaubt die Definition konkreter Auszeichnungssprachen mittels DTDs (oder Schemata)

HTML vs XHTML

- Definition von HTML mittels XML (anstelle von SGML)
- XHTML ist eine XML-basierte Auszeichnungssprache mit dem gleichen
- Sprachumfang wie HTML
    - XHTML 1.0 entspricht HTML 4.0
    - HTML5 kann sowohl als XML oder als SMGL definiert werden (Umgangssprachlich: HTML5 / XHTML5)

Vorteile von XHTML

- Kompatibilität zu anderen XML-basierten Auszeichnungssprachen (e.g. WML, XSL, etc.)
- Ermöglicht Zugriff auf XHTML-Dokumente via DOM (Document Object Model)

Verarbeitung von XHTML im Browser

- Zwei Arten der Verarbeitung von XHTML
  - Als XML (mittels des XML-Parsers des Browsers)
  - Als HTML (mittels des HTML-Parsers des Browsers)
- MIME-Typen für XHTML
  - `text/html` (Verarbeitung als HTML-Datei)
  - `text/xml` oder application/xml (... als normale XML-Datei)
  - `application/xhtml+xml` (... als XHTML-Datei)
  - Wichtig: XML-Parser sind strenger als HTML-Parser
  - Ältere Browser unterstützen nur text/html
- Dateinamen
  - `.htm` oder `.html` (Verarbeitung als HTML-Datei)
  - `.xhtml` (... als XHTML-Datei)

Unterschied: Dokumenttyp-Deklaration

XHTML 1.0

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml">
5 <!-- usw. -->
```

XHTML5 (XHTML Variante von HTML5)

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <!-- usw. -->
```

Quelle: HTML5 Handbuch

Unterschied: Dokumenttyp-Deklaration #2

- XML-Deklaration `<?xml version="1.0" encoding="UTF-8" ?>`
- XML-Deklaration ist optional und sollte bei Verarbeitung als HTML-Datei weggelassen werden
- Dokumenttyp-Deklaration (s.o.) *
- Namensraumangabe
  - `<html xmlns="http://www.w3.org/1999/xhtml">`
- Groß-/Kleinschreibung
  - XML unterscheidet zwischen Groß- und Kleinschreibung
  - In XHTML werden alle Element- oder Attributnamen klein geschrieben

[*] Mit XHTML5 ändert sich die Lage insofern, als XHTML5 gar keine Dokumenttyp-Deklaration mehr fordert — aus dem einfachen Grund, weil für XHTML5 gar keine DTD existiert. XHTML5 ist lediglich eine XML-gerechte Variante von HTML5. Und HTML5 basiert nicht mehr auf Markup-Dokumenttyp-Deklarationen, sondern auf DOM-Datenstrukturen. Quelle: HTML5 Handbuch

Besonderheiten von XHTML

- Standalone-Elemente
  - Standalone-Elemente werden mit “ /” als inhaltsleer gekennzeichnet (z.B. `<br />`)
- Elemente ohne Inhalt sollten nicht in obiger Notation sondern mit einem schließenden Tag notiert werden (z.B. `<p></p>`)
- Elemente mit optionalem Abschluss-Tag
  - Notieren sie alle Elemente die Inhalt haben können mit einem schließenden Tag
- Alleinstehende Attribute
  - In XHTML muss allen Attributen ein Wert zugewiesen werden
  - Alleinstehende Attribute (z.B. `checked`, `readonly`) werden in der Form notiert:
  - `[attribut]="[attribut]"`

Besonderheiten von XHTML #2

- Verweise zu Ankern
  - Anker in XHTML: Universalattribute `id` statt `name` (beide Notationen möglich)
  - Beispiel: `<a id="[Anker]" name="[Anker]">...</a>`
- Script- und Stylesheet-Bereiche
  - Inhalte von `script` und `style` sind in XHTML PCDATA (statt CDATA) (Sonderzeichen müssen maskiert werden) Script- oder Stylesheet-Bereiche in CDATA einschließen

`script` Tag in XHMTL

```
1 <script type="text/javascript">
2 /* <![CDATA[ */ ... /* ]]> */
3 </script>
```

HTML & Semantik

Links

TODO: Lang, accesskey etc. [???]

HTML-Kopfdaten

Meta-Angaben: Optional seit HTML 4.0 (zukünftig mittels RDF)

Angaben für Web-Browser und Suchmaschinen

```
1 <meta name="author" content="Leonie Mueller">
```

Angaben für Web-Server (heute Auswertung meist erst durch Browser)

```
1 <meta http-equiv="content-type" content="text/html; charset=UTF-8">
```

Meta-Angaben für heutige Suchmaschinen von geringer Bedeutung (aufgrund von Missbrauch)

```
1 <meta name="description" content="Text erscheint als Beschreibung bei Suchmaschinentreffern"
2 <meta name="keywords" content="Leonie ; Homepage">
3 <meta name="date" content="2009-03-10T10:15:34+01:00">
```

HTML-Kopfdaten

Mehrsprachige Meta-Angaben

```
1 <meta name="description" lang="de" content="Text">
2 <meta name="description" lang="en" content="Text">
```

Angabe eines externen Profils für Meta-Angaben

```
1 <head profile="http://dublincore.org/documents/dcq-html/">
```

Angabe eines Schemas für eine Meta-Angabe

```
1 <meta name="Typ" scheme="MIME-Type" content="text/html">
```

HTML-Kopfdaten

Suchmaschinenzugriff erlauben/verbieten

```
1 <meta name="robots" content="index, noindex, nofollow, all">
```

Zeichenkodierung

```
1 <meta http-equiv="content-type" content="text/html; charset=UTF-8">
```

Script- und Stylesheet-Sprache

```
1 <meta http-equiv="Content-Script-Type" content="text/javascript">
2 <meta http-equiv="Content-Style-Type" content="text/css">
```

HTML-Kopfdaten

Gültigkeit von Zwischenspeichern (Browser-Cache, Proxy-Cache)

```
1 <meta http-equiv="expires" content="0">
2 <meta http-equiv="expires" content="Mon, 23 Mar 2009 12:00:00 GMT">
```

Automatische Weiterleitung

```
1 <meta http-equiv="refresh" content="3; URL=http://www.hs-weingarten.de/">
```

- Achtung:
  - Weiterleitungen werden nicht von allen Browsern unterstützt
  - Zeit 0 führt zum Festhängen auf der Seite beim „Zurück“-Button

Logische Beziehungen

Einbettung einer Hypertextdatei in ihren Kontext

Darstellung der Abhängigkeiten zu anderen Dateien

```
1 <link rel="next" title="Next" href="zweite_seite.html">
```

Anerkannte Link-Arten

`contents, chapter, section, subsection, index,glossary, appendix, search, author, copyright,next, prev, first, last, up, top, help, bookmark,alternate stylesheet, alternate stylesheet, shortcut icon`

Google stellt einige dieser Links in den Ergebnissen dar!

Dublin Core Metadata Initiative

TODO: DC

HTML5 Metadata

TODO: HTML5

Microformats Metadata

TODO: Microformats

Java Script

Java Script

- Standardisiert als ECMAScript (ECMA 262)
- Skriptsprache für das Web (ähnliche Synax wie C, Java)
- Einführung durch Netscape 1995
- Dynamisches genereiren von Inhalten möglich
- Moderne Webanwendungen (z. B. GMail) nutzen massiv JavaScript
- Der Name JavaScript stammt aus den damals populären Java-Applets und hat nichts mit der Java Virtual Machine zu tun!
- Ziele und Eigenschaften
    - Überprüfen von Eingaben
    - Manipulation von HTML-Dokumenten (dynamisches HTML)
    - Objektorientiert, dynamisch typisierend, klassenlos

Beispiel

```
 1 <html>
 2 <head>
 3 <script type="text/javascript">
 4     function myfunction(txt)
 5     {
 6         alert(txt);
 7     }
 8 </script>
 9 </head>
10 <body>
11
12 <form>
13     <input type="button" onclick="myfunction('Hello')" value="Call function">
14 </form>
15
16 </body>
17 </html>
```

Quelle: W3C Schools Demo

Einbettung von JS

In separaten Dateien

Datei quadrat.js:

```
1 function Quadrat() {
2 var Ergebnis = document.Formular.Eingabe.value *
3 document.Formular.Eingabe.value; alert("Das Quadrat von " +
4 document.Formular.Eingabe.value + " = " + Ergebnis);
5 }
```

HTML-Datei:

```
1 <head>
2 <title>JavaScript-Test</title>
3 <script src="quadrat.js" type="text/javascript">
4 </script>
5 </head>
```

JS: Allgemeine Regeln

Anweisungen:

```
1 Quadrat = Zahl * Zahl;
```

Anweisungsblöcke:

```
1 function SageQuadrat (x) {
2     var Ergebnis = x * x; /* Kommentar: x mal x */
3     alert(Ergebnis); }
```

- Selbstvergebene Namen (z. B. Variablen)
  - Beginnen mit Buchstaben
  - Einziges Sonderzeichen: _
  - Unterscheidung Groß-/Kleinschreibung
  - Reservierte Wörter beachten

Kommentare: `//, /* ... */`

Aufrufen von Javascript-Funktionen

Event-Handler

Beispiel Button-Event

```
1 <form name="Formular" action="">
2 <input type="text" name="Eingabe" size="3">
3 <input type="button" value="Quadrat errechnen"
4 onclick="Quadrat()"> </form>
```

Beispiel Body-Event

```
1 <body onload="ZeitAnzeigen()">
```

Variablen

- Definition von Variablen: var
  - Globale Variable:
    - var-Defintion außerhalb einer Funktion
    - Zuweisung ohne var-Definition
  - Lokale Variable:
    - var-Definition innerhalb einer Funktion
  - Namensregeln beachten

Beispiel

```
1 var Hinweis = "Gleich werden Quadratzahlen ausgegeben";
```

JS: Objekte

Vordefinierte Objekte
Beispiel: Aktuelle Uhrzeit

```
1 var Jetzt = new Date();
2 var Stunden = Jetzt.getHours();
3 var Minuten = Jetzt.getMinutes();
4 document.write("<h2>Guten Tag!<\/h2><b>Es ist jetzt " + Stunden + " Uhr und " + Minuten
5 + " Minuten<\/b>");
```

Eigene Objekte über Funktionen
Beispiel: Hintergrund wechseln

```
1 function Farbe (R, G, B) { this.R = R; this.G = G; this.B = B; this.hex = "#";}
2 function HintergrundWechseln () {
3 var Hintergrund = new Farbe("E0", "FF", "E0");
4 document.bgColor = Hintergrund.hex + Hintergrund.R + Hintergrund.G + Hintergrund.B; }
```

Verfügbarkeitsprüfung: `if (!document.images) {...} else {...}`

JS: Funktionen

Schlüsselwort und Parameter

`function (param1, param2)`

Anweisungsblock

`{}` Optionaler Rückgabewert

`return`

```
1 function BruttoBetrag (Netto, Prozente)
2 {
3     var Ergebnis = Netto * (1 + (Prozente / 100));
4     return Ergebnis;
5 }
```

JS: Operatoren

Zuweisungsoperator: `=`
Vergleichsoperatoren: `<, <=, >, >=, ==, !=`
Berechnungsoperatoren: `+, -, *, /, %`
Logische Operatoren: `&&, ||, !`
`if (!name)`: Wahr bei `void, null, 0, "", undefined`
Zeichenkettenverknüpfung: `+`

```
1 var strasse = "Hauptstrasse" + " 42";
```

JS: Bedingte-Anweisungen

Wenn-Dann-Abfrage

```
1 if (Bedingung) then {...} else {...}
```

Einfaches Entweder-Oder

```
1 var erg = (Bedingung) ? "richtig" : "falsch"
```

Fallunterscheidung

```
1 var Eingabe = window.prompt("Zahl eingeben: 1 - 4");
2 switch (Eingabe) {
3     case "1": alert("Sie sind sehr bescheiden"); break;
4     case "2": alert("Sie sind ein aufrichtiger Zweibeiner"); break;
5     case "3": alert("Sie haben ein Dreirad gewonnen"); break;
6     case "4": alert("Gehen Sie auf allen Vieren und werden Sie bescheidener"); break;
7     default: alert("Sie bleiben leider dumm"); break;
8 }
```

JS: Schleifen

```
1 /* While-Schleife */
2 while (Bedingung){ ...}
3 /* For-Schleife */
4 for (var i = startwert; i <= endwert; i++) {...}
5 /* Do-While-Schleife */
6 do { ... } while(Bedingung)
```

- Kontrollfluss innerhalb der Schleife:
  - `break;` => Schleife wird verlassen
  - `continue;` => Nächster Schleifendurchlauf

Event-Handler

- Bindeglied zwischen HTML und JavaScript
- Definition über Attribute von HTML-Tags
- W3C legt fest, welche Handler für welche HTML-Elemente existieren
- Populäre Event-Handler
  - `onclick` (bei Anklicken z.B. von Buttons)
  - `onload` (nach Laden einer Datei, z.B. im Body-Element um sicherzustellen, dass Dokument vor Ausführung von Code geladen)

Beispielanbindung

```
1 <input type="button" value="Quadrat errechnen" onclick="Quadrat()">
```

DOM

- Ziel: Standardisierung des Zugriffs auf HTML- und XML-Dokumente
- Standard wird von W3C gepflegt
- DOM Level 1 – 3
- Baumstruktur mit Objekten und Methoden
- Historische Probleme
  - Unterschiedliche Browser-Unterstützung
  - Historisch unterschiedliche Modelle (Netscape, IE)
  - Vergleich: http://www.webdevout.net/browser-support

DOM Beispiel

Ermöglicht das manipulieren eine Webseite mit JavaScript

```
 1 <table>
 2   <thead>
 3     <tr>
 4       <th>Vorname</th>
 5       <th>Name</th>
 6     </tr>
 7   </thead>
 8   <tbody>
 9     <tr>
10       <td>Donald</td>
11       <td>Duck</td>
12     </tr>
13   </tbody>
14 </table>
```

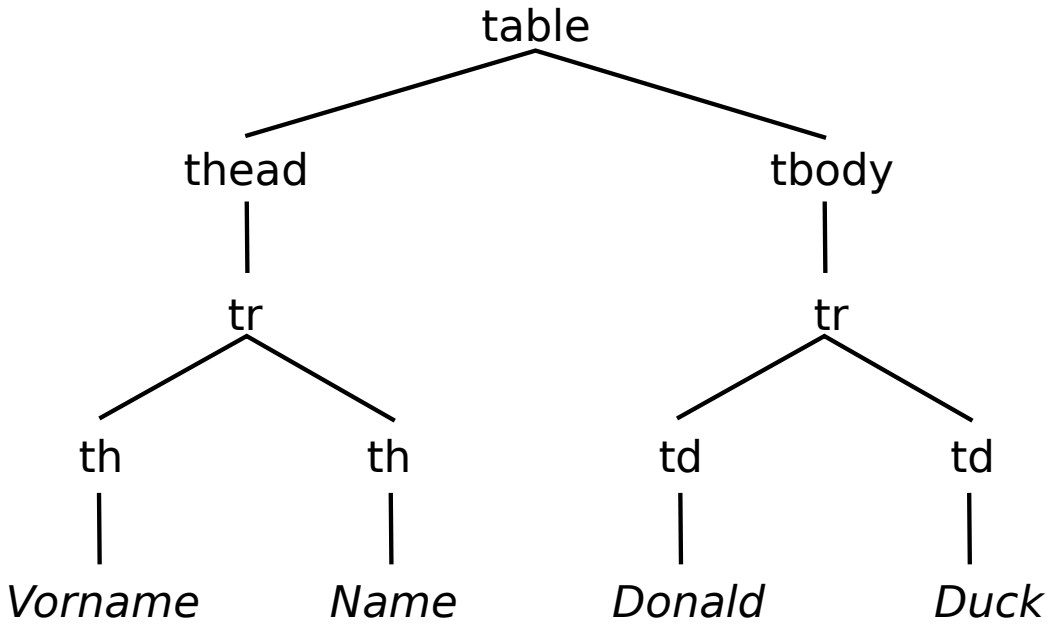


table
thead
tbody
tr
tr
th
th
td
td
Vorname
Name
Donald
Duck

DOM: Beziehungen

Kinder

Kinder von Knoten `table`: `thead`, `tbody`

```
1 document.getElementById("...").firstChild;
```

Geschwister

Knoten mit gemeinsamem Elternknoten `thead` und `tbody` sind Geschwister

```
1 document.getElementById("...").nextSibling;
```

Vater

`table` ist Vater von `thead`, `tbody`

```
1 document.getElementById("...").parentNode;
```

DOM: Knotentypen

Wurzelknoten des Dokuments:

`document`

Elementknoten

Spezialisierung eines allgemeinen Knotens (node)
Entspricht HTML-/XML-Element
Text ist ebenfalls ein Knoten

```
1 el = document.getElementById("...");
2 var text = el.nodeValue;
```

Attribute

Attribute eines Elementknotens

```
1 attr = document.getElementById("...").getAttribute("...");
```

Spezialisierte Typen, z.B. HTMLDivElement

DOM: Verarbeitung

Navigation

Definiert im DOM Core
Methoden auf node-Objekt
Bsp.: `firstChild, lastChild, parentNode, nextSibling`

DHTML: Erzeugen, verschieben, löschen:
`createElement, appendChild, removeChild, replaceChild`

```
1 var myH1 = document.createElement("h1");
2 var myText = document.createTextNode("Eine sehr dynamische Seite");
3 myH1.appendChild(myText);
4 var Ausgabebereich = document.getElementById("Bereich");
5 Ausgabebereich.appendChild(myH1);
```

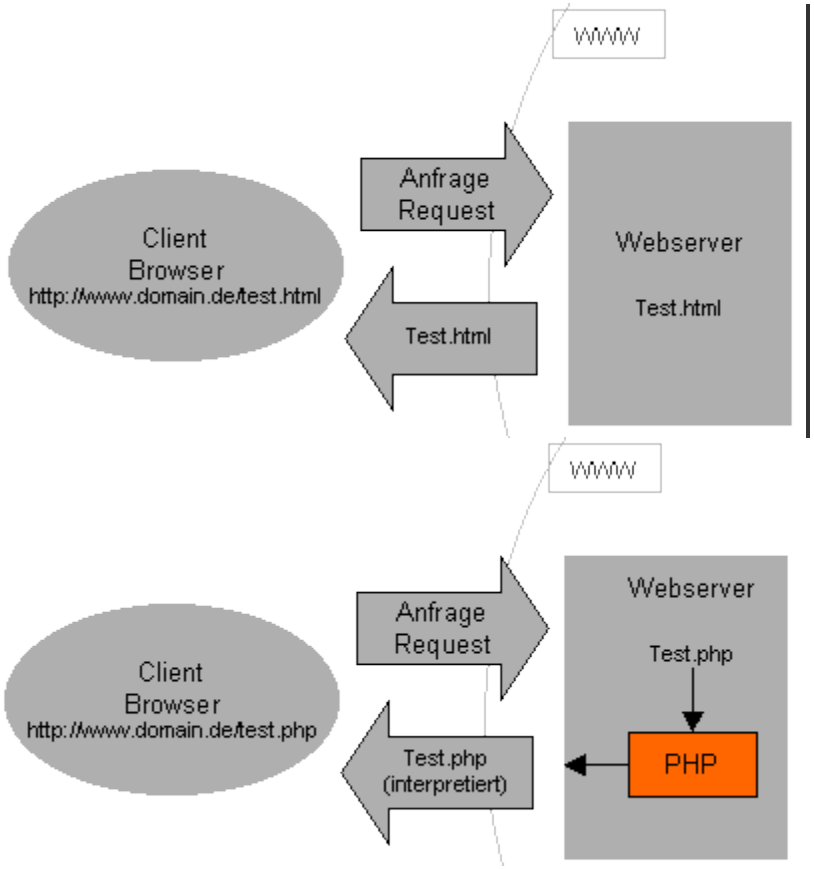
Auslesen / Schreiben: `el.firstChild.nodeValue = "Test";`

PHP

PHP: Hypertext Preprocessor

- Die populärste serverseitige Skriptsprache
- Syntax: Mischung aus Perl und C
- Früher: Einfache Skriptsprache; Heute: Durchaus ernsthafte Programmiersprache
- Vorteile:
  - Schnelle Lernkurve, Web-Entwickler kommt oft schnelle zum Ziel
  - Fast überall verfügbar (Shared-Webhosting)
  - Sehr gute und leicht verständliche Doku
- Nachteile:
  - Sprache ist nicht sehr "sauber"
  - Läd ein Fehler zu machen

Serverseitige Skriptsprachen




```
1 <html>
2 <head>
3 <title>Beispiel</title>
4 </head>
```

```
 5 <body>
 6 <h1>Aktuelle Serverzeit</h1>
 7 <?php
 8     print time();
 9 ?>
10 </body>
11 </html>
```

XAMPP
PHP selber am lokalen Rechner testen

All-in-One Serverpaket:

- Web-Server: Apache
  - MySQL
  - PHP
  - phpMyAdmin
  - Download: http://www.apachefriends.org/de/xampp.html

Was ist PHP eigentlich?

- Einbettbar in HTML: `<?php CODE ?>`
- Syntax ähnlich C: `if() { } else { }`
- Dynamische Typisierung
- `function name($var) { }` statt statt void/int
- Arrays sind zentraler Datentyp in PHP
    - Assoziative Arrays

Code Beispiele

```
1 <?php
2     $var["key"] = "value";
3     $var[0] = "value";
4     $var = "text";
5     $var = 5;                         //Variable
6     "Hallo"."Welt" ==  "HalloWelt"; # . (PUNKT) Verkettet Zeichen
7 ?>
```

Dokumentation: http://php.net/manual/de/index.php

PHP als Frame Ersatz

```php
<?php
    $file = basename($_GET['f']);
    if(is_file($file)) {
        include($file);
    }
?>
```

Links mit `index.php?f=datei.html` (Query String) ACHTUNG: Sicherheitskritisch: "basename" verhindert das einbinden von Systemdateien

PHP als Frame Ersatz Variante 2

- Fixe Teile der Website auslagern und jeweils einbinden
- Jede Seite erhält eigene Datei Endung der Datei muss von .html in .php geändert werden

Code

```
1 <?php include('header.php'); ?>
2 <div id="content">Mein Inhalt</div>
3 <?php include('footer.php'); ?>
```

Problem: Titel einer Seite etc. Lösung: Variablen; `<?php $subtitle = “Impressum“; ?>`