



3. Zeichen und Schrift

- 3.1 Medien Zeichen, Text, Schrift
- 3.2 Mikro-Typografie: Zeichensätze
- 3.3 Makro-Typografie: Gestalten mit Schrift
- 3.4 Hypertext und HTML
(Fortsetzung) 

- Allgemeines
- Textstrukturierung
- Tabellen
- Cascading Style Sheets 
- Strukturierte Seiten
- Medieneinbettung

Weitere Informationen: <http://de.selfhtml.org/>

Selbstdefinierte Stilklassen

- Eigene Stilklassen (außer den HTML-Elementen)
 - können frei definiert und verwandt werden
- Deklaration
 - bei der Stildefinition (mit dem Namen vorangestelltem Punkt)
 - z.B. `.navigation {font-size:16pt; color:blue;}`
- Anwendung
 - mit dem universellen `class`-Attribut aller HTML-Tags
 - z.B. `<li class="navigation">Home`

Blockweise Formatierung mit CSS

- Ganze Textbereiche einheitlich formatieren
- Verwendung des *Inline-Elements* ` ... `
 - Keinerlei Effekt auf die Dokumentstruktur
 - Kann Text oder andere Inline-Elemente enthalten
 - Wendet angegebene Stilangaben auf den eingeschlossenen Textbereich an
- Verwendung des *allgemeinen Blockelements* `<div> ... </div>`
 - Kann Text oder andere Blockelemente enthalten, z.B. auch Grafiken
 - Weitergabe der angegebenen Stilangaben zu allen enthaltenen HTML-Elementen
 - Kann mit der CSS-Eigenschaft `position` absolut positioniert werden
 - Kann mit Skripten ein- und ausgeblendet werden
 - Anmerkung: Ersetzt das alte Netscape-spezifische Element "layer"

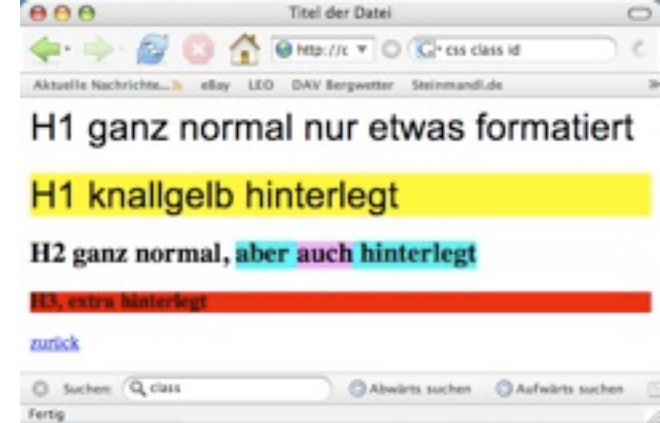
Beispiel zu selbstdefinierten Stilklassen

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
<html>
  <head>
    <title>Beispiel zu CSS: Selbstdefinierte Klassen</title>
    <link rel="stylesheet" type="text/css" href="styles1.css">
  </head>
  <body>
    <p>Dies ist ein ganz normaler Absatz ohne spezielle
    Formatierung. </p>
    <p class="merksatz">Dies ist ein Merksatz,
    speziell formatiert mit Hilfe von CSS.</p>
    <p>Dies ist wieder ein ganz normaler Absatz.</p>
    <span class="programm">
      <p>Dies sind zwei aufeinander folgende Absätze,
      die speziell formatiert werden.</p>
      <p>Dies ist der zweite solche Absatz.</p>
    </span>
  </body>
</html>
```

stylesclass.html

Klassen: Komplexeres Beispiel

```
<html>
<head>
<title>Titel der Datei</title>
<style type="text/css">
h1 {font-family:Arial,sans-serif; font-size:2em; font-weight:normal;}
h1.hinterlegt { background-color:yellow }
*.hinterlegt { background-color:cyan}
.extra { background-color:magenta}
.extra.hinterlegt { background-color:red}
</style>
</head>
<body>
<h1>H1 ganz normal nur etwas formatiert</h1>
<h1 class="hinterlegt">H1 knallgelb hinterlegt</h1>
<h2>H2 ganz normal, <span class="hinterlegt"> aber <b
class="extra">auch</b> hinterlegt</span></h2>
<h3 class="extra hinterlegt">H3, extra hinterlegt</h3>
</body>
</html>
```





stylescomplex.html

ID statt class?

- Im HTML file: `<h1 ID="blau">Überschrift </h1>`
- Im CSS file: `#blau {color : blue;}`
- Die Benutzung einer ID bietet einige Vorteile:
 - ID kann als Sprungziel für Hyperlinks verwendet werden.
 - IDs können mit Javascript angesprochen werden, mit `getElementById()`
 - IDs überstimmen Klassen.
- Nachteile von IDs:
 - Eine ID darf auf einer Seite nur einmal verwendet werden. (Auch wenn so mancher Browser das anders sieht...)
 - IDs können nicht wie Klassen kombiniert werden.
- Klassen und IDs können gemeinsam genutzt werden. Beispiel:
`<h1 class="bigblue" ID="T1">Überschrift </h1>`

3. Zeichen und Schrift

- 3.1 Medien Zeichen, Text, Schrift
- 3.2 Mikro-Typografie: Zeichensätze
- 3.3 Makro-Typografie: Gestalten mit Schrift
- 3.4 Hypertext und HTML
(Fortsetzung) 

- Allgemeines
- Textstrukturierung
- Cascading Style Sheets
- Tabellen
- Strukturierte Seiten 
- Medieneinbettung

Weitere Informationen: <http://de.selfhtml.org/>

Strukturierte Seiten (Layout)

- Moderne Webseiten haben ein klar definiertes Layout (Satzspiegel)
 - Bestandteile mit verschiedener Funktion (z.B. Kopf, Navigation, Hauptteil, Fußzeile)
 - Freies 2-dimensionales Layout (oft nebeneinander platzierte Einheiten)
- Realisierungsmöglichkeiten:
 - Framesets (Element `<frame>`): Veraltet und nicht empfohlen
 - Tabellen:
 - » Weit verbreitet und effektiv, aber auch nicht mehr empfohlen
 - » Keine Trennung von logischer Struktur und Layout
 - Divisions (Element `<div>`):
 - » Elegante moderne Lösung zusammen mit CSS
 - » Dominanz des `<div>`-Elements ("Divitis")
 - HTML5-Strukturelemente und CSS:
 - » Derzeit beste Entkopplung von Struktur und Layout

Definition der logischen Struktur

- Welche separaten Bereiche enthält unsere Seite?
 - Möglichst übergreifend über alle Seiten eines Webauftritts
- Beispiel:
 - Kopfbereich:
 - » Für alle Seiten gleich
 - Navigationsbereich
 - » Für alle Seiten gleich; enthält Liste von Einträgen (Links)
 - Hauptteil:
 - » Soll Liste von Einträgen (Artikeln) enthalten
 - Fußzeile:
 - » Für alle Seiten gleich; kurzer Text (Impressum)

Logische Struktur in HTML 4

```
<body>
  <div id="header">
    <h1>Structured Page (HTML4)</h1>
  </div>
  <div id="nav">
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </div>
  <div id="main">
    <h1>This is the main content area of the page.</h1>
    <p>This is the first content article.</p>
    <p>This is the second content article.</p>
  </div>
  <div id="footer">
    <p>Heinrich Hußmann, LMU, 2010</p>
  </div>
</body>
```

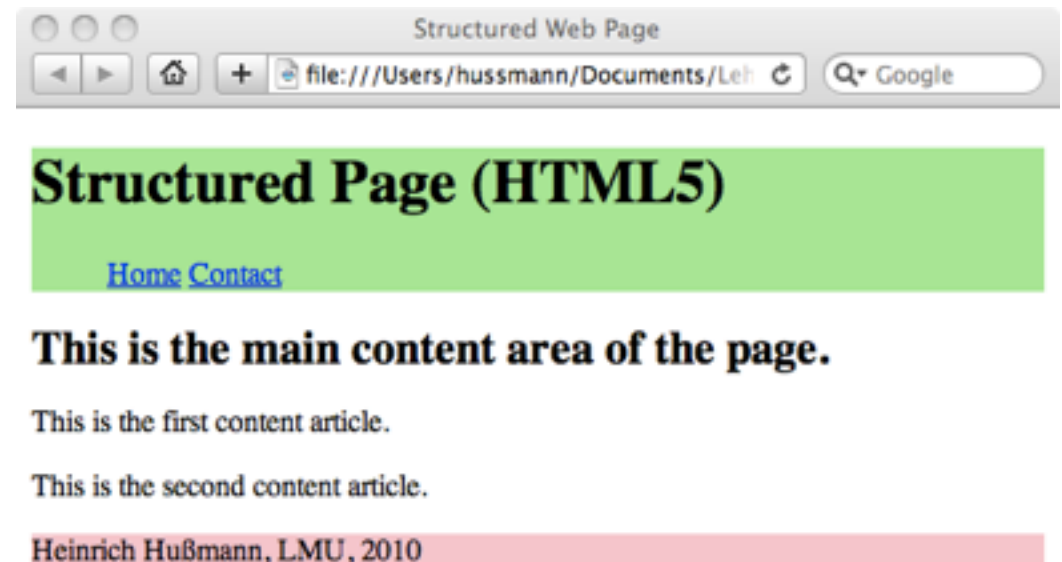
Logische Struktur in HTML5

```
<body>
  <header>
    <h1>Structured Page (HTML5)</h1>
  </header>
  <nav>
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </nav>
  <section id="main">
    <h1>This is the main content area of the page.</h1>
    <article>
      <p>This is the first content article.</p>
    </article>
    <article>
      <p>This is the second content article.</p>
    </article>
  </section>
  <footer>
    <p>Heinrich Huzlig;mann, LMU, 2010</p>
  </footer>
</body>
```

Einfaches Vertikal-Layout mit CSS

- CSS-Stylesheet zum Beispiel:

```
<style>
  header {
    background-color:lightgreen;
  }
  footer {
    background-color:pink;
  }
  nav li {
    display:inline;
  }
  header h1 {
    font-size:2em;
  }
  #main h1 {
    font-size:1.5em;
  }
</style>
```

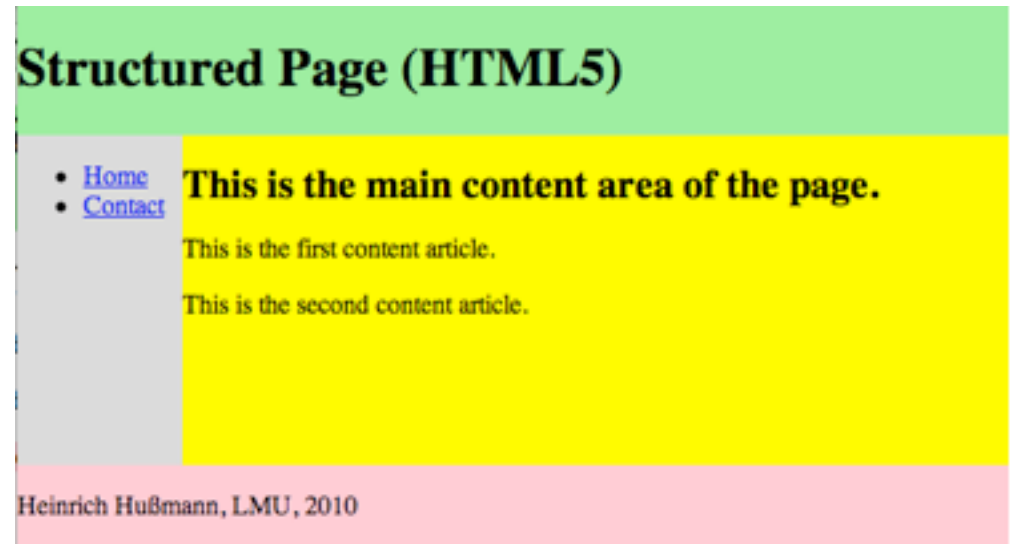


html5struct_vert.html

Mehrspalten-Layout mit CSS (feste Größen)

```
header {
    background-color:lightgreen;
    height:80px;
    width:600px;
    position:absolute;
    left:0px;
    top:0px;
}
nav {
    height:200px;
    width:100px;
    background-color:lightgrey;
    position:absolute;
    top:80px;
    left:0px;
}
#main {
    background-color:yellow;
    position:absolute;
    height:200px;
    width:500px;
    left:100px;
    top:80px;
}

footer {
    background-color:pink;
    width:600px;
    position:absolute;
    top:280px;
    left:0px;
}
header h1 {
    font-size:2em;
}
#main h1 {
    font-size:1.5em;
}
```



html5struct_fixed.html

Mehrspalten-Layout mit CSS ("flüssig")

```
header {
  background-color:lightgreen;
  height:20%;
  width:100%;
  position:absolute;
  left:0%;
  top:0%;
}
nav {
  height:70%;
  width:10%;
  background-color:lightgrey;
  position:absolute;
  top:20%;
  left:0%;
}
#main {
  background-color:yellow;
  position:absolute;
  height:70%;
  width:90%;
  left:10%;
  top:20%;
}
```

```
footer {
  background-color:pink;
  height:10%;
  width:100%;
  position:absolute;
  top:90%;
  left:10%;
}...
```




html5struct_liquid.html


Technische Details (Auswahl)

- HTML5-Elemente benutzen mit alten HTML-4-konformen Browsern (v.a. Internet Explorer):
 - Unbekannte Elemente via JavaScript bekannt machen
 - `document.createElement("article");` etc.
- Zentrale Dateien für ganze Web Sites:
 - Unproblematisch für Style Sheets (css-Dateien)
 - Nicht-trivial für feste Bestandteile des HTML-Codes (z.B. Navigation)
- "Erbschaft" aus den veralteten Frame Sets:
 - Werte für "target"-Attribut in Links (Anchor-Tag `<a>`):
 - » `_blank` = Verweis in neuem Fenster öffnen
 - » `_self` = Verweis im gleichen Fenster öffnen
 - Beispiel:

Hier ist ein
`Link`,
der ein neues Fenster öffnet.

3. Zeichen und Schrift

- 3.1 Medien Zeichen, Text, Schrift
- 3.2 Mikro-Typografie: Zeichensätze
- 3.3 Makro-Typografie: Gestalten mit Schrift
- 3.4 Hypertext und HTML
(Fortsetzung) 

- Allgemeines
- Textstrukturierung
- Cascading Style Sheets
- Tabellen
- Framesets
- Medieneinbettung 

Weitere Informationen: <http://de.selfhtml.org>

Integration von Bildern

- Bilder einbinden mit ``
- Attribut `src` gibt Quelle an (auch von anderen Servern möglich)
 - Achtung Copyright-Fragen!
- Größenangaben mit `width` und `height`
 - Bei Angabe beider Werte Verzerrung möglich
- Bilder können auch als Inhalt eines Verweises vorkommen
 - z.B. grafische Navigationsleisten

```
<html> ...  
  <body>  
    <h1>Ein JPEG-Bild des Eiffelturms</h1>  
  
    <p>  
    </p>  
  
  </body>  
</html>
```

Integration anderer Dateien

- Prinzipiell alle Dateien einbettbar
 - mit dem `<object>`-Tag (standardkonform)
 - als Hyperlinks
- Leider Behandlung von Medien in Browsern uneinheitlich!
- HTML5 versucht Vereinheitlichung zu erreichen.

```
<audio src="xyz.ogg" autoplay>  
  Your browser does not support the <code>audio</code> element.  
</audio>
```

- HTML5 Video:

```
<video controls>  
  Your browser does not support the <code>video</code> element.  
  <source src="big_buck_bunny_480p_stereo.ogg" type="video/ogg">  
  <source src="big_buck_bunny_480p_surround-fix.avi">  
</video>
```

MIME

- MIME = Multipurpose Internet Mail Extensions
 - In HTML mit dem `type`-Attribut an vielen Stellen angebar (z.B. `<link>`, `<object>`)
 - Erleichtert dem Browser (bzw. seinem Benutzer) die Entscheidung, wie Dateien zu behandeln sind
 - Jeder Browser führt eine Liste der akzeptierten MIME-Extensions und Regeln für die Behandlung (z.B. speichern, Programm aufrufen)
 - Liste siehe <http://www.iana.org/assignments/media-types>
- Syntax:
 - Medientyp / Untertyp*
 - Medientypen: text, image, video, audio, application, ...
 - Untertypen, die auf dem Server auszuführen sind, beginnen meist mit x-
 - Hersteller- (*vendor*-)spezifische Untertypen im speziellen Unterbaum "vnd."

Design vs. Flexibilität

- Aus gestalterischer Motivation werden oft folgende Konzepte verwendet:
 - Feste Formatvorgaben für die Seite
 - Spezial-Schriften
 - Feste Schriftgrößen
 - Frames
 - Aufwändige grafische Elemente
- Die maximale Flexibilität in der Verwendung spricht für:
 - Flexible Fenstergröße („liquid design“)
 - Unabhängigkeit von Schriftwahl
 - Vom Benutzer bestimmbare Schriftgrößen
 - Keine oder sehr eingeschränkte Benutzung von Frames
 - Kleine, sparsame grafische Elemente

Barrierefreiheit von Webseiten

- Gesellschaftliche Funktionen des WWW:
 - Wesentliches Medium für staatliche Informationsdienste und Bürgerservice
 - Tendenziell besonders leicht zugänglich für Personen, die andere Zugänge nur schwer nutzen können (z.B. Behinderte)
 - Generell ein demokratisches Medium, das für alle offen sein soll
- **Was passiert mit Menschen, die mit Einschränkungen ihrer Nutzungsmöglichkeiten leben müssen?**
 - Äquivalent zu "barrierefreien" Zugängen zu Gebäuden
- Nutzung des WWW bei eingeschränkten Wahrnehmungs- und Aktionsmöglichkeiten
 - Seh- oder Hörbehinderung
 - Leseschwäche, Aufnahmeschwäche, Lernschwäche
 - Einschränkungen bei der Benutzung von „zeigenden“ Eingabegeräten

Barrierefreiheit von Webseiten

- Wichtigste Richtlinie:
 - Web Accessibility Initiative (<http://www.w3.org/WAI/>)
 - Übernommen in vielen nationalen Regelungen, z.B. BITV in Deutschland
 - » "Barrierefreie Informationstechnik-Verordnung"
 - » Seit 2006 verbindlich für alle öffentlich zugänglichen Webauftritte des Bundes
 - » Seit 2007 auch in Bayern verbindlich für staatliche Webauftritte (BayBITV)
- Beispiele für Regeln zur Barrierefreiheit:
 - Ergänzung grafischer Information durch textuelle Beschreibung
 - » Auch bei zeitabhängigen Medien (Untertitel zu Video)
 - Benutzbarkeit mit Tastatur (d.h. auch mit Spracheingabe)
 - Orientierung durch klare Struktur und kleine Textblöcke erleichtern
 - Hoher Kontrast zwischen Vordergrund und Hintergrund
 - Auslösung epileptischer Anfälle durch blinkende Inhalte verhindern