

Vorlesungsreihe Grundlagen der Informatik

HTML und CSS-Stylesheets

Prof. Dr.-Ing. Thomas Wiedemann
email: wiedem@informatik.htw-dresden.de



HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN (FH)
Fachbereich Informatik/Mathematik

- Übersicht (Historie / Allgemeines)
 - HTML
 - CSS-Stylesheets
-
- Hinweis: Aufgrund des sehr zahlreich und detailliert vorliegenden Online-Materials stellen die folgenden Folien nur eine Übersicht und Zusammenfassung zum Thema dar.
 - Details und konkrete Beispiele – siehe Quellen im Web
<http://wiki.selfhtml.org/> (Deutsch)
<http://www.selfhtml.org/> (englisch)
 - CSS- > <http://www.w3schools.com/css/> <http://www.css4you.de/>

- HTML und XML sind Datenbeschreibungssprachen auf Anwendungsebene (entspricht OSI-Schichten 6+7)
- Transport erfolgt über TCP im Client/Server-Modus (XML teilweise auch über UDP) oder aufsetzende Protokolle wie http
- Gemeinsame Wurzel ist SGML, eine sehr komplexe Dokumentenbeschreibungssprache aus den 80er Jahren
- HTML (Hypertext Markup Language) ist eine Beschreibungssprache für Text- und Multimediadarstellungen
- XML dient zur Definition neuer, anwendungsspezifischer Sprachen
- beide sind als Klartext-Format definiert
 - erfordern prinzipiell keine speziellen Editoren oder proprietären Programme
 - Verarbeitung erfolgt durch Interpretation (Parsen + Ausführung von zugeordneten Visualisierungsfunktionen)
 - vor allem der starke Leistungszuwachs der Prozessoren erlaubt eine derartige Echtzeitinterpretation ohne nennenswerte Verzögerungen beim Seitenaufbau

Historie von HTML

- HTML 1.0 entstand vor 1995, nicht mehr relevant (Tim Berner Lee am CERN)
- **HTML 2.0 wurde 1995 offizieller Sprachstandard**
 - ist kleinster gemeinsame Nenner aller Browser
 - war aber bereits zu seiner Verabschiedung veraltet, da im Rahmen des Browserkrieges durch Netscape zahlreiche Neuerungen wie Frames verfügbar waren
- **HTML 3.2 wurde Anfang 1997 verabschiedet**
 - Vorgängerversion 3.0 wurde ebenfalls von der Browserentwicklung überholt
 - HTML 3.2 entstand nach einer völligen Umarbeitung
 - neue Funktionen : Tabellen, aber Befehle zur physischen Textauszeichnung, was im Widerspruch zum Charakter von HTML stand und zum Teil daher wieder aufgehoben wurde
 - parallel entstand CSS als Zusatzdefinition zur genauen Textformatierung
- **HTML 4.0 wurde 1998 als Sprachstandard verabschiedet**
 - Schwerpunkt war die Rückbesinnung auf die Kernaufgaben von HTML
 - Neu: Internationalisierung, Einbindung von CSS Style Sheets und JavaScript
 - **Nach Fehlerbehebung mit Version 4.01 KEINE neuen Funktionen mehr**
- **Bis 2009 : XHTML-Neufassung von HTML 4.01 - 2009 EINGESTELLT**
-> HTML5 ab 2010 -> siehe HTML5-Vorlesung !

Das HTML-Format (Hyper Text Markup Language)

- ist ein universelles Format für alle Texte und die Einbindung von Grafiken und anderen Multimediaobjekten
- Hauptziel ist die möglichst flexible Darstellung auf verschiedensten Geräten
- HTML definiert daher (zumindest vom Grundkonzept her) keine festen Schriftarten oder Zeichengrößen, dies unterliegt der Freiheit des Anwenders
- von Tim Berners Lee anfangs als universelles Medium für verteilte Daten konzipiert (die Verlinkung war eine der ersten Funktionalitäten)
- Eingebettete, umfangreichere Grafiken und Multimediaobjekte werden erst nach dem Laden der HTML-Seite mit erneuten http-Request geladen und angezeigt
- generelle Struktur:
 - o einfaches Textformat mit Marken (Tags) zur Formatierung
 - o öffnende Tags werden mit `<...>` definiert
 - o schließende Tags werden mit `</...>`
 - `<H1>Hauptüberschrift</H1>` `Herzlich Willkommen ! `
- Tags können auch zusätzliche Attribute haben :
`Beispiel zur Textformatierung`

Allgemeine Regeln bei HTML

Im Gegensatz zu XML (!) werden

- Groß- und Kleinschreibung der Tags nicht unterschieden
 ist da gleiche wie
- muß die Reihenfolge des Öffnen und Schließens nicht unbedingt eingehalten werden : <I>Text</I> ist bei HTML erlaubt und funktioniert
- Umlaute oder andere Sonderzeichen sind speziell zu kodieren:
 - ä -> ä Ä -> Ä
 - ö -> ö ü -> ü ß -> ß
 - € -> €
 - < muß ersetzt werden durch < (besonders gefährlich !!!)
 - > muß ersetzt werden durch > (besonders gefährlich !!!)
 - & muß ersetzt werden durch &
 - " muß ersetzt werden durch "
 - Ergibt

In München steht ein Hofbräuhaus.

Dort gibt es Bier aus Maßkrügen.

Das Grundgerüst einer HTML ist wie folgt definiert :

```
<HTML>
<HEAD>
<TITLE>Dokumenttitel ...</TITLE>
</HEAD>
<BODY>
...
</BODY>
</HTML>
```

<HTML> startet die HTML-Datei
<HEAD> startet Kopfinformationen
<TITLE> definiert Dokumenttitel
</HEAD> beendet Kopfdefinition
<BODY> startet eigentlichen Inhalt
... Inhalt des Dokumentes
</BODY> beendet Inhalt
</HTML> beendet HTML-Datei

- Verstöße gegen diese Struktur oder nachfolgende Formatierungsregeln führen meist zu Fehlern bei der Dokumentdarstellung oder zu nicht darstellbaren Dokumenten.

Kurzüberblick zu den Formatierungsbefehlen (Details -> selfhtml)

- Zeichenformatierung `Fettschrift` `<I>Kursiv</I>` `Fett`
`<i>Kursiv</i>``<s>Durchgestrichen</s>` `<big>Groß</big>` `<small>Klein</small>`
`_{Tief}``^{Hoch}``<tt>Schreibmaschine</tt>`
- Zeilenumbruch `
` Achtung: Der normale Zeilenumbruch in der HTML-Datei wird ignoriert ! Eine Ausnahme bildet der Tag `<pre> ...</pre>` - dieser stellt den Text wie Standardtext mit Zeilenumbrüchen dar (sinnvoll für Listings)
- Absatzformatierung (führt auch zu Zeilenumbrüchen) `<P>Absatztext ... </P>`
- Absatzausrichtung `<P align="center"> ...</P>` - Standardausrichtung ist linksbündig
- Schriftdefinition `Test`
Achtung: sollte zukünftig durch CSS-Definitionen ersetzt werden
- Überschriften: `<H1>Ganz groß</H1>` ... `<H6>sehr kleine Überschrift</H1>`
- Logische Auszeichnungen (nicht vollständig) : `Stark`
`<cite>Zitat</cite>` `<var>Variable</var>` `<code>Programmcode</code>`
- Trennlinie `<hr size=7 size=20 width=50% align=left color="darkblue">`

- Aufzählungslisten (...)
- Nummerierte Listen (...)
- Definitionslisten (<dl>...</dl>)
- Verzeichnislisten (<dir>...</dir>)
- Menülisten (<menu>...</menu>)
- Das -Tag gibt in allen Arten den Listeneintrag an

Beispiel:

```
<ul>
```

```
<li>Probieren geht &uuml;ber Studieren</li>
```

```
<li>Liebe geht über Triebe</li>
```

```
<li>Tante f&auml;llt &uuml;ber Kante</li>
```

```
</ul>
```

- **Die Links stellen das mächtigste Element von HTML dar.**
- **Erst durch die Links entstand das World Wide Web.**

Verschiedene Optionen

- Verweise innerhalb des gleichen Webs (im gleichen Verzeichnis)
Preise -> Darstellung: [Preise](#)
- Verweise innerhalb des gleichen Webs in anderen Verzeichnissen **relativ**
Preise
- Verweise innerhalb des gleichen Webs in anderen Verzeichnissen **absolut**
Preise
- Verweise im gleichen Dokument auf eine Marke
AGB
erfordert Existenz von Marke Unsere AGB´s
- **Verweise in ein anderes Web**
Preise
- Verweis auf eine Emailadresse (führt meist zum Start des Emailprogramms)
Mailen Sie mir !

Einbinden von Grafiken

- Prinzipiell werden Grafiken oder andere Multimediaelemente über entsprechende Links auf die Multimediadatei eingebunden
- Für Grafiken existiert ein spezielles Tag ``
``
- das optionale Attribut `align="texttop"` erlaubt verschiedene Ausrichtungen

Hinweise:

- im `src`-Attribut ist der Link zur Datei (kann auch extern sein) anzugeben
- `width` und `height` der Grafik sollten angegeben werden, damit der Browser auch ohne vollständiges Laden der Grafik mit dem Aufbau der Seite beginnen kann !
- Der `alt`-Infotext dient zur textuellen Kennung (z.B. für Blinde, welche sich vorlesen lassen oder als Hinweis auf den Inhalt, falls die Grafik nicht ladbar ist)
- Grafiken können auch als Link definiert werden:
` `
- mehrere Links auf einer Grafik sind durch (->) Imagemaps realisierbar

Farbangaben

- NETSCAPE definierte für den RGB-Farbraum jeweils 6 Stufen für jeden Farbwert, damit sind $6*6*6 = 216$ Farben vordefiniert
- Angabe mit Hexdezimalwert
 - Knallroter Text

 - Blauer Text
 - Grüner Text
- Alle modernen Browser verstehen 240 Farbnamen, siehe http://www.w3schools.com/html/html_colornames.asp

Koodinatenangaben

- HTML erlaubt 2 verschiedene Arten der Angabe von Abmessungen oder Positionen :
 - In Prozent vom verfügbaren Platz - erkennbar am %-Zeichen
 - Absolut in Pixel –erkennbar an einheitenlosen Zahlenangaben
- beide Verfahren haben ihre Vor- und Nachteile
 - Prozentuale Werte passen sich gut an wechselnde Auflösungen an und sind für Hauptfenster
 - Absolute Angaben sind günstig für Bereiche mit festen Abmessungen (z.B. Menüleiste)

HTML-Tabellen

- Tabellen sind eines der wichtigsten Mittel zum Strukturieren des Inhalts (NICHT mehr des Designs von HTML-Seiten)
- zum Aufbau von Tabellen werden folgende Tags verwendet

<code><table></code>				
<code><tr></code>	<code><th></code> <code></th></code>	<code><th></code> <code></th></code>	<code><th></code> <code></th></code>	<code></tr></code>
<code><tr></code>	<code><td></code> <code></td></code>	<code><td></code> <code></td></code>	<code><td></code> <code></td></code>	<code></tr></code>
<code><tr></code>	<code><td></code> <code></td></code>	<code><td></code> <code></td></code>	<code><td></code> <code></td></code>	<code></tr></code>
				<code></table></code>

Quelle : selfhtml

- die Eigenschaften der Zellen können relativ genau definiert werden
- sehr häufig werden auch unsichtbare Tabellen verwendet
- innerhalb der Zellen sind fast alle anderen HTML-Elemente zulässig
- *Hinweis : Die Verwendung von Tabellen zur Formatierung von Webseiten ist nicht mehr üblich und **VERSTÖSST gegen die Barrierefreiheit !***

Frames bzw. Iframes dienen zur Anzeige EXTERNER Dokumente
(Die alten Frames wurden vielfach zur Strukturierung der Webseite genutzt)

Die Verwendung ist heute umstritten, da einige Nachteile existieren:

- Es wird ein externes Layout eingebunden, welches sowohl bei der Suche (Favoriten) und bei Änderungen Probleme bereiten kann.
- Auch Copyrightfragen sind zu klären !

Alle kritischen Fragen – siehe : <https://wiki.selfhtml.org/wiki/HTML/Frames>

- Wenn, dann sollte heute statt mit Frames mit Iframes gearbeitet werden !

Beispiel einer Einbindung einer externen Website

<iframe

src="http://server.com/doc.html"

style="border: 0; width:200px; height:80px;">

sandbox="allow-scripts allow-popups allow-forms"

</iframe>

Mit dem Attribut **sandbox** können zusätzliche Sicherheitsmaßnahmen für das externen Dokument definiert werden (und ggf. auch fallweise wieder aufgehoben werden)

HTML-Formulare (besonders relevant für Webanwendungen)

- HTML-Formulare bieten eine erste, noch recht einfache Möglichkeit zum Aufbau von webbasierten Anwendungen
- Eingaben oder Selektionen in den Formularelementen werden zum Server unter Angabe eines Verarbeitungsprogramms gesendet
- Die Übertragung der Daten erfolgt ASCII-kodiert mit einem GET oder POST-http-Request

Grundstruktur von Formularen

```
<form action="http://www.firma.de/cgi-bin/feedback.pl"  
method="get">
```

```
<!-- hier sind die Formularelemente zu platzieren -->
```

```
</form>
```

HTML-Formularelemente

- Einzeilige Eingabefelder

```
<input type=text name=name size=xx maxlength=xx>
```

- Mehrzeilige Eingabefelder

```
<textarea name=Kritik rows=xx cols=xx></textarea>
```

- Auswahllisten

```
<select name="BROWSER:" size=1>
```

```
<option>Netscape 2.x </option> <option>Netscape 3.x </option> </select>
```

- Tasten (generell mindestens eine zum Absenden)

```
<input type=submit value="Abschicken">
```

```
<input type=reset value="Löschen">
```

- Radiobuttons

```
<input type=radio name="Gruppenname" value="interner Name">
```

- Optionsschalter (Checkboxes)

```
<input type=checkbox name="Gruppenname" value="interner Name">
```


Definition von einen oder mehreren Skriptbereichen :

- im Dateikopf zwischen `<head>` und `</head>` oder
- auch innerhalb des Dateikörpers `<body>` und `</body>`
- Innerhalb von Script-Bereichen können Anweisungen der verwendeten Script-Sprache notiert werden :

```
<head>
```

```
  <script> function Zeit ()
```

```
    { var Jetzt = new Date(); // .. Weiterer Code
```

```
      document.write("<h1>" + Datum + "</h1>"); }
```

```
</script> </head>
```

```
<body> <script type="text/javascript" >
```

```
    Zeit(); </script>
```

```
    <script >
```

- Unter HTML5 ist `type="text/javascript"` als Default definiert und kann entfallen.
- Das alte Attribut `language="JavaScript"` wurde als veraltet ("deprecated") eingestuft und sollte nicht mehr verwendet werden !

- HTML ist ausgerichtet auf die flexible und Hardware-unabhängige Anzeige
- genaue Formatangaben und pixelgenaue Positionierungen von Grafiken sind insbesondere für eine große Anzahl von Browserversionen schwierig bis unmöglich
- Nach einigen direkten Erweiterungen in HTML wurden die **"Cascading Style Sheets" (CSS) definiert**
- CSS stellt eine Erweiterung von HTML dar und wird in dieses eingebunden
- haben relativ große Ähnlichkeit zu bekannten Formatvorlagen
- **erste Version CSS 1.0 bereits von 1996** – meisten Browsern unterstützen 100%
- **Version 2.0 wurde bereits 1998 verabschiedet**
 - Bis 2010 gab es jedoch keine 100%ige Unterstützung durch die Browser
 - deshalb wurde 2011 CSS 2.1 mit einigen Streichung im Vergl. zu CSS 2.0 verabschiedet
 - die meisten CSS 2.1 Funktionen sollten heute durch die Browser umgesetzt werden (Tests sind bei sehr spezifischen Optionen jedoch immer noch sinnvoll)
- **aktuell Arbeit an der Versionen CSS 3.0**
 - seit 2000 auch Draft zu CSS 3.0, aber keine noch kein finaler Standard
 - CSS 3.0 wird als modulares System konzipiert und die Module werden einzelnen weiterentwickelt und tw. auch schon unterstützt (z.B. komplexe Media Queries, CSS Color Level 3, Namespaces, Selectors Level 3 ...) -> **siehe VL Responsive Design**

CSS definiert genaue Formatierungsvorschriften und kann diese festlegen

- in einzelnen HTML-Tags
- zentral am Beginn eines HTML-Dokumentes für alle folgenden Formatierungen
`<style...> ... </style>`
- in einer externen Datei, welche von HTML-Dateien referenziert wird und damit einer größeren Anzahl von Dateien ein einheitliches Aussehen gibt
`<link rel="stylesheet" type="text/css" href="formate.css">`
- falls diese 3 Arten gleichzeitig angewendet werden, wird entsprechend der obigen Reihenfolge eine Priorität von lokalen vor globalen Vorgaben beachtet
- CSS unterstützt eine Anwendung verschiedener CSS-Vorschriften für verschiedene Ausgabemedien :
`<link rel="stylesheet" media="screen" href="website.css">`
`<link rel="stylesheet" media="print" href="druck.css">`
Weitere Optionen : speech (ab CSS2.1), braille, handheld, projection, tty, tv, ...

Arten und Abarbeitungsreihenfolge von Stylesheets

Je nach Vorhandensein werden folgende Stylesheets bei der Formatierung einer HTML-Seite berücksichtigt :

- Browser-Stylesheet – immer implizit vorhanden (quasi im Quellcode des Browsers) – wird bei fehlenden CSS-Styles verwendet
- **Autoren-Stylesheet – bei Verwendung von CSS die Regel**
- Benutzer-Stylesheet – optionales Stylesheet des jeweiligen Benutzers (zwecks persönlicher Anpassung -> Sehschwäche /Farbenblindheit ?)
 - bei IE über Extras -> Allgemein-> Eingabehilfen -> Benutzer-Stylesheet
 - beim Mozilla-B. über userContent.css im Unterverzeichnis chrome des Profils
- **Priorität der Abarbeitung**
 1. Benutzer-Stylesheet mit !important
 2. Autoren-Stylesheet mit !important p { font-size:1em !important; }
 3. Autoren-Stylesheet
 4. Benutzer-Stylesheet
 5. Browser-Stylesheet

- Innerhalb von HTML-Elementen werden Formatangaben über das style-Attribut definiert:

```
<style type="text/css">  
  <!--  
  body { background-color:#000000; color:#E0E0E0 }  
  -->  
</style>
```

```
<p style="background-color:#808040; color:#D8FD02;  
font-family:'Century Schoolbook',serif; font-size:24pt; letter-spacing:3px;  
padding:40px; border:double #D8FD02 4px;"  
title="Zitat "> ... Text </p>
```

Zentrale CSS- Formatierungen für ein Dokument

- Innerhalb des <style>-Bereiches am Anfang der Datei werden durch Aufzählung der Tagnamen ohne < > die Formate definiert

```
<style type="text/css">
```

```
<!--
```

```
body { background-color:#FFFFCC;  
margin-left:100px; }
```

```
h1, h2 { font-size:48pt; color:#FF0000; font-style:italic;  
border-bottom:solid thin black; }
```

```
p,li { font-size:12pt;  
line-height:14pt;  
font-family:Helvetica,Arial,sans-serif;  
letter-spacing:0.2mm; word-spacing:0.8mm;  
color:blue; }
```

```
-->
```

```
</style>
```

Externe CSS- Formatierungen

- Analog zum <style>-Abschnitt werden in externen *.css-Dateien ohne <style> die Formatierungen definiert

```
/* DATEI: democss */
```

```
h1,h2,h3,h4,p,ul,ol,li,div,td,th,address,blockquote,nobr,b,i {  
font-family:Arial,sans-serif; }
```

```
h1 { font-size:26px; margin-bottom:18px; }
```

```
h2 { font-size:21px; margin-bottom:18px; }
```

```
h2.sh2 { font-size:18px; }
```

```
...
```

- die HTML-Elemente div und span werden sehr oft eingesetzt, da sie selbst relativ eigenschaftslos sind und damit gut beeinflussbar sind
 - div-Element erzwingt eine neue Zeile im Textfluss
 - span kann innerhalb eines Textes verwendet werden kann und erzeugt keinen neuen Absatz
- **Darstellung** kann mit Attribut display beeinflusst werden :
 - none –keine Anzeige, table / table-cell – Tabellenlayout
 - list, block , inline ...
- **Sichtbarkeit** kann mit visible gesteuert werden :
 - visible – sichtbar , hidden - unsichtbar
- collapse – zum temporären Unsichtbarmachen von Zellen etc., wird noch nicht von allen Browsern unterstützt ..)
- Verhalten des Textes kann mit float / clear gesteuert werden

- Selektoren dienen zur spezifischen Ansprache der CSS-Parameter aus externen (oder Header-) Styles innerhalb eines HTML-Dokuments
- Grundsyntax : **Selektor { Eigenschaft:Wert; } oder**
Selektor { Eigenschaft:Wert;
Eigenschaft:Wert;
... }
- Die Kombination aus Eigenschaft und Wert wird als Deklaration bezeichnet.
- allgemeine Schreibweise von Selektoren :
 - Groß- und Kleinschreibung wird bei HTML/CSS nicht beachtet, bei XHTML dagegen unterschieden !!! (Regel : -> generell klein schreiben)
 - nur (*a-z, A-Z*), Ziffern (*0-9*) und - (keine Umlaute, keine Ziffer am Anfang !)

Universal-Selektoren *

- verknüpfen jedes vorhandene Element mit den angegebenen Stylesheetangaben : * { font-size:14px; }
- werden ggf. durch spezifischere Selektoren überschrieben

Typ-Selektoren

- verknüpfen einen Elementtyp (HTML-Tag ohne <>) mit einer Formatierung : **h1 { font-size:18px; font-style:italic; }**
- durch Komma getrennte Aufzählungen sind möglich :
h1,h2,h3,h4 { font-family:Arial,sans-serif; }
- Ansprache im Dokument : normale Angabe des Tags
<h1> Überschrifttext </h1>

- für Elemente (Tags) können Klassen (Ausprägungen) definiert werden
- Klassenname kann bis auf allg. Schreibregeln frei gewählt werden :
h1.spezi { font-size:22px; font-style:italic; }
h1.klein { font-size:16px; font-style:italic; }
- auch Kombination mit Universalselektor möglich, gilt dann für alle Elemente, wird aber von spez. Klassendefinitionen überschrieben :
***.spezi { font-size:20px; }** oder auch ohne *
.spezi { font-size:20px; }
.rot {background-color:#FF0000 ; }
- Ansprache im Dokument : Angabe des Tags mit class-Attribut
<h1 class = "spezi" > Überschrifttext </h1>
<h2 class = "spezi" > Überschrifttext 2. Ebene </h2>
<h2 class = "spezi rot" > Überschrifttext 2. Ebene in rot </h2>

CSS- Selektor-Arten - Individualformate mit id

- Mit einem eindeutigen Wert für das Universalattribut id lassen sich Elementtypweit (und i.d.R. auch dateiweit) eindeutige Zuordnungen definieren :
- ID-Selektoren beginnen mit einem # oder bestehen aus *Element#id* (und gelten dann nur für das angegebene Element).

```
#bereich1 { font-size:22px; font-style:italic; }
```

```
h1#bereich2 { font-size:16px; font-style:italic; }
```

- Ansprache im Dokument : Angabe des Tags mit id-Attribut

```
<p id = "bereich1" > text ... </h1>
```

```
<h1 id = "bereich2" > Überschrifttext 1. Ebene </h2>
```

```
<h2 id = "bereich2" > Überschrifttext 2. Ebene </h2>
```

- Attributabhängige Selektoren prüfen auf Existenz oder Werte von Attributen und formatieren dann ggf. entsprechend.

Optionen :

- **Existenz** von Attr. : `p[align] { color:red; }`
 - Tags mit align-Attr. werden rot formatiert
- **Wert** von Attributen : `p[align=center] { color:blue; }`
- **Vergleich** auf Textinhalt (= substring-Suche) `E[attr ~= "Wert"]`
`p[zeile ~= "gerade"] { color:blue; }`
`<p zeile="uebersicht gerade Farben"> text </p>`
- **genauer Textinhalt** (ggf. gefolgt von -) `E[attr |= "Wert"]`
`*[lang|=en] { background-color:#FF0000; }`
`<p lang="en-US"> ... english text </p>`

CSS- Pseudoformate

- Pseudoformate verknüpfen Formate mit einem Element, das einen bestimmten Zustand hat, oder Formate mit Elementteil

Pseudoklassen in CSS 1 + 2 (definiert als tag : Pseudoklassenname):

- :link Nichtbesucher Hyperlink.
- :visited Besucher Hyperlink
- :active Aktiver Hyperlink (Wenn der User mit der Maus anklickt.)
- :hover Mouseover-Effekt. (Mauszeiger über Hyperlink)
- :focus - Element mit Fokus hat. z.B. Texteingabefelder
- :first-child - erstes Kind eines anderen Elements
- Folgende Pseudoelemente sind in CSS 1 + 2 festgelegt:
- :first-line - erstes Zeile in einem Absatz.
- :first-letter - erstes Zeichen in einer Zeile.
- :before - fügt vor einem Element einen beliebigen Inhalt ein
- :after - fügt nach einem Element einen beliebigen Inhalt ein

CSS-Formatierung verschachtelter HTML-Elemente

- Formatierung in Abhängigkeit von der Tag-Hierarchie :
h1 { color:red; }
h1 i { color:blue; font-style:normal; }
<p> **hh** <i> rrr </i></p> <h1>**t..** <i> ii </i></h1>
- Kind-Selektor (analog zu vorher, jedoch GENAU eine Ebene tiefer)
h1 > i { color:blue; font-style:normal; }
- Tiefere Vererbungshierarchie mit Selektor * Selektor – 2. Ebene etc.
h1 * i { color:blue; font-style:normal; }
- Nachbarschaftsselector - wird aktiviert bei direkter Nachfolge auf angegebenes Tag (sinnvoll zur Abstandsjustierung)
div + p { margin-top:100px; }
- Kombination aller Formatierungsoptionen möglich :
div p *[href] {color:#ff0000;}
-> rote Schrift, wenn p-Tag in div-Tag mit enthaltenem href in Subtags

Kaskadierung von CSS (Prioritätsberechnung)

Generell setzt sich der spezifischste Style-Selektor durch, d.h. lokale vor globalen style-Definitionen, danach Berechnung und Priorisierung nach folgender Vorschrift :

- A enthält den Wert 1, bei direkter Formatierung mit style-Attribut
- B entspricht der Anzahl der selektierten Seite ID-Attribute (#id)
- C entspricht der Anzahl der selektierten anderen Attribute (z.B. Seite Klassen) und Seite Pseudoklassen (.klasse, :pseudoklasse)
- D entspricht der Anzahl der selektierten Elementnamen (e) und Seite Pseudoelemente (:pseudoelement)
- **Die Bewertung geht immer von A-D nur jeweils nach Spaltenwert und entscheidet bei Gleichheit anhand der Reihenfolge.**

Selektor	A	B	C	D
style="..." (HTML)	1	0	0	0
#nav a.xy	0	1	1	1
#nav li a	0	1	0	2
#nav a	0	1	0	1
#nav	0	1	0	0
ul li .xy	0	0	1	2
a:link	0	0	1	1
a.xy	0	0	1	1
ul[id="nav"]	0	0	1	1
*.xy	0	0	1	0
li a	0	0	0	2
a:first-line	0	0	0	2
a	0	0	0	1

Quelle : <http://de.selfhtml.org/css/formate/kaskade.htm>

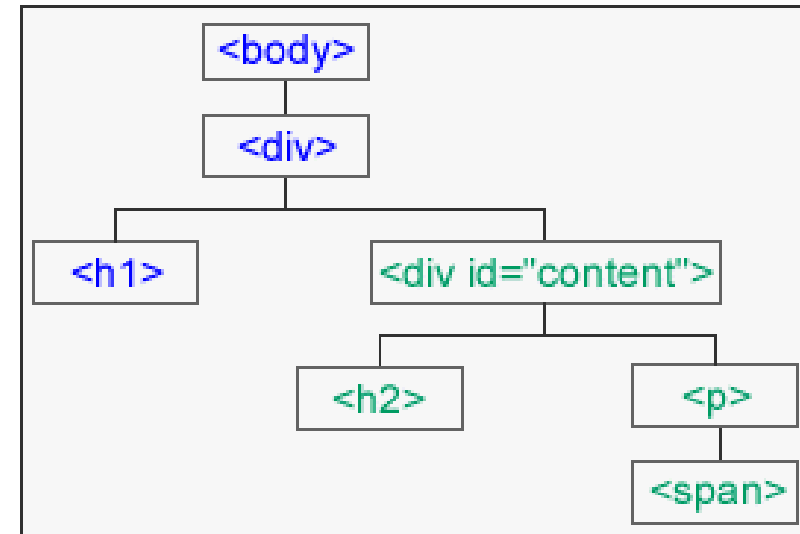
Der hierarchische Aufbau von Html-Dateien macht eine teilweise Vererbung von Eigenschaften auf untergeordnete Elemente sinnvoll.

- Falls keine genaueren Spezifikationen erfolgen, werden Farb- und Schriftinformationen vererbt.

```
body { background-color:#f7f7f7;  
        color:blue; }  
#content { color:green; }
```

- Einige Attribute wie border-Formatierungen werden nicht standardmäßig vererbt.
- Falls es dennoch notwendig ist, kann mit **Inherit** eine Vererbung erzwungen.

```
#div1 { border:1px solid red; }  
#div2 { border:inherit; margin:10px; }
```



Maßeinheit :

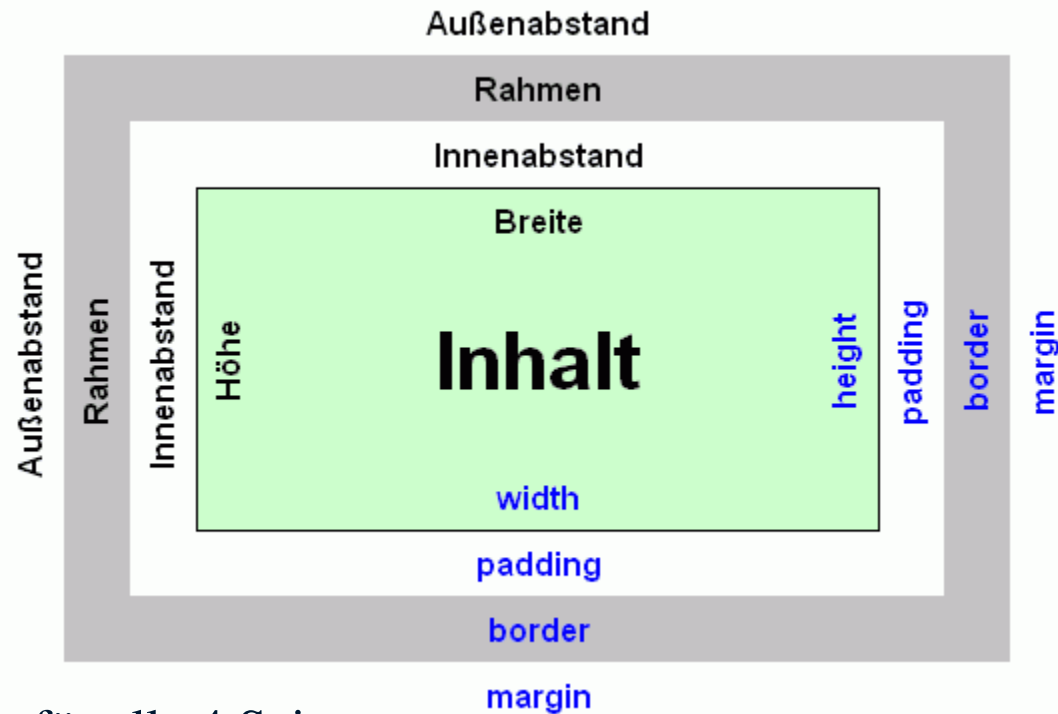
Beispiel :

- pt absolut - Punkt, 1 Punkt entspricht 1/72 Inches **font-size:12pt;**
- pc absolut - Pica, entspricht 12 Punkten. **font-size:12pt;**
- in = inch, mm = Millimeter, cm = Zentimeter - alles absolut
- px absolut/relativ - Pixel = f(Pixeldichte des Ausgabegeräts), relativ bezogen auf vers. Ausgabegeräte, absolut für ein Gerät
- em relativ - bezogen auf Schriftgröße des Elements, bei Def. der Schrift selbst -> Bezug auf Eltern-Schriftgröße **font-size :1.5em;**
- ex relativ - bezogen auf Höhe des Kleinbuchstaben x im Element. Ausnahme: bei Angabe font-size selbst, dann wieder bezogen auf die Höhe des Kleinbuchstaben x im Elternelement. **font-size:1.3ex;**
- % relativ - in Prozent je nach CSS-Eigenschaft relativ zur elementeigenen Größe, oder zu der des Elternelements, oder zu einem allgemeineren Kontext (Fenster)

Probleme : . – verwenden, Vorsicht bei relativen Kettenmaßen

Das Box - Modell

- Für jedes Element einer Seite wird nach den CSS-Regeln ein rechteckiger Bereich reserviert, der in dem sog. Boxmodell beschrieben ist.



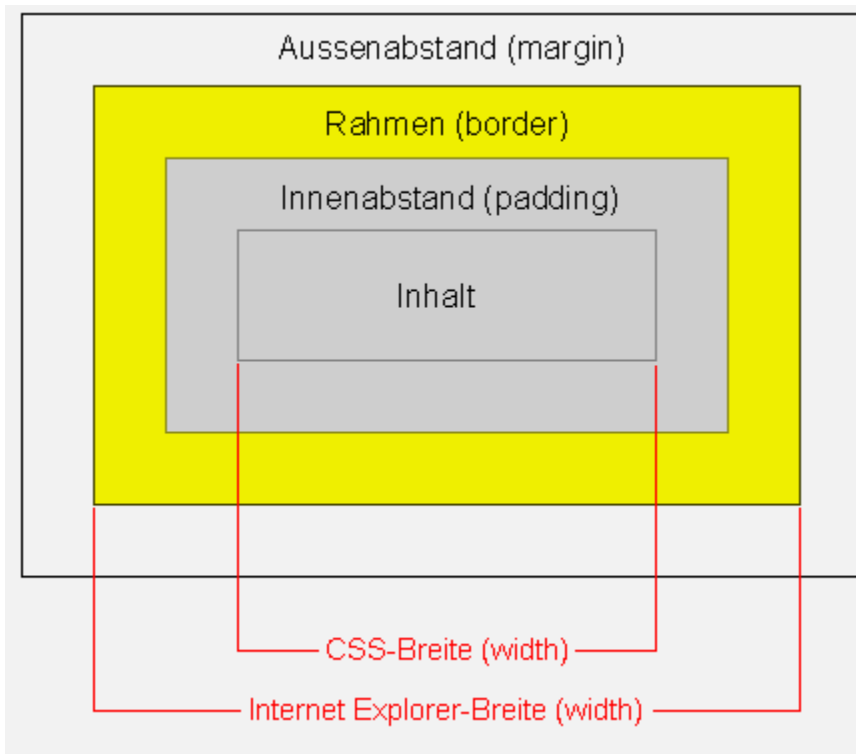
Jeder Parameter existiert für alle 4 Seiten :

- padding-left, padding-right, padding-top, padding-bottom,
- border-left, border-right, border-top, border-bottom,
- margin-left, margin-right, margin-top, margin-bottom

Falls die Abmessungen gleich sind, kann auch nur mit margin, border und padding gearbeitet werden.

Box-Modell-Fehler des Internet Explorer

- Als "Box Model Bug" wird der Fehler in älteren Windows-Versionen des Internet Explorer (einschließlich 5.5) bezeichnet, die Innenabstände und Rahmenstärken entgegen der Spezifikation nicht zur Gesamtbreite zu addieren - dies ist nur beim Außenabstand korrekt der Fall.



Lösungen :

- Wenn kein Rahmen und keine Hintergrundfarbe benötigt werden, sollte statt padding mit margin gearbeitet werden
- spezielle Browser-Weichen und Browser-Hacks zur browserspezifischen Formatierung

- "Box Model Bug" bis Internet Explorer 5.5
- Microsoft führte mit Version 6 des Browsers eine Umschaltmöglichkeit zwischen falschem und der korrekten Verhalten ein :
 - Dokumente, die als syntaktisch strenge (X)HTML-Variante deklariert sind, werden vom Internet Explorer 6 unter Berücksichtigung des korrekten Box-Modells dargestellt - im sogenannten "standard mode".
`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 strict //EN"
"http://www.w3.org/TR/html4/strict.dtd">`
 - Auf Seiten, die keine oder eine weniger strenge Deklaration aufweisen, wird das fehlerhafte Microsoft-Box-Modell angewandt, um Abwärtskompatibilität zu gewährleisten. Dieser Modus wird als "quirks mode" bezeichnet.
`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">`
 - Auch das Fehlen der Adresse, unter welcher die Seite Dokumenttyp-Definition (DTD) beim W3-Konsortium abgerufen werden kann, bewirkt bei HTML 4.01 teilweise einen Rückfall in den Quirks-Modus.
- Weitere Option : spezielle Browserhacks (siehe http://de.selfhtml.org/css/formate/box_modell.htm)

- mit CSS-basierten Layouts ist eine effektive (auch arbeitsteilige) Entwicklung von Content und Layout möglich

Vorteile :

- weniger Code (bessere Performance) als bei Tabellen (... - 40%)
- leichte Unterstützung unterschiedlicher Ausgabemedien

Mögliche Probleme

- Genauer Test des Verhaltens der verschiedenen Browser notwendig – auch mit neuen Versionen !
- Empfehlungen für einen guten CSS-Stil (auch Projektorg.)
http://wiki.selfhtml.org/wiki/CSS/Anwendung_und_Praxis/Guter_CSS-Stil

Weitere Details und Beispiele unter

<http://wiki.selfhtml.org/wiki/CSS>