

Ziele des Moduls (offiziell)

Internettechnologien bilden die Basis crossmedialer Strategien. Die Studierenden erlernen diese komplexen Technologien und erlangen **Kompetenzen zum technischen Design** multifunktionaler Web-Systeme und erweitern ihr Wissen über **Einsatzmöglichkeiten wichtiger Klassen von Web-Anwendungssystemen**. Nach erfolgreichem Abschluss des Moduls kennen die Studierenden die technischen Grundlagen der Internettechnologien. Sie verfügen über **Grundwissen im Bereich Softwaretechnologien für Webanwendungen und Mobile Computing**. Sie können das Erlernte **selbständig anwenden**.

Dozentenplan_IT-2_Tramp

Zusammenfassung (offiziell)

Baustein IT 2 vermittelt die Softwaretechnik für die Entwicklung von Web-Applikationen. Im Rahmen des Web-Engineering werden **relevante Standards und Technologien** vorgestellt. Darüber hinaus werden **wichtige Klassen von Web-Anwendungssystemen eingeführt**. Dazu zählen insbesondere ECommerce-Systeme, Virtuelle Welten, Lehr- und Lernsysteme. Ein weiterer Schwerpunkt ist die Vorstellung von **Technologien zur Entwicklung ubiquitärer Anwendungen** wie Apps und Informationssystem für mobile Endgeräte, Tablet-PCs etc.

<https://www.leipzigschoolofmedia.de/masterstudiengaenge/crossmedia-management/module-studieninhalte/aufbau-und-nutzung-von-internettechnologien.html>

Prüfungsleistung: Projektarbeit; Bearbeitungszeit: 4 Wochen;
Themenausgabe: 05.03.2016; Abgabetermin: 02.04.2016

Baustein Tragweite / Buzzword Bingo

Relevante Standards und Technologien

- **HTML***, **CSS*** (+ SASS, LESS), **SVG**, **JavaScript / ECMAScript**, **DOM**, Flash, **jquery** + jquery ui, ...
- **Semantic Web**, **RDFa**, **Microdata**, **schema.org**
- **HTTP**, **Ajax / Ajaj**, **JSON**, **JSONP**, **CORS**, ***RPC**, ...
- **PHP**, **Ruby**, **node.js**, **Java**, ...

Wichtige Klassen von Web-Anwendungssystemen

- **E-Commerce**, Social Web / Social Network, CMS, **Blogs**, E-Learning, Virtuelle Welten, ...

Technologien zur Entwicklung von ubiquitärer Anwendungen

- iOS, Android, Windows Mobile, BlackBerry, Chrome OS, ...
- Native vs. Framework, PhoneGap / Apache Cordova, Jo, **jQuery Mobile**, Sencha Touch, M-Project, jQTouch, Titanium, ...

Themen dieses Bausteins - nach Tag

14.01.2016

- **HTTP / Big Picture (oder: Wie das Web funktioniert)**
- **HTML(5) (oder: Wie man Webseiten baut)**
- **CSS (oder: Wie man Webseiten styled)**

11.02.2016

- Semantic Web Primer, schema.org
- Apache / PHP / WordPress Installation
- JavaScript (oder: Wie man spannende Webseiten baut)

05.03.2016

- PHP / Datenbanken (oder: Wie man Webseiten generiert)
- Magento
- Entwicklung für Mobile Devices
- Management von Software-Projekten
- Projektbesprechung

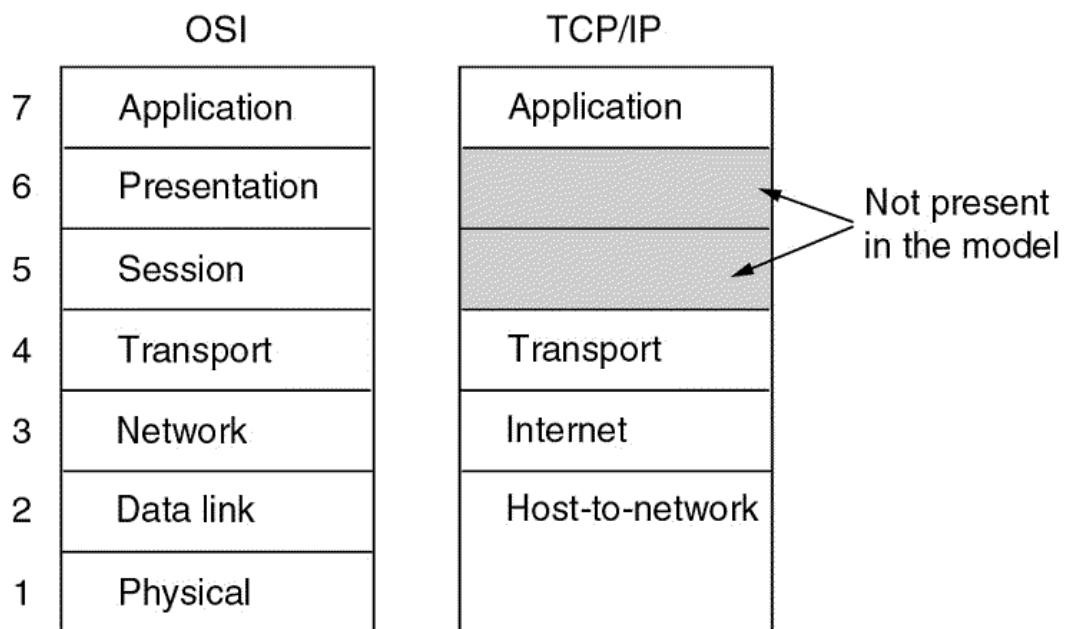
Hypertext Transfer Protocol (HTTP) ...

the mother of all communication on the

World Wide Web ... wait - not true

anymore!

ISO/OSI vs. TCP/IP Protocol Stack



Via: marco.panizza.name

Hypertext Transfer Protocol (HTTP)

Basic Attributes

- HTTP is a stateless request - response protocol
- HTTP is an application layer protocol
- HTTP is a protocol to fetch and manipulate resources

Basic Workflow

- The client sends a request message
- The server answers with a response message
- since HTTP 1.1: persistent connections and pipelining
- outlook HTTP 2.0: multiplexing / simultaneous streams / server push and other speed enhancements (based on SPDY)

Typical Stakeholder

- Typical clients: browser or applications on your desktop computer or mobile device
 - but also HTTP(S) in file sync, operating system updates and other parts of your software stack
- Typical server: web server application on a host in a data center

HTTP Request Message: Overview

- Request line

```
GET /index.html HTTP/1.1
```

- Header

```
Accept-Language: en
```

- Message body (optional)

HTTP Request Line

```
GET /index.html HTTP/1.1
```

Verbs

- GET, HEAD, POST, OPTIONS (the most important ones)
- PUT, DELETE, TRACE, CONNECT, PATCH

Requested Resource

The local part of an URL (without fragment identifier)

- URL: Uniform Resource Locator

```
scheme://domain:port/path?query_string#fragment_id
```

```
http://www.bing.com/search?q=Leipzig+School+of+Media
```

URLs, Resources and Representations

Concepts

- An URL identifies a "resource"
- A resource is anything identified by an URL
- Resources may have representations, which can be received by dereferencing the URL
- Representations are concrete data and may vary with time
- Resources are abstract
- Please distinguish between information resources vs. non-information resources !!!

Usage

- Resources are used to define an information space
- URIs do not only identify resources; they can be used to access and interact with resources as well

HTTP Request Header

- name-value pairs
- colon separated
- clear text
- used for content negotiation (see: [quality values](#))
- [List of HTTP header fields](#)

Example Request

```
HEAD / HTTP/1.1
User-Agent: curl/7.21.3 (i686-pc-linux-gnu)
Host: google.de
Accept: */*
```

HTTP Response Example

Tool: cURL

```
curl(.exe) -X HEAD -v http://google.de
```

Output

```
HTTP/1.1 301 Moved Permanently
Location: http://www.google.de/
Content-Type: text/html; charset=UTF-8
Date: Thu, 17 Nov 2011 00:21:37 GMT
Expires: Sat, 17 Dec 2011 00:21:37 GMT
Cache-Control: public, max-age=2592000
Server: gws
Content-Length: 218
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
```

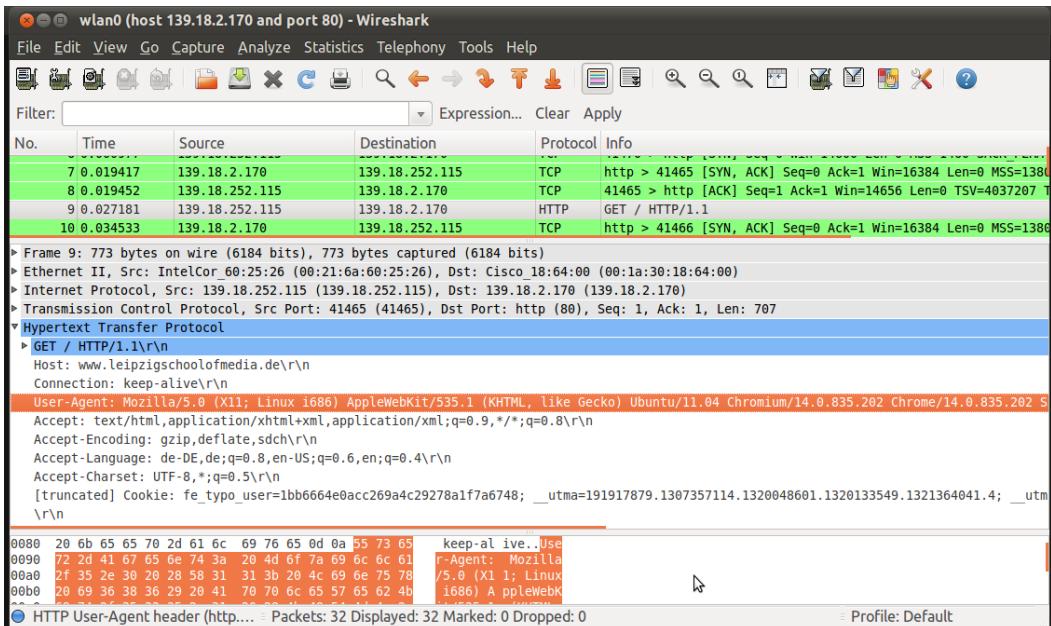
HTTP Response Codes

Important Status Codes

- 200 OK ... everything is fine
- 301 Moved Permanently ... change your bookmark
- 303 See Other ... follow the light
- 401 Unauthorized ... please send credentials
- 403 Forbidden ... not for you
- 404 Not Found ... broken link
- 500 Internal Server Error ... something is wrong

see also: [List of HTTP Status Codes](#)

Practice: Deep-Inspection of your Browser Communication



Challenge: Use wireshark to analyze what your browser sends and receives.

Limit the network traffic capturing to HTTP communication with the LSoM server.

Review the sent and received HTTP packets and notice, which HTTP header fields your browser creates.

Resources: Web Architecture / HTTP

- [Architecture of the World Wide Web, Volume One](#); W3C Recommendation 15 December 2004 ... this architecture document discusses the core design.
- [Web Architecture and Information Management](#); Course from the School of Information, UC Berkeley
- Wikipedia pages about:
 - [Uniform Resource Locator](#)
 - [Hypertext Transfer Protocol](#)
 - [WebDAV](#) (to look beyond basic HTTP)
- [wireshark](#) - network protocol analyzer
- [curl](#) - command line networking tool
- [Live HTTP headers](#) - Firefox plugin to watch HTTP headers

HTML Wakeup please! :-)

HTML Introduction

HTML is the acronym for **HyperText Markup Language**

- **Hypertext:** text with hyperlinks
 - Hyperlinks: references which can be used for more than text references (comes from the Greek prefix "ὑπέρ- and means "over" or "beyond")
 - Both terms were coined by Ted Nelson and Douglas Engelbart influenced by the article As We May Think by Vannevar Bush from 1945 (!)
- **Markup:** annotations in a document's content
- **Language:** an artificial language designed to express computations

HTML was invented by Tim Berners-Lee in 1991 and is now maintained by the W3C and (since HTML5) the WHATWG.

HTML Markup Basics: Tags

Plain text is annotated with tag elements:

```
not annotated text <tagname>annotated text</tagname> not  
annotated text
```

Tag elements can be used nested:

```
<ol>  
    <li>plain list item</li>  
    <li>list item <q>with <span>wrong</q></span>  
nesting</li>  
</ol>
```

HTML Markup Basics: attributes

Tag elements can be extended by attributes

```
<tagname attribute1="value1"  
attribute2="value2">text</tagname>  
<a href="http://aksw.org/">AKSW Homepage</a>
```

Some attributes can be used with all tags, some only with specific tags:

- Global attributes:
 - `class`, `id`, `title`, `style`, `dir`, ...
 - + `data-*` (custom data attributes)
 - + `on*` (event handler attributes)
- Attributes for specific tag elements:
 - e.g. `href`, `media`, `rel` are specifically allowed for the anchor tag a
 - please refer this list of tag elements

HTML Document Structure

A basic HTML(5) document is structured like this:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sample page</title>
  </head>
  <body>
    <h1>Sample page</h1>
    <p>This is a <a href="demo.html">simple</a> sample.</p>
    <!-- this is a comment -->
  </body>
</html>
```

document = doctype declaration + HTML document (= HTML head + HTML body)

Physical vs. Logical Markup

Physical Markup

- Tag elements which do strictly define the style: **b** (bold), big (font bigger)

Logical / Semantic Markup

- Tag elements which annotate a meaning and foster separation between style and structure
- Example: **em** or **strong** instead of **b**
- HTML5 introduces a lot new semantic tag elements: article, section, header / footer, nav, aside, figure / figcaption
- `<time datetim="2014-02-13">13.02.2014</time>`

Try to avoid physical markup completely!

- 10 HTML Tag Crimes You Really Shouldn't Commit
- Tableless Web Design

HTML Entities

Reserved characters must be replaced with character entities

Most important entities

- < (less than): **<**;
- > (greater than): **>**;
- & (ampersand): **&**;

Other entities

[List of HTML Character entity references](#)

Specific Tags: h*, p, a

Headings

Headings are defined with the h1 (most important) to h6 (least important) tags.

```
<h1>Leipzig School of Media<h1>
```

Paragraphs

```
<p>A nice little paragraph<p>
```

Links

```
<a href="http://aksw.org">AKSW</a>
```

Specific Tags: div, span

div

div elements are used to structure the page (where no other structure element is suitable)

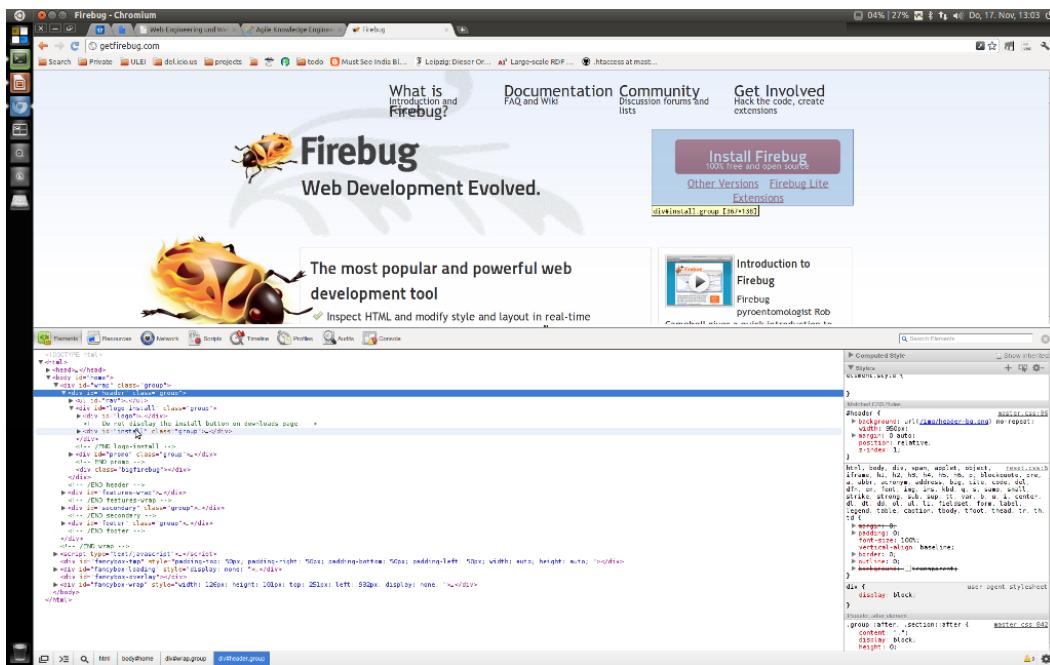
```
<div class="conclusion"><p>...</p><p>...</p></div>
```

span

span elements are used to annotate the text inline (where no other inline element is suitable)

```
The concert took place in <span  
class="location">Leipzig</span>.
```

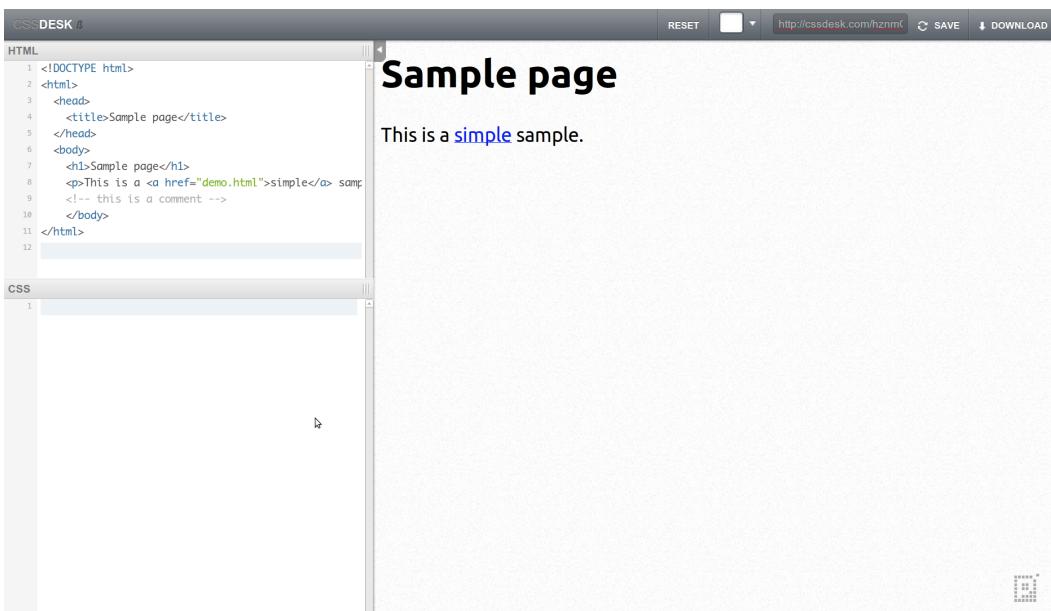
Practice: HTML Inspection with Firebug or Chrome Developer Tools



Challenge: Install [Firebug for Mozilla Firefox](#) or use the [Chrome Developer Tools](#) to inspect the webpage of your choice.

Try to find physical markup and other bad tag or table based layouts.

Practice: Use CSSDesk play around with HTML Tag Elements



The screenshot shows the CSSDesk interface. On the left, there are two panes: 'HTML' and 'CSS'. The 'HTML' pane contains the following code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Sample page</title>
5   </head>
6   <body>
7     <h1>Sample page</h1>
8     <p>This is a <a href="demo.html">simple</a> sample
9     <!-- this is a comment -->
10    </body>
11 </html>
```

The 'CSS' pane is currently empty, showing only a single line of code.

On the right, the preview area displays the output of the HTML code. It shows a large 'Sample page' heading and a paragraph below it. The word 'simple' in the paragraph is blue and underlined, indicating it is a link.

Challenge: Use CSSDesk or dabblet and play around with different HTML tag elements. This should include: headings, paragraphs, links, lists and logical inline markup.

HTML Resources

These resources will help you as tutorials, references and on-line tools:

- [CSSDesk / dabblet](#) (Online HTML/CSS Sandbox)
- [Detailed list of HTML elements](#) (Reference)
- [SelfHTML](#) (Tutorial / Reference / German)
- [Markup validation service @ W3C](#) (Online service / tool)
- [HTML @ w3schools.com](#) (Tutorial)
- [List of HTML Character entity references](#) (Reference)

CSS Cascading Style Sheets

Inhaltsverzeichnis



- Übersicht
- Einbindung
- Selektoren
- Schriftformatierung
- Textstrukturierung
- Listen- und Tabellenformatierung
- Positionierung

- CSS – Cascading Style Sheets
- Sprache zur Definition von Layouteigenschaften für alle Elemente in einem HTML-Dokument (ebenso bei XML-Dokumenten)

```
h1 {  
    color:red;  
    font-size:2em;  
    font-family:sans-serif;  
    margin-bottom:1em;  
    text-align:center;  
}
```

- für verschiedene Ausgabemedien können unterschiedliche Stylesheets definiert werden (Bildschirm, Ausdruck, Sprachausgabe, ...)
- Versionen (erarbeitet vom W3C)
 - 1.0: 1996 – in aktuellen Browsern fast vollständig unterstützt
 - 2.0: 1998 – teilweise unvollständig / nicht korrekt unterstützt
 - Arbeiten an 2.1 und 3.0

- CSS-Definitionen können in eigener Datei gespeichert und von HTML-Dokumenten aus referenziert werden
 - wichtigster Anwendungsfall, erlaubt Layoutänderungen an zentraler Stelle, die sich auf viele (alle) HTML-Seiten einer Website auswirken
- CSS-Definitionen können auch in HTML-Datei an zentraler Stelle bzw. in Elementen eingebettet werden
- Fall 1: CSS in externer Datei, Referenz. im HTML-Dok.

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

bib.css

```
h1 { color:red; }
p { margin-left:20px; } ...
```

```
<head>...
<style type="text/css">
  @import url("css/bib.css");
</style>...
</head>
```

Hinweise:

- mehrere *link*-Elemente oder *@import*-Anweisungen möglich
- Inhalt von *<style>* kann durch *<!-- INHALT -->* vor alten Browsern geschützt werden

Einbindung von CSS



MASTER
PROGRAMM
MEDIEN
LEIPZIG

- Fall 2: CSS im HTML-Dokument zentral definiert

```
<head>...
<style type="text/css">
  h1 { color:red; }
  p { margin-left:20px; }
  ...
</style>...
</head>
```

- ermöglicht Überschreiben importierter CSS-Angaben

Hochschulbibliothek

Willkommen auf den Internetseiten der Bibliothek der Hochschule für Technik, Wirtschaft und Kultur Leipzig. Der Internetauftritt dient der Informationssuche und Online-Recherche.

- Fall 3: CSS direkt im Element definieren

```
<body>...
<h1 style="color:red;">Hochschulbibliothek</h1>

<p style="margin-left:20px;">
  Willkommen auf ...
</p>

</body>
```

allgemein: <ELEM style="eigenschaft1:wert1; eigenschaft2:wert2;...">

- durch CSS wird Trennung von Struktur+Inhalt (HTML-Dokument) und Layout (CSS-Datei) erreicht
 - ermöglicht Wiederverwendung des Inhaltes in verschiedenen Ausgabemedien
 - pro Ausgabemedium unterschiedliches Layout

```
<link rel="stylesheet" type="text/css" href="css/bib.css" media="all" />
<link rel="stylesheet" type="text/css" href="css/druck.css" media="print" />
```

```
<style type="text/css">
  @import url("css/bib.css") all;
  @import url("css/druck.css") print;
  @import url("css/screen.css") screen, projection;
  ...
</style>
```

```
<style type="text/css">
  @media screen {
    h1 { color:red; }
    ...
  }
</style>
```

Ziele (Auswahl):
all, braille,
handheld, print,
projection,
screen, tty, tv

- wenn Style-Angaben nicht direkt im Element erfolgen, muss über einen Selektor angegeben werden, auf welches / welche Element(e) sie sich beziehen

```
SELECTOR {  
    eigenschaft1:wert1;  
    eigenschaft2:wert2;  
    ...  
}
```

Beispiel

```
h1 {  
    color:red;  
    font-size:2em;  
}
```

- Selektoren (CSS 1.0)
 - * – gilt für alle Elemente
 - e1 – gilt für Elementtyp e1
 - e1, e2 – gilt für Elementtypen e1 und e2
 - e1 e2 – gilt für Elementtypen e2, die unterhalb von e1 liegen

```
* {margin:0; padding:0}
```

```
p {margin-left:20px}
```

```
h1, h2 {font-weight:bold}
```

```
ul ul {font-size:0.8em}
```

- weitere Selektoren (CSS 2.0)
 - $e1 * e2$ – gilt für alle Elementtypen $e2$, die mindestens 2 Ebenen unterhalb von $e1$ liegen

```
div * li {margin-left:20px}
```

- $e1 > e2$ – gilt für Elementtypen $e2$, die auf einer Ebenenstufe unterhalb von $e1$ liegen **h1 > strong {font-style:italic}**
- $e1 + e2$ – gilt für Elementtypen $e2$, die unmittelbar auf Elemente vom Typ $e1$ folgen **h1 + p {text-align:center}**
- Achtung: $,>$ und $,+$ vom Internet Explorer 6 noch nicht unterstützt

- Selektion über Klassen
 - Elemente können Klassen zugeordnet werden

```
<p class="abstract">...</p>  
<li class="menu active">...</li>
```

mehrere Klassen sind möglich

- eigene CSS-Definitionen für Klassen

```
.abstract {  
    text-align:center;  
}
```

wird auf alle Elemente,
der die Klasse *abstract*
zugewiesen wurde,
angewendet

```
p.abstract {  
    text-align:center;  
}
```

wird nur auf *p*-Elemente,
denen die Klasse *abstract*
zugewiesen wurde,
angewendet

- Selektion über *id*-Attribut (Individualformate)
 - jedes Element kann ein *id*-Attribut enthalten, dessen Wert dokumentweit eindeutig sein muss
 - über den *id*-Attributwert kann ein Element eindeutig referenziert (angesprochen) werden

```
<p id="abstract">...</p>
```

- CSS-Definition für *id*-Wert

```
#abstract {  
    text-align:center;  
}
```

- Pseudoklassen (mehr in CSS3)
 - ermöglichen Formatdefinitionen für Inhalte, die sich nicht über die Angabe eines Elementes bestimmen lassen (z.B. noch nicht besuchter Verweis, erste Zeile eines Absatzes)
 - Pseudoklassen für Verweise

```
a { font-weight:bold; }  
a:link { color:blue;  
           text-decoration:none; }  
a:visited { color:silver;  
           text-decoration:none; }  
a:focus {color:red;  
           text-decoration:underline; }  
a:hover {color:white;  
           background:blue;  
           text-decoration:none; }  
a:active {color:lime;  
           text-decoration:underline; }
```

Dieses ist ein **normaler Link** und hier haben wir einen **besuchten Link**. Weiterhin hat **dieser Link** z.B. durch Tastaturauswahl den Fokus und über **diesem Link** befindet sich die Maus (hover).

Hier wird **der Link** gerade angeklickt und ist damit 'active'.

- Pseudoklassen für Absätze

```
.glosar p:first-line { font-weight:bold }  
.glosar p:first-letter { font-size:200%; color:red }  
.glosar p:first-child { background-color:#C0C0C0; }
```

```
<div class="glosar">  
  <p>Dieses kurze Glosar dient der  
    Demonstration der  
    Absatz-Pseudoklassen.  
  </p>  
  <p>a ist ein Element, mit dessen  
    Hilfe Verweise eingefügt werden  
  </p>  
  <p>form dient zur Definition von  
    Formularen</p>  
  <p>table erlaubt die Erstellung von  
    Tabellen</p>  
</div>
```

Dieses kurze Glosar
dient der Demonstration der
Absatz-Pseudoklassen.

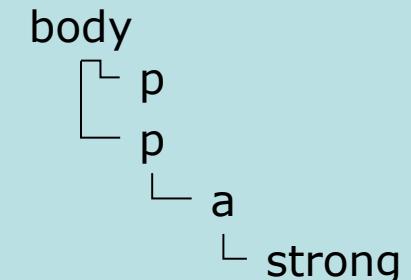
a ist ein Element, mit
dessen Hilfe Verweise
eingefügt werden

form dient zur Definition
von Formularen

table erlaubt die
Erstellung von Tabellen

- viele durch CSS festgelegte Eigenschaften werden an untergeordnete Elemente weiter vererbt
 - spart Arbeitsaufwand
 - vereinfacht Wartung

```
<p style="font-family:sans-serif;">  
  Ein Absatz mit einer serifenlosen Schriftart.  
</p>  
<p style="background:yellow;">  
  Hier ein Absatz mit einem  
  <a style="color:red;"  
    href="#test"><strong>wichtigen</strong>Verweis</a>,  
  der die Vererbung von CSS-Eigenschaften demonstriert.  
</p>
```



Ein Absatz mit einer serifenlosen Schriftart.

Hier ein Absatz mit einem wichtigenVerweis, der die Vererbung von CSS-Eigenschaften demonstriert.

- es kann mehrere CSS-Angaben geben, die auf ein Element zutreffen

```
* {color:black;}  
p {color:gray;}  
.wichtig {color:red;}  
#abstract {color:blue;}
```

```
<p id="abstract" class="wichtig">  
  In welcher Farbe werde ich  
  dargestellt?  
</p>
```

- Regeln zur Kaskadierung (Cascading) legen die Priorität der CSS-Angaben fest
 - Reihenfolge (absteigende Priorität):
 - style-Attribut in Element
 - Selektion über id-Attribut
 - Selektion über Klasse
 - Selektion über Element
 - bei mehreren Angaben mit gleicher Wichtung entscheidet Reihenfolge: die letzte Angabe überschreibt vorangegangene

Container-Elemente



- Container-Elemente `<div>`, `` dienen nur der Gruppierung von Inhalten / Elementen in logische Bereiche zwecks gemeinsamer Formatierung
- `<div>` Block-Element, kann weitere Block-Elemente enthalten

```
<h1>DIV-Test</h1>
<div style="border:solid; background:lightblue;
             margin-left:20px; padding:5px;">
    <h2>DIV-Block</h2>
    <p>Mit dem div-Element...</p>
    <p>können mehrere Block-Elemente...</p>
    <p>gruppiert und gemeinsam
       formatiert werden.</p>
</div>
```

DIV-Test

DIV-Block

Mit dem div-Element...

können mehrere Block-Elemente...

gruppiert und gemeinsam formatiert werden.

- z.B. verwendet, um Webseite in Bereiche / Spalten aufzuteilen

- Inline-Element, speziell für Markierungen im Text

<p>

Mit dem span-Element lassen sich

bestimmte Textstellen,
die auch selber
Inline-Elemente enthalten können,
auszeichnen und durch CSS formatieren.

</p>

Mit dem span-Element lassen sich
bestimmte Textstellen, die auch **selber**
Inline-Elemente enthalten können,
auszeichnen und durch CSS formatieren.

Schriftformatierung



- Schriftart: *font-family*

```
<p style="font-family:'Times New Roman',Times,serif">
```

Schrift mit Serifen.

```
</p>
```

```
<p style="font-family:Arial,Helvetica,sans-serif">
```

Schrift ohne Serifen.

```
</p>
```

```
<p style="font-family:cursive">
```

Eine Schreibschrift.

```
</p>
```

```
<p style="font-family:fantasy">
```

Eine ungewöhnliche Schrift.

```
</p>
```

```
<p style="font-family:monospace">
```

Eine dictengleiche Schrift.

```
</p>
```

Schrift mit Serifen.

Schrift ohne Serifen.

Eine Schreibschrift.

Eine ungewöhnliche Schrift.

Eine dictengleiche Schrift.

generische
Schriftfamilie



Schriftformatierung



- Schriftstil: *font-style*

```
<p style="font-style:normal">
```

Normale Schrift

```
</p>
```

```
<p style="font-style:italic">
```

Kursiver Schriftstil

```
</p>
```

```
<p style="font-style:oblique">
```

Schräggestellter Schriftstil

```
</p>
```

Normale Schrift

Kursiver Schriftstil

Schräggestellter Schriftstil

- Schriftvariante: *font-variant*

```
<p style="font-variant:normal">
```

Normale Schrift

```
</p>
```

Normale Schrift

```
<p style="font-variant:small-caps">
```

Kapitälchen

```
</p>
```

KAPITÄLCHEN

- Schriftgröße: *font-size*
 - absolut über Schlüsselworte

```
<span style="font-size:xx-small">xx-small</span>
<span style="font-size:x-small">x-small</span>
<span style="font-size:small">small</span>
<span style="font-size:medium">medium</span>
<span style="font-size:large">large</span>
<span style="font-size:x-large">x-large</span>
<span style="font-size:xx-large">xx-large</span>
```

xx-small x-small small medium large x-large XX-large

```
- <p>Normaler Text
    <span style="font-size:smaller">kleiner
        <span style="font-size:smaller">noch kleiner</span>
    </span><span style="font-size:larger">größer als normal</span>
</p>
```

Normaler Text kleiner noch kleiner größer als normal

- Schriftgröße: *font-size*
 - numerische Angabe

Exkurs: Einheiten für numerische Angaben in CSS

- absolute Einheiten: `pt` (1/72 Inches), `mm`, ...
- relative Einheiten: `%`, `em` (aktuelle Schriftgröße), `ex` (Höhe x')
- absolut/relativ: `px` (Pixel; absolut pro Anzeigegerät, relativ zw. verschiedenen Ausgabegeräten → Punktdichte)

Einheiten für Bildschirm-Schriftgröße

- relative Einheiten sollten verwendet werden
- px erlaubt genaue Einpassung der Schrift in festes Layout
aber: Schriftgröße abhängig von Punktdichte; teilweise keine Skalierung durch Benutzer möglich

```
<p>Normaler Text
  <span style="font-size:0.8em">kleiner
    <span style="font-size:80%">noch kleiner</span>
  </span><span style="font-size:1.3em">größer als normal</span>
</p>
```

Normaler Text kleiner noch kleiner größer als normal

- Schriftstärke: *font-weight*

```
<p>Normaler Text
  <span style="font-weight:bold">fett</span>
  <span style="font-weight:bolder">extrafett</span>
  <span style="font-weight:lighter">dünner</span>
</p><p>
  <span style="font-weight:100">100</span>
  ...
  <span style="font-weight:900">900</span>
</p>
```

Normaler Text **fett extrafett** dünner

100 200 300 400 500 **600 700 800 900**

- Zusammenfassung von Schriftattributen:

font: Stil Variante Stärke Größe/Zeilenhöhe Schriftfamilie

```
<p style="font:italic normal bold 1.3em/1.5em Verdana,sans-serif;">
  Eine kursive, fette und größere Schrift ohne Serifen</p>
```

Eine kursive, fette und größere Schrift ohne Serifen

- Schriftfarbe: *color*

```
<p style="color:red;">Rote Schrift</p>
<p style="color:#00ff00;">Grüne Schrift</p>
```

Rote Schrift

Grüne Schrift

Exkurs: Farbangaben in CSS

- Farben werden als Rot-, Grün- und Blau-Anteil (RGB) beschrieben
 - hexadezimal: **#rrggbb** – jeder Anteil im Bereich 00 – ff
Beispiel: #0080ff
 - dezimal: **rgb(ROT,GRÜN,BLAU)** – jeweils 0 – 255 oder 0% – 100%
Beispiel: rgb(0,128,255) rgb(0%,50%,100%)
- Verwendung reservierter Farbnamen

| | | | |
|--------|---------|---------|---------|
| black | #000000 | gray | #808080 |
| maroon | #800000 | red | #FF0000 |
| green | #008000 | lime | #00FF00 |
| olive | #808000 | yellow | #FFFF00 |
| navy | #000080 | blue | #0000FF |
| purple | #800080 | fuchsia | #FF00FF |
| teal | #008080 | aqua | #00FFFF |
| silver | #C0C0C0 | white | #FFFFFF |

Quelle: <http://de.selfhtml.org>

- Abstände innerhalb des Textes: *word-spacing, letter-spacing*

```
<p><span style="word-spacing:1em;">1em Abstand zwischen Wörtern und zusätzlich<br/><span style="letter-spacing:10px;">10px Abstand zwischen Zeichen</span><br/></span> </p>
```

1em Abstand zwischen Wörtern und zusätzlich 10px
A b s t a n d z w i s c h e n Z e i c h e n

- Textdekoration: *text-decoration*

```
<p> <span style="text-decoration:underline;">unterstrichen</span><br/><span style="text-decoration:overline;">überstrichen</span><br/><span style="text-decoration:line-through;">durchgestrichen</span> </p>
```

unterstrichen überstrichen durchgestrichen

- Texttransformation: *text-transform*

```
<p> <span style="text-transform:capitalize;">alle Wörter gross beginnen</span><br/><span style="text-transform:uppercase;">Grossschreibung</span><br/><span style="text-transform:lowercase;">Kleinschreibung</span> </p>
```

Alle Wörter Gross Beginnen GROSSSCHREIBUNG kleinschreibung

- Textausrichtung horizontal: *text-align*

```
<p style="text-align:left">Dieser Absatz ist linksbündig ausgerichtet. ...</p>
<p style="text-align:right">Dieser Absatz ist rechtsbündig ausgerichtet. ...</p>
<p style="text-align:center">Dieser Absatz ist zentriert. ...</p>
<p style="text-align:justify">Dieser Absatz ist in Blocksatz ausgerichtet. ...</p>
```

Dieser Absatz ist linksbündig ausgerichtet. Dieser Absatz ist linksbündig ausgerichtet. Dieser Absatz ist linksbündig ausgerichtet.

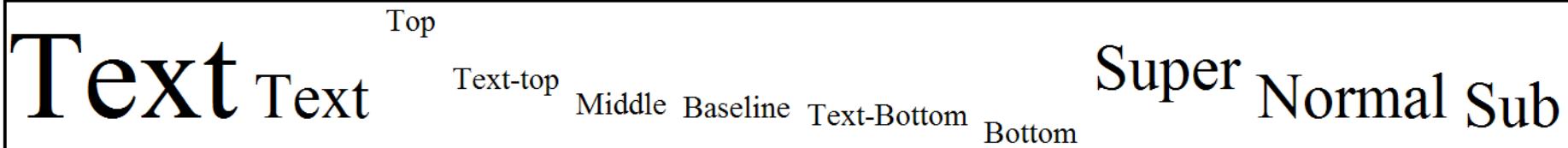
Dieser Absatz ist rechtsbündig ausgerichtet. Dieser Absatz ist rechtsbündig ausgerichtet. Dieser Absatz ist rechtsbündig ausgerichtet.

Dieser Absatz ist zentriert. Dieser Absatz ist zentriert. Dieser Absatz ist zentriert.

Dieser Absatz ist in Blocksatz ausgerichtet. Dieser Absatz ist in Blocksatz ausgerichtet. Dieser Absatz ist in Blocksatz ausgerichtet.

- Textausrichtung vertikal: *vertical-align*

```
<p style="border:solid; font-size:2em">  
  <span style="font-size:2em">Text</span>  
  Text  
  <span style="vertical-align:top; font-size:0.5em">Top</span>  
  <span style="vertical-align:text-top; font-size:0.5em">Text-top</span>  
  <span style="vertical-align:middle; font-size:0.5em">Middle</span>  
  <span style="vertical-align:baseline; font-size:0.5em">Baseline</span>  
  <span style="vertical-align:text-bottom; font-size:0.5em">Text-Bottom</span>  
  <span style="vertical-align:bottom; font-size:0.5em">Bottom</span>  
  <span style="vertical-align:super">Super</span>  
  Normal  
  <span style="vertical-align:sub">Sub</span>  
</p>
```



Textausrichtung / Absatzkontrolle



- Texteinrückung: *text-indent*

```
<p style="text-indent:1em; text-align:justify">
```

Die erste Zeile in diesem Absatz ist um 1em eingerückt.

```
</p>
```

Die erste Zeile in diesem Absatz ist um 1em eingerückt.

- Zeilenhöhe: *line-height*

```
<p style="line-height:1.5em">
```

Ein Absatz mit 1,5-facher Zeilenhöhe.

```
</p>
```

Ein Absatz mit 1,5-facher Zeilenhöhe.

- Steuerung des Textumbruchverhaltens: *white-space*

```
<p>
```

Ein normaler Text.

```
<span style="white-space:nowrap">
```

Ich darf nicht
umgebrochen werden.

```
</span>
```

```
<span style="white-space:pre">
```

Ich werde wie im
Quelldokument angezeigt.

```
</span>
```

```
</p>
```

Ein normaler Text.

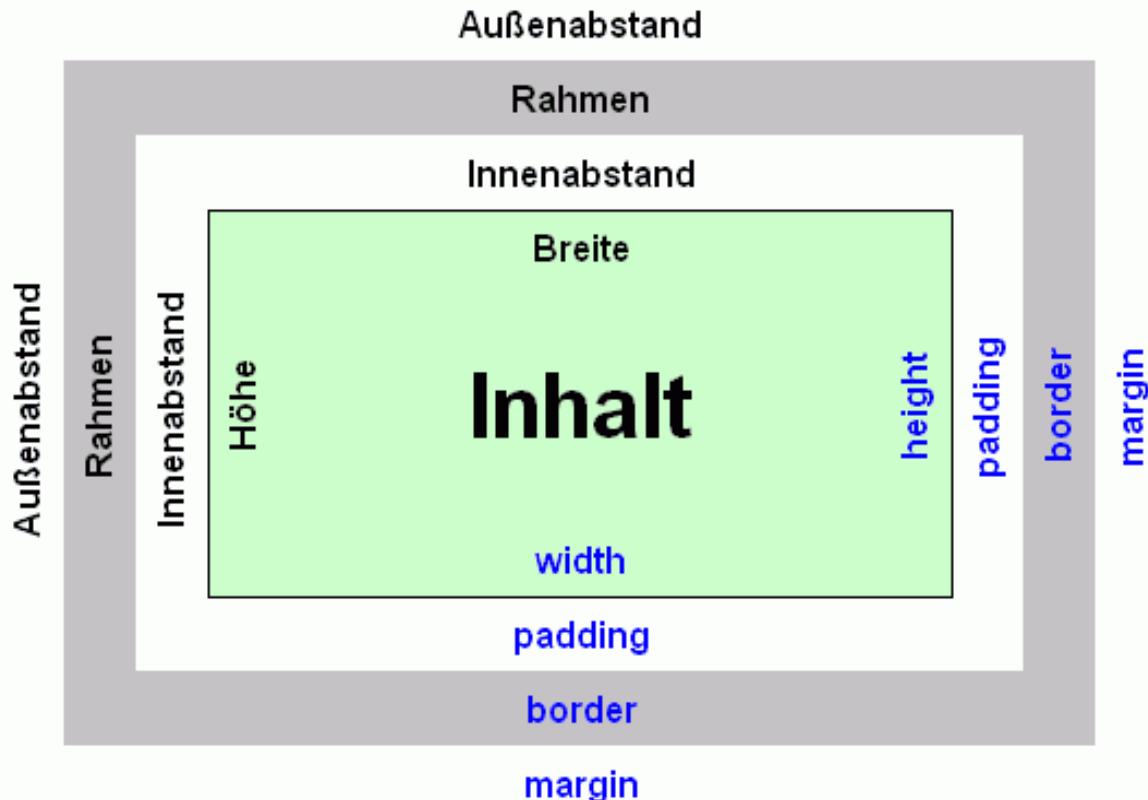
Ich darf nicht umgebrochen werden.

Ich werde wie im

Quelldokument angezeigt.

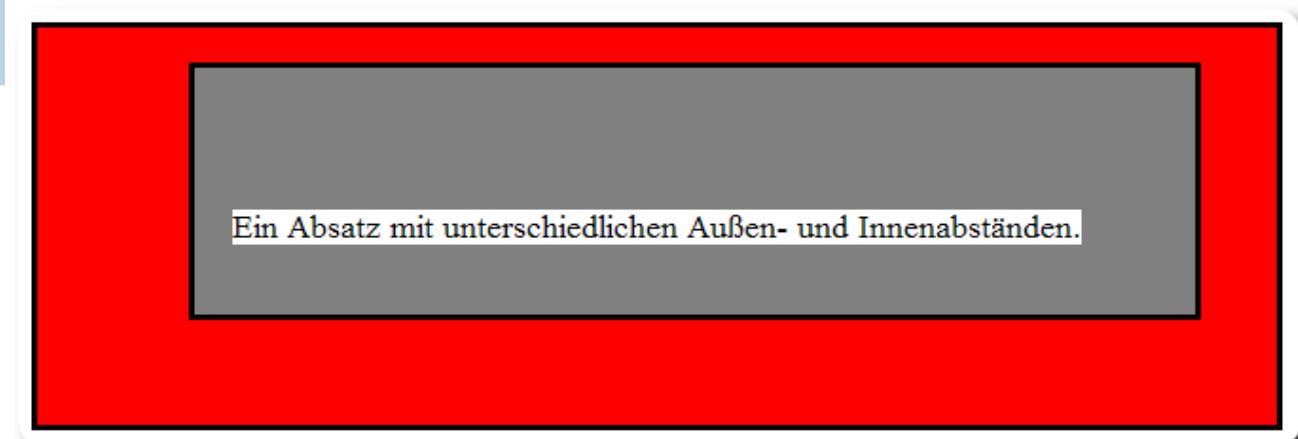
CSS Boxmodell

- Definiert die Berechnung der Breite und Höhe von Elementen



- Außenabstand: *margin* (*margin-top*, *margin-right*, *margin-bottom*, *margin-left*)
- Innenabstand: *padding*
(*padding-top*, *padding-right*, *padding-bottom*, *padding-left*)

```
<div style="border:solid; background:red; float:left">
  <p style="margin-top:20px; margin-right:40px; margin-bottom:60px; margin-left:80px;
    padding-top:80px; padding-right:60px; padding-bottom:40px;
    padding-left:20px; border:solid; background:gray">
    <span style="background:white">
      Ein Absatz mit unterschiedlichen Außen- und Innenabständen.</span>
  </p>
</div>
```



- alternativ: *style="margin:20px 40px 60px 80px; padding:80px 60px 40px 20px;..."*

- Rahmen: *border (-top, -right, -bottom, -left) (-width, -color, -style)*

```
<p style="border-top-width:thin; border-top-color:#ff0000; border-top-style:solid;  
border-right-width:5px; border-right-color:#00ff00; border-right-style:dotted;  
border-bottom-width:medium; border-bottom-color:#0000ff;  
border-bottom-style:dashed;  
border-left-width:thick; border-left-color:#808000; border-left-style:double;  
padding:5px"; float:left>
```

Eine bunte Rahmendemonstration.

```
</p>
```

Eine bunte Rahmendemonstration.

border-style: groove

border-style: ridge

border-style: inset

border-style: outset

Hintergrundgestaltung

- Hintergrund: *background* (*-color*, *-image*, *-repeat*, *-position*, *-attachment*)

```
<p style="background-color:lime;  
padding:20px; float:left">  
Ein hellgrüner Hintergrund.  
</p>
```

Ein hellgrüner Hintergrund.

```
<p style="background-image:url(logo.png);  
padding:20px; color:red; float:left">  
Ein Hintergrund-Logo gekachelt.  
</p>
```



Ein Logo gekachelt.

```
<p style="background-image:url(logo.png);  
background-repeat:repeat-x;  
padding:20px; float:left">  
Hintergrund-Logo, horizontal wiederholt.  
</p>
```



Hintergrund-Logo, horizontal wiederholt.

```
<p style="background-image:url(logo.png);  
background-repeat:repeat-y;  
padding:20px; float:left">  
Hintergrund-Logo, vertikal wiederholt.  
</p>
```



Hintergrund-Logo, vertikal wiederholt.

Hintergrundgestaltung



- Hintergrund: *background* (*-color*, *-image*, *-repeat*, *-position*, *-attachment*)

```
<p style="background-image:url(logo.png);  
         background-repeat:no-repeat;  
         padding:20px; float:left">  
Hintergrund-Logo, nicht wiederholt.  
</p>
```



Hintergrund-Logo, nicht wiederholt.

- Hintergrundbild-Wiederholung für ressourcensparende Verläufe
 - Bild:  *repeat-x*

1-Pixel Vertikalverlauf, horizontal wiederholt.

 1-Pixel Horizontalverlauf, vertikal wiederholt.

repeat-y

```
<p style="background-image:url(logo.png);  
         background-repeat:no-repeat;  
         background-position:center center;  
  
         padding:20px; float:left">  
Hintergrund-Logo, zentriert.  
</p>
```

Hintergrund-Logo, zentriert.

Hintergrundgestaltung



- Hintergrund: *background* (-color, -image, -repeat, -position, -attachment)

- Hintergrundbild-Positionierung: *background-position* x y
x, y: entweder numerische Angabe, oder

```
x: left, center, right; y: top, center, bottom  
<p style="background-image:url(logo.png);  
    background-repeat:no-repeat;  
    background-position:center center;  
        padding:20px; float:left">
```

Hintergrund-Logo, zentriert.

Hintergrund-Logo, zentriert.

- feststehender Hintergrund (wird beim Rollen des Inhalts nicht bewegt)

```
<body style="background-image:url(logo.png);  
    background-repeat:no-repeat;  
    background-position:left center;  
    background-attachment:fixed">...
```

```
<p style="padding:20px; float:left">
```

Erster Satz; Hintergrundlogo bleibt stehen.

Zweiter Satz; Hintergrundlogo bleibt stehen.

...

```
</p>...
```

Erster Satz; Hintergrundlogo bleibt stehen.
Zweiter Satz; Hintergrundlogo bleibt stehen.
Dritter Satz; Hintergrundlogo bleibt stehen.

Erster Satz; Hintergrundlogo bleibt stehen.
Zweiter Satz; Hintergrundlogo bleibt stehen.
Dritter Satz; Hintergrundlogo bleibt stehen.
Vierter Satz; Hintergrundlogo bleibt stehen.
Fünfter Satz; Hintergrundlogo bleibt stehen.

- Listenformatierung: *list-style* (*-type*, *-position*, *-image*)
 - Listentyp für geordnete Listen:
`decimal`, `lower-roman`, `upper-roman`, `lower-alpha`, `upper-alpha`

```
<ol style="list-style-type:lower-roman">
  <li>erster Punkt</li>
  <li>zweiter Punkt</li>
</ol>
```

- i. erster Punkt
- ii. zweiter Punkt

- Listentyp für ungeordnete Listen:
`disc`, `circle`, `square`

```
<ul style="list-style-type:disc">
  ...
</ul>  <ul style="list-style-type:circle">
  ...
</ul>  <ul style="list-style-type:square">
  ...
</ul>
```

- erster Punkt
 - zweiter Punkt
-
- erster Punkt
 - zweiter Punkt
-
- erster Punkt
 - zweiter Punkt

- kein Aufzählungszeichen: *list-style-type:none*

Listenformatierung

- Listenformatierung: *list-style* (*-type*, *-position*, *-image*)
 - Grafik als Aufzählungszeichen: *list-style-image*

```
<ul style="list-style-image:url(bullet_kugel.png)">
  <li>erster Punkt</li>
  <li>zweiter Punkt</li>
</ul>
```

- erster Punkt
- zweiter Punkt

- Einrückungsverhalten: *list-style-position*: inside | outside

```
<ol style="list-style-position:inside">
  <li>erster Punkt <br />
    zweite Zeile</li>
  <li>zweiter Punkt <br />
    zweite Zeile</li>
</ol>
```

- 1. erster Punkt
zweite Zeile
- 2. zweiter Punkt
zweite Zeile

```
<ol style="list-style-position:outside">
  <li>erster Punkt <br />
    zweite Zeile</li>
  <li>zweiter Punkt <br />
    zweite Zeile</li>
</ol>
```

- 1. erster Punkt
zweite Zeile
- 2. zweiter Punkt
zweite Zeile

- bisher behandelte CSS-Attribute können auch auf Tabellenelemente angewendet werden
(Textformat., Ausrichtung, Absatzkontrolle, Abstand, Rahmen, Hintergrund)
- noch benötigt
 - Festlegung Spaltenbreite unabhängig vom Inhalt
 - Zellenabstände / Zusammenfassung von Zellrahmen
- Spaltenbreite
 - normalerweise durch Inhalt bestimmt; Angabe von *width* ignoriert, wenn Inhalt mehr Platz beansprucht
 - mittels *table-layout:fixed* erhält *width* Vorrang

```
<table style="table-layout:fixed; width:100%">
  <tr>
    <td style="width:30px">Breite unabhängig von Riesenwörter</td>
    <td style="width:25%">Diese Spalte nimmt ¼ der Breite ein</td>
    <td>Spalte ohne definerte Breite; erhält den verbliebenen Platz</td>
  </tr>
</table>
```

| | | |
|---------------------------------------------|----------------------------------------|-------------------------------------------------------------|
| Breite unabhängig von Riesenwörter | Diese Spalte nimmt ¼ der Breite ein | Spalte ohne definerte Breite; erhält den verbliebenen Platz |
|---------------------------------------------|----------------------------------------|-------------------------------------------------------------|

- Zusammenfassung von Rahmen: *border-collapse:collapse* | *separate*

```
<!-- head Style-Definition -->
  |
```

```
<table style="border-collapse:separate">
<tr>
  <td>1. Spalte mit separatem Rahmen</td>
  <td>2. Spalte mit separatem Rahmen</td>
</tr>
</table>
```

| | |
|--------------------------------|--------------------------------|
| 1. Spalte mit separatem Rahmen | 2. Spalte mit separatem Rahmen |
|--------------------------------|--------------------------------|

```
<!-- head Style-Definition -->
td {border:thin solid; padding:5px}
```

```
<table style="border-collapse:collapse">
<tr>
  <td>1. Spalte, gemeinsamer Rahmen</td>
  <td>2. Spalte, gemeinsamer Rahmen</td>
</tr>
</table>
```

| | |
|-------------------------------|-------------------------------|
| 1. Spalte, gemeinsamer Rahmen | 2. Spalte, gemeinsamer Rahmen |
|-------------------------------|-------------------------------|

- Abstand zwischen Zellrahmen: *border-spacing*

```
<table style="border-spacing:20px;  
border:thin solid">  
<tr>  
  <td>Die Zellrahmen haben</td>  
  <td>einen Abstand von 20 Pixeln</td>  
</tr>  
</table>
```

Die Zellrahmen haben einen Abstand von 20 Pixeln

- Achtung: Internet Explorer (einschließlich Version 7) kennt diese CSS-Angabe nicht! Hierfür Attribut *cellspacing* im *table*-Element verwenden:
<table cellspacing="20px">...</table>

- Standardvorgaben der Positionierung von Elementinhalten können mit CSS überschrieben werden
 - Standard
 - Inhalte werden von links nach rechts und von oben nach unten auf Seite angeordnet
 - Block-Elemente beginnen auf neuer Zeile und erzeugen einen Absatz (maximale horizontale Ausdehnung)
- Umfließen von Elementen: **float: left | right | none**

```
<h1 style="float:left; width:3em;  
margin:0 0.2em 0.2em 0">panta rhei</h1>
```

```
<p>
```

Mit Hilfe der CSS-Angabe **<code>float</code>** können alle Elemente links oder rechts ausgerichtet und von den folgenden Inhalten umflossen werden. Dieses wird häufig für Bilder verwendet.

```
</p>
```

panta
rhei

Mit Hilfe der CSS-Angabe **float** können alle Elemente links oder rechts ausgerichtet und von den folgenden Inhalten umflossen werden. Dieses wird häufig für Bilder verwendet.

- soll nachfolgendes Element ein vorangegangenes nicht mehr umfließen, sondern unterhalb fortsetzen: **clear: left | right | both**

- Breite von Elementen festlegen: **width**

```
<p style="width:50px; border:thin solid; float:left;  
margin:0; color:blue">
```

Ein breitenbegrenzter Absatz.

```
</p>
```

```
<p style="width:150px; border:thin solid; float:left;  
margin:0">
```

Wieso ragt der Inhalt des vorangegangenen
Absatzes hier hinein?

```
</p>
```

Ein breitenbegrenzter Absatz. Wieso ragt der Inhalt des vorangegangenen Absatzes hier hinein?

- Festlegen, was mit zu großem Inhalt geschieht: **overflow**
visible | **hidden** | **scroll** | **auto**

style="...; overflow:scroll"

Ein breitenbegrenzter Absatz. Wieso ragt der Inhalt des vorangegangenen Absatzes hier hinein?

style="...; overflow:hidden"

Ein breitenbegrenzter Absatz. Wieso ragt der Inhalt des vorangegangenen Absatzes hier hinein?

- Höhe von Elementen festlegen: *height*

```
<p style="height:3em; width:6em; overflow:auto;  
border:thin solid; margin:0; color:blue">
```

Dieser Absatz wurde in Höhe und Breite begrenzt.

```
</p>
```

```
<p style="border:thin solid; margin:0">
```

Durch die overflow-Angabe ragt der vorangegangene Inhalt nicht in diesen Absatz hinein.

```
</p>
```

Dieser
Absatz
wurde in

Durch die overflow-Angabe ragt der vorangegangene Inhalt nicht in diesen Absatz hinein.

- neben der genauen Höhe / Breite lassen sich auch minimale und maximale Höhen / Breiten angeben:

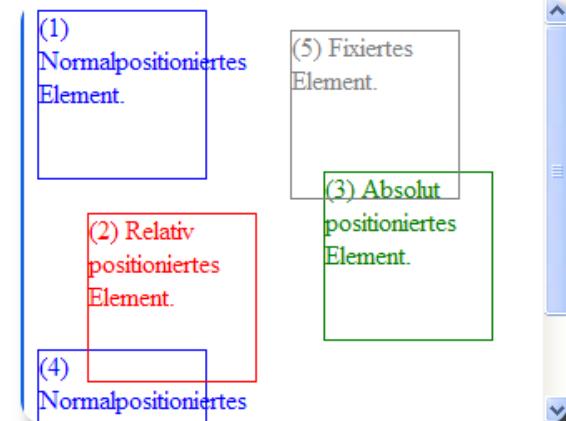
min-height, max-height, min-width, max-width

- aber: der Internet Explorer bis Version 6 interpretiert diese Angaben noch nicht

- Elementposition festlegen: *position*: static | relative | absolute | fixed
 - Elementinhalte können relativ zu ihrer Normalposition (*relative*), absolut auf der Webseite (*absolute*) oder an einem bestimmten Ort des Browserfensters (*fixed* – verändert Position beim Bildlauf („Scrollen“) des Fensterinhaltes nicht) positioniert werden; *static* entspricht dem Normalfall
 - die (relative) Position kann mittels *top*, *left*, *bottom*, *right* angegeben werden

```
<!-- head-css -->
p {height:100px; width:50px; border:thin solid; margin:0;}

<p style="color:blue">(1) Normalpositioniertes
Element.</p>
<p style="position:relative; top:20px; left:30px; color:red">
(2) Relativ positioniertes Element.</p>
<p style="position:absolute; bottom:50px; right:30px;
color:green">
(3) Absolut positioniertes Element.</p>
<p style="color:blue">(4) Normalpositioniertes
Element.</p>
<p style="position:fixed; top:20px; right:50px; color:gray">
(5) Fixiertes Element.</p>
```



- Anzeigeart: **display**: block | inline | none
 - kann ein Inline-Elemente in ein Block-Element (*block*) bzw. ein Block-Element in ein Inline-Element (*inline*) verwandeln
 - mit *none* wird das Element nicht dargestellt und auch kein Platz frei gehalten

```
<h1 style="display:inline">Die Überschrift</h1>
<p style="display:inline">
verhält sich wie ein Inline-Element
(wie auch dieser Absatz).
</p>
<p style="display:none">
Dieser Absatz wird nicht angezeigt.
</p>
<p>
Eine Hervorhebung wird
<em style="display:block">plötzlich</em> als
Block dargestellt.
</p>
```

Die Überschrift verhält sich wie ein Inline-Element (wie auch dieser Absatz).

Eine Hervorhebung wird *plötzlich* als Block dargestellt.

- SelfHTML (Dokumentation, die beste die es gibt :-))
 - <http://de.selfhtml.org/>
- CSS Validation Service
 - <http://jigsaw.w3.org/css-validator/>
- CSS Sandkasten
 - <http://cssdesk.com/>
- Neue HTML5 / CSS3 Features
 - <http://leaverou.me/ft2010/>
 - <http://slides.html5rocks.com/#semantics-markup-title>

← Aktuelle Browser benötigt !!

Übung: Wie kann dieses Printmedium im Web umgesetzt werden?



- Anordnung ohne Tabellen (Bootstrap, 960gs, blueprint)
- Was ist im Web sinnvoll, was nicht?
- Responsive Web Design? Mobile Devices?

CSS Resources

- [LSoM Slides \(PDF\)](#)
- [SelfHTML](#) (oldschool but still recommendable)
- Neue HTML5 / CSS3 Features:
 - <http://lea.verou.me/ft2010/>
- Helper:
 - [CSS Validation Service](#)
 - [CSS Desk](#) / [dabblet](#) (text sandbox)
 - [CSS 3.0 maker](#) (graphical sandbox)
 - [caniuse.com](#) (browser issue directory)
 - [saucelabs.com](#) (automated browser testing)
 - [Google Web Fonts](#)

Thank you for your Attention!

My WebID:

- <http://sebastian.tramp.name>