

HTML und CSS Kurs

Christoph Peter Neumann

Student der Informatik, FAU Erlangen-Nürnberg

Seminar Fit@Webdesign, THA Gummersbach,
17.-19. September 2004

Part I

HTML und CSS Kurs (Einstieg)

Agenda

1 Einleitung

- Benötigte Programme
- (X)HTML-Grundgerüst
- Doctype

2 Grundelemente

- Text
- Bilder
- CSS-Einführung
- Links
- Anchors

3 Übungen

- HTML-Kit
- Erste Seite
- Vorschau
- Inhalte
- Shortcuts
- Plug-ins

Benötigte Programme

Praktisch keine Voraussetzungen!

HTML ist reiner (ASCII-)Text. Wenn man alle Elemente von HTML und CSS mehr oder weniger auswendig beherrscht, dann reicht der simpelste Texteditor wie notepad aus.

ASCII: **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

[ASCII beinhaltet keinerlei Sonderzeichen, wie dt. Umlaute. Ebenso die Codierung von asiat., hebrä. oder arab. Zeichen erfordert (eine später diskutierte) Vorgehensweise.]

Trotzdem kann man sich das Editieren durch einige Programme einfacher machen (hier nur Windows-Freeware!):

- HTML-Kit – HTML Source-code Editor
- Nvu – WYSIWYG
- TopStyle – CSS Editor
- Mozilla ActiveX Control – Gecko als Browser-preview
- FTP-Uploader – für späteren Upload ins Internet
- HTML-Kit Plug-ins: includeHTML , w3c offline Referenzen

(X)HTML-Grundgerüst

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Titeltext der Seite</title>
</head>
<body>
... TEXT ...
</body>
</html>
```

Das `html`-Element beinhaltet zwei Elemente: das `head`- und das `body`-Element. Das `head`-Element beinhaltet Metadaten über die Seite. Das `body`-Element enthält den eigentlichen (vom Browser angezeigten) Inhalt.

Doctype – von HTML und XML zu XHTML+CSS

HTML als Auszeichnungssprache brauchte (wie jede Programmiersprache auch!) 10 Jahre, von 1990 bis 2000, um über ihre **Kinderkrankheiten** hinwegzukommen. Im Jahr 1999 endete mit **HTML 4.01** der Sprachzweig, der **Inhalt/Struktur** und **Präsentation** vermischte.

Im Jahr 2000 wurde **XHTML** in Version 1.0 standardisiert. XHTML enthält *ausschließlich Inhalt und Struktur*, entsprechend dem **XML**-Vorbild. Die *Präsentations*-Elemente sind vollständig durch die Cascading Style Sheet (**CSS**) Sprache ersetzt worden.

Vorteil

Die strikte Trennung erlaubt die einfachere Verarbeitung und Erstellung von HTML.

(Browser-Implementierung, Konformität in der Anzeige, ...)

Doctype(2)

Der Vollständigkeit halber soll angegeben werden wie ein **HTML 4.01 Dokument** deklariert wird:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>...
```

Die **Verknüpfung von (X)HTML und CSS** findet im html-Dokument statt:

```
<head> ...
<link rel="stylesheet" type="text/css"
      href="yourCSSfile.css" media="all" />
</head>
```

(Der Laie möge beachten, dass dies auch auf viele andere Arten möglich wäre, z.B. in Form einer dritten Datei, die als Katalog-Datei die Zuweisung zwischen HTML- und CSS-Dateien beschreiben könnte.)

Doctype(3) – XHTML als XML

Ausserdem macht alleine die Doctype-Deklaration eines XHTML-Dokuments dieses Dokument noch nicht zu **gültigem XML**. Dies wird erst durch folgenden Zusatz gewährleistet:

```
<?xml version="1.0"?>  
<!DOCTYPE ... XHTML ...
```

Allerdings sollte im Moment darauf verzichtet werden, weil:

- ... **unvollständiger Browser-Support** ...
- ... HTML-Engine versus XML-Engine ...

Text – Absätze und Zeilenumbrüche

Normaler Fließtext ist meist in verschiedene **Absätze** unterteilt. In html deklariert das **paragraph**-Tag einen Absatz:

```
<p>
... ganz viel
    Absatz Text ...
</p>
```

Browser zeigen Absätze meist durch eine Zeilenabstand etwas **abgesetzt** von anderen Elementen (wie z.B. anderen Absätzen) an. Möchte man (z.B. innerhalb eines Absatzes) einen **Zeilenumbruch** – der sich im Gegensatz zu einem neuen Absatz nicht absetzt – dann benötigt man das **break**-Tag:

```
<br />
```

Diese spezielle Schreibweise mit dem “/” am Ende des Tags, erspart ein Ende-Tag zu schreiben. Es wird immer dann eingesetzt, wenn ein Tag für sich selbst steht und keinen Text umschließt. (z.B. auch Bilder: **img**-Tag → siehe später)

Text(2) – Überschriften

Überschriften (= headings) gibt es in sechs Stufen:

```
<h1>Überschrift der Stufe 1</h1>
```

...

```
<h6>Überschrift der Stufe 6</h6>
```

Diese werden von Browsern meist in unterschiedlicher Größe dargestellt.

Text(3) – Zeichenkodierung

Viele Zeichen sind in der ASCII-Kodierung nicht enthalten, z.B. deutsche Umlaute. Daher muss man sie mit einer speziellen ASCII-codierten Zeichenkette umschreiben.

Beispiele

Das geschützte Leerzeichen (), die Umlaute ä (ä), ö (ö), ü (ü), das scharfe B (ß), das Euro Zeichen € (€) und das Copyright Zeichen © (©).

Nicht auswendig merken!

Mit **HTML Tidy**, integriert in HTML-Kit, läßt sich die Zeichentransformation automatisieren. Es dürfen also deutsche Zeichen eingegeben werden!

Bilder

Bilder können durch einen Verweis auf die Bild-Datei "eingefügt" werden. Dies geschieht mit dem **image**-Tag:

```

```

Das `src=""`-Attribut gibt den **Dateinamen** an.

Da das Bild erst durch den Browser eingefügt wird, und da es auch nicht-graphische Browser gibt (z.B. für Behinderte oder auch Linux-User ;-), sollte *immer* mit dem `alt=""`-Attribut ein **ersatzweiser Text** angegeben werden.

CSS-Einführung – Einbindung oder Einbettung in XHTML

Einleitung:

- CSS ist eine **unmittelbare Ergänzung** zu (X)HTML
- CSS dient zur Definition von **Formateigenschaften** einzelner html-Tags
- Möglichkeit: Formate **zentral** zu **verwalten** – spart Kodierarbeit und macht die HTML-Dateien kleiner
- Aktuelle Version: **CSS 2.0**

Drei Orte CSS Auszeichnungen zu schreiben:

- 1 In externer Datei (referenziert im Header)
- 2 Im html-Header
- 3 Tag-lokal (im style=""-Attribut eines Tags)

CSS-Einführung(2) – CSS in externer Datei

Im [html-Header](#) wird die externe CSS Datei referenziert:

```
<head> ...  
<link rel="stylesheet" type="text/css"  
      href="yourCSSfile.css" media="all" />  
</head>
```

In der [yourCSSfile.css](#) Datei steht dann einfach:

```
body {  
  font-family: Arial , sans-serif ;  
  color: #333333 ;  
  background: White ;  
}
```

CSS-Einführung(3) – Im Header eingebettetes CSS

CSS kann für jede html-Datei einzeln definiert werden:

```
<head> ...  
  <style type="text/css">  
  <!--  
    body{  
      font-family: Arial, sans-serif;  
      color: #333333;  
      background: White; }  
  -->  
</style> ...  
</head>
```

CSS-Einführung(4) – Tag-lokales CSS

Im `style=""`-Attribut eines jeden Tags können lokale CSS Auszeichnungen beschrieben werden:

```
<body style="font-family: Arial , sans-serif;  
           color: #333333; background: White;" >  
    ...  
</body>
```

CSS-Einführung(5) – CSS-Selektoren

Die Formatierung durch CSS kann anhand von drei Kategorien vorgenommen werden:

- Tag-Namen
- Klassen (→ html class=""-Attribut)
- Identifikatoren (→ html id=""-Attribut)

Tags derselben Klasse können mehrfach in einem html-Dokument vorkommen. Ein Tag mit einer spez. ID darf nur genau einmal in einem html-Dokument existieren.

Vererbung

CSS Formatierung eines übergeordneten Tags werden auf eingeschlossene Tags vererbt.

Zum Beispiel gilt die "font"-Angabe des body-Tags für alle in body eingeschlossenen Tags – solange nicht eine Tag-spezifischere CSS-Formatierung dies "überschreibt".

CSS-Einführung(6) – CSS-Selektoren(2)

Bsp: Formate für HTML-Tags

p { ... }		
p, i { ... }	<p>YES<p>	<i>YES</i>
p i { ... }	<p>NO<i>YES</i><p>	<i>NO</i>

Bsp: Formate auf Basis der html-Klasse

```
.classNameBeliebig { ... }  
*.classNameBeliebig { ... }  
p.classNameBeliebig { ... }
```

Bsp: Formate auf Basis einer html-ID

```
#idNameBeliebig { ... }  
*#idNameBeliebig { ... }  
p#idNameBeliebig { ... }
```

CSS-Einführung(7) – CSS-Selektoren(3) advanced

Es gibt noch weitere Möglichkeiten für **verschachtelte HTML-Elemente**:

Genau eine Ebene unterhalb:

```
div > p { ... }      <div> <p>YES</p> <p>YES</p> </div>
```

Genau zwei Ebenen unterhalb:

```
div * i { ... }      <div> <i>NO</i> <p>..<i>YES</i>..</p> </div>
```

Elemente **unmittelbar** auf ein Element **folgend**:

```
div + p { ... }      <div> <p>YES</p> <p>NO</p> </div>
```

“+” wird von den Browsern (leider) noch nicht gut unterstützt!

CSS-Einführung(8) – Ausrichtung von Bildern

Bilder kann man durch CSS verschieden **ausrichten**:

```

```

- **“float:”** sorgt für eine “Textumfließung” und richtet das Bild am rechten oder linken Rand aus. (Nur nachfolgender Text wird zur Umfließung verwendet!)
- **“padding:”** ist der Rand nach aussen
- **“margin:”** ist der Rand nach innen

CSS-Einführung(9) – Ausrichtung von Bildern(2)

Man kann Bilder auch **pixelgenau** positionieren:

```

```

```
img#test {
  position: absolute;
  top: 100px;
  left: 300px;
}
```

- “top:” und “left:” sind Abstandsangaben (Es gibt auch “bottom:” und “right:”)
- Durch “position: **absolute**” gelten die Abstände immer relativ zur gesamten html-Seite.

CSS-Einführung(10) – Ausrichtung von Bildern(3)

- Statt “absolute” gibt es z.B. auch “**relative**”: Wenn das Bild z.B. innerhalb eines speziellen Bereichs (div-Tag → später!) steht, der eine gewisse Position einnimmt, dann wird bei “relative” das Bild relativ zum umgebenden Bereich positioniert
- Neben “absolute” und “relative” gibt es auch “**fixed**”. Dies wirkt ähnlich zu “absolute”. Doch während “absolute” zur ganzen Seite hin positioniert, wirkt “fixed” bezüglich des Browser-Fensters! D.h. beim Scrollen des Inhalt wird das “position: fixed;”-positionierte Element **nicht gescrollt!**

Links → anchor-Tag mit href=

Das eine Elemente durch das sich HTML im wesentlichen definiert ist der **HyperLink**:

```
<a href="http://www.8ung.at/chr15t0ph/">TEXT der  
nun verlink ist</a>
```

Eine der wichtigsten Unterscheidungen ist die zwischen:

- Sprechende Links
- Nicht-sprechende Links

p.t.o →

Links(2) – Sprechend versus Nicht-sprechend

Bsp: Ein **nicht-sprechender** Link:

Den Download zu HTML-Kit finden Sie

```
<a href="http://www.chami.com/html-kit/download/">hier</a>.
```

Bsp: Besser – aber **nicht wirklich sprechend**:

HTML-Kit kann man kostenlos

```
<a href="http://www.chami.com/html-kit/download/">downloaden</a>.
```

Bsp: Jetzt der **sprechende** Link:

```
<a href="http://www.chami.com/html-kit/">HTML-Kit</a>
```

kann man kostenlos downloaden.

So soll's sein – Sprechend²

```
<a href="http://www.chami.com/html-kit/">HTML-Kit</a>
```

```
kann man kostenlos <a href="http://www.chami.com/html-kit/download/">downloaden</a>.
```

Links(3) – Verlinkte Bilder

Um ein **Bild mit einem HyperLink** zu unterlegen, muss man das Bild einfach nur in ein a-Tag geben:

```
<a href="http://www.w3.org/TR/xhtml1/">  
      
</a>
```

In der Regel werden Browser dann das gesamte Bild mit einem **Rahmen** umgeben. Wenn man das **nicht** möchte muss man für solche Bilder eine CSS-Regel angeben:

```
a img {border: none;}
```

Links(4) – Spezielles CSS

Wenn der Benutzer einen Link schon besucht hat, weiß der Browser davon und ist in der Lage unbesuchte Links (`a:link`) und besuchte Links (`a:visited`) eine jeweils andere Farbe zu geben. Dies ist wiederum durch CSS steuerbar:

```
a:link , a:visited , a:hover {
    color: #006699;
    text-decoration: none;
}
a:visited { color: #004466; }
a:hover { text-decoration: underline; }
```

`a:hover` bestimmt das Aussehen des Links, wenn der Benutzer den Maus-Cursor über den Link bewegt.

Außerdem sahen wir hier auch ein Beispiel, wie sich Definitionen für ein und dasselbe Element auch durchaus verteilen darf.

Anchors → anchor-Tag mit id=

Das **a** anchor-Tag, das zur Definition von HyperLinks dient, hat auch noch eine **andere Bedeutung**:

Er erlaubt Anchors/“**Ansprungspunkte**” zu deklarieren, v.a. im Text, die dann z.B. durch eine Navigationsleiste angesprungen werden können.

```
<a id="beliebigerAnchorName"></a>An diese Stelle  
kann man per Link verweisen...
```

Und jetzt verweisen wir an die

```
<a href="#beliebigerAnchorName">Stelle mit  
dem Anchor</a>.
```

Anchors(2)

Beachte:

- Der Anchor-Name entspricht der Id des a-Tags
- Um einen Anchor anzuspringen, muss man der entsprechenden Id ein # voranstellen.
- Man kann auch Anchors von "ausen" anspringen:

```
<p>So verweisen wir von "ausen"  
<a href="http://www.8ung.at/  
chr15t0ph/webPublishing.html#WebSpaceProviderFree">  
in eine meiner Seiten hinein</a>.  
</p>
```

Überholt

Seit XHTML 1.0 kann im Grunde jedes Element mit eigener Id durch einen #-Link angesprungen werden!

Dadurch sind a-Tags für Anspringpunkte eigentlich **nicht mehr von Bedeutung!**

(In HTML gab es das name=""-Attribut, anstelle der Id!)

HTML-Kit – Installation

Als erstes installieren wir als Vorbereitung

“**Mozilla ActiveX Control v1.6**”:

- 1 “Install” ... “Close”
- 2 Das war’s schon. Jetzt wird in HTML-Kit die Gecko-Engine als Preview-Möglichkeit funktionieren.

Jetzt installieren wir **HTML-Kit**:

- 1 Klicke “Ja”, “Next”, “Yes”, “Next”, “Next” (Full Installation), “Next” und “Install” ... “Finish”
- 2 HTML-Kit wird das erste mal gestartet: “Yes”, “OK” (Windows Integration Fenster), “Yes”; dann “No” (Download Web Links now?), “No” (Check for Updates) und “No” (take a web tour?)

HTML-Kit – Die erste HTML Seite

- 1 Schließlich erscheint der Standard-Begrüßungs-Wizard: Und wir können gleich mal “[Create a new file](#)” mit “OK” bestätigen.
- 2 Wir löschen als erstes mit [Strg+A](#) (“alles markieren”) und [Delete](#) das Standard-Grundgerüst, weil es für das veraltete HTML 4.01 ist. Dann erstellt man das [XHTML 1.0 Grundgerüst](#) ganz einfach mit: [Actions >> Document >> “HTML 4.01 Document - Transitional” >> “XHTML 1.0 Document - Strict”]
 - (Der Zwischenmenü-Eintrag “HTML 4.01 Document - Transitional” heisst so, weil’s der erste Eintrag in der darauffolgenden List ist. Blöd, is’ aber so ;-)
 - Das Menü unter [Actions] ist übrigens über die [Symbolleiste](#) eins-zu-eins abgebildet! Also [SymbL >> Document >> [rechtestes Symb→Dropdown] >> “XHTML 1.0 Document - Strict”]

HTML-Kit – Browser-Vorschau

- 1 Wir schreiben einen kurzen Text zwischen die body-Tags ...
- 2 **Browser-Vorschau:**
 - Die Taste **F12** toggelt zwischen der “Editor”- und der “Preview”-Ansicht
 - Gleich mal ausprobieren: mehrmals F12 drücken!
 - In der Preview-Ansicht kann auch statt IE die Gecko Engine verwendet werden: Am unteren Rand der Preview-Ansicht auf [[Gecko Mode](#)] klicken.

HTML-Kit – Text-Inhalte

Freies üben

Bitte jetzt einige **Überschriften** und **Absätze** eingeben.

Ein bisschen CSS: z.B. **Schriftart** auf Arial.

Dann vielleicht ein **Bild** einbinden?

Schließlich mindestens einen **Link** setzen (z.B. zu www.google.de).

Nach Lust und Laune **Anchor**s setzen und “anspringende” Links erstellen.

Aber nicht vorschnell sein!!

Unter keinen Umständen ungeduldig sein!

Auf keinen Fall ausprobieren: **Listen** und **Tabellen**.

Immer eins nach dem anderen!

HTML-Kit – Keyboard Shortcuts

Keyboard Shortcuts kann man eingeben unter:

[Tools » Customize » “Customize Keyboard Shortcuts...”
(Ctrl-F10) && “Add...”]

1 Links

- “Shortcut key”: Ctrl-Shift-L
- “New Command” » “Insert text”:

```
<a href="|">{{SELTEXT}}</a>
```

- “« Add” nicht vergessen, erst dann “OK”

2 Anchors

- “Shortcut key”: Ctrl-Shift-A
- “New Command” » “Insert text”:

```
<a id="{{SELTEXT}}|"></a>{{SELTEXT}}
```

HTML-Kit – Keyboard Shortcuts(2)

1 HTML Tidy

- “Shortcut key”: Ctrl-Shift-Alt-T
- “New Command” \gg “Action”:
“Tools > HTML Tidy” plus “Convert to XHTML”

2 HTML Tidy: Beautify

- “Shortcut key”: Ctrl-Shift-Alt-B
- “New Command” \gg “Action”:
“Tools > HTML Tidy” plus “Indent tags / beautify”

3 Copy Output to Editor

- “Shortcut key”: Ctrl-Shift-Alt-C
- “New Command” \gg “Menu command”:
“MCopyOutputToEditor”

HTML-Kit – Plug-ins

Zusätzlich noch die wichtigsten HTML-Kit Plug-ins:

1 includeHTML

- “Next”, “Yes”, “Next”, wähle “Full installation”, “Next”, “Next”, “Next”, “Install” ... (dann: deaktiviere “run” und “open help” Auswahl) “Finish”

2 w3c offline Referenzen

- “Next”, “Yes”, “Next”, “Install” ... “Finish”
- **HTML-Kit starten**, und unter [Edit >> Preferences >> Help:“Help Options”] die Option “Use only local keyword help files” aktivieren!)
- Mit **F1** wird in der Editor-Ansicht dann die kontext-sensitive Referenz aufgerufen.

Beachte: Die obigen Plug-ins sind mit die einzigen, die einen eigenen Installer mitbringen. Normalerweise laufen Plug-in Installationen aus dem gestartetem HTML-Kit heraus – über: [Tools >> Install >> “Install Plugin...”].

Zusammenfassung

Im diesem ersten Teil des Seminars haben wir kennengelernt:

- 1 Doctype und Grundgerüst
- 2 Text und Überschriften
- 3 Bilder
- 4 CSS – Erste Einführung
- 5 Links and Anchors
- 6 HTML-Kit – Installation und erste Schritte

Noch Fragen?

- Folien erstellt mit \LaTeX (MiKTeX), TeXnicCenter Editor und den latex-beamer Klassen.
- Folien erhältlich unter www.chr15t0ph.de/webPublishing/_pdf/fitAtWebdesign.pdf
- © Christoph Peter Neumann – Quellcode der Folien auf Anfrage: c.p.neumann@web.de

Part II

HTML und CSS Kurs (Fortsetzung)

Agenda

- 4 Grundlemente – 2. Teil
 - Wiederholung des ersten Teils
 - Listen
 - Übung
 - Tabellen
 - Übung

- 5 Strukturierung durch “DIVisions”
 - DIV
 - CSS für Fortgeschrittene
 - Übung

- 6 Redundanzen Vermeiden
 - Editor Side Includes
 - includeHTML
 - Übung

Wiederholung des ersten Teils

Im ersten Teil des Seminars haben wir kennengelernt:

- 1 Doctype und Grundgerüst
- 2 Text und Überschriften
- 3 Bilder
- 4 CSS – erste Einführung
- 5 Links and Anchors
- 6 HTML-Kit – Installation und erste Schritte

Was fehlt?

Als nächstes kommen [Listen](#) und [Tabellen](#).

Dann [DIVs](#) und [mehr CSS](#) zur Strukturierung der Seite, z.B. für Navigationsleisten.

Ausserdem Tools (→ [includeHTML](#)) zur zentralen Speicherung von redundanten Inhalten, wie Header, Footer und NavBar.

Listen

In jeder Textverarbeitung gibt es **zwei Sorten** von Listen:

- Aufzählungsliste – ``-Tag (engl. “**unordered list**”)
- Nummerierte Liste – ``-Tag (engl. “**ordered list**”)

Die einzelnen Listeneinträge sind innerhalb der `ul/ol`-Tags durch das ``-Tag ausgezeichnet.

Jenachdem in welcher Listenumgebung die Einträge stehen, bekommen sie entweder einen führenden “Bömmel” (`ul`), oder eine inkrementell aufsteigende Nummer (`ol`).

Bsp. "Unordered List"

```
<ul>  
  <li>Hund</li>  
  <li>Katze</li>  
</ul>
```

Anzeige ähnlich wie hier durch \LaTeX simuliert:

- Hund
 - Katze
-

Bsp. "Ordered List"

```
<ol>  
  <li>Hund</li>  
  <li>Katze</li>  
</ol>
```

Anzeige ähnlich wie hier durch \LaTeX simuliert:

- 1 Hund
 - 2 Katze
-

Verschachtelte Listen

Unter verschachtelten Listen verstehen wir einfach Listen unterschiedlicher Ebene, die sich meist durch ihre Einrückung unterscheiden.

Hier wieder eine \LaTeX -“Simulation” als Beispiel:

- ① Hund
 - ① Jagt Katzen
 - ② ...
 - ② Katze
 - ① ...
-

Mix ist erlaubt

Wie aus Textverarbeitungen bekannt, lassen sich UL und OL Listen beliebig mischen!

Verschachtelte Listen in HTML

```
<ol>
  <li>Hund
    <ol>
      <li>Jagt Katzen</li>
      <li>...</li>
    </ol>
  </li>
  <li>Katze
    <ol><li>...</li></ol>
  </li>
</ol>
```

Attention Please!

Man beachte, dass das LI-Tags die "eingeschachtelten" Listen vollständig umschließen! Direkt unterhalb eines OL/UL-Tags dürfen ausschließlich LI-Tags stehen!

Übung WYSIWYG – Nvu

- Spass machen tut die Eingabe von Listen nur mit WYSIWYG-Programmen wie **Nvu**. Dort hat man die übliche **Eingabeunterstützung** – es lassen sich geschachtelte Listen verschiedener Tiefe über Tab (und Shift-Tab) sehr leicht eingeben.
 - **Installation:** “Next”, wähle “**I accept the agreement**”, “Next”, “Next”, “Next”, “Next” und “Install” ... “Finish”
 - **Erster Start:** “Close” (Tip of the Day). **Dann:** “Edit” >> “Preferences...” && “General” >> “When Saving or Publishing Pages”: “**retain original source formatting**”
-
- Nvu erleichtert auch die Eingabe von normalen **Paragrafen** und **Überschriften!**

Übung WYSIWYG – Nvu – **Achtung**

- Besonders bei chaotisch wiederholtem Einfügen und Löschen von Zeilenwechselln und Paragraphen können sich bei Nvu im HTML-Quelltext **überflüssige Tag-Fragmente** anhäufen.
- Ausserdem wird in der Regel während der Eingabe von verschachtelten Listen, gegen die “ausschließlich LI-Tags unterhalb von UL/OL”-**Regel verstoßen!**
- ⇒ **Lösung:**
 - ① Dinge durch “Rückgängig” (**StrG-Z**) rückgängig machen, und nicht durch Löschen+Neuschreiben.
 - ② Mindestens vor dem Speichern:
“Tools” >> “Markup cleaner” >> “Clean up” aufrufen!!

Übung CSS – TopStyle

- Die Eingabe von **CSS** kann von einem Programm wie **TopStyle** durch **Auto-Completion** ungemein vereinfacht werden (StrG-Space, wie in Programmierumgebungen üblich).
- **Installation:** “Next”, “Yes”, “Next”, “Next” ... “Finish”
- **Erster Start:** “OK”. **Dann:** wähle im “Style Inspector”: “**CSS Level 2**”

Übung CSS – Beispiele

```
ul{
    list-style-type: square;
}

ul ul, ol ul{
    list-style-type: disc;
}

ol {
    list-style-type: upper-roman;
}
```

Tabellen

```
<table>
  <tr>
    <td>Personal information</td>
  </tr>
  <tr>
    <td>Name</td>
    <td>Christoph Peter Neumann</td>
  </tr>
</table>
```

- 1 Das Basis Tag nennt sich `<table>`
- 2 Es werden zuerst die **Reihen** (engl. "table row", `tr`) angegeben.
- 3 Spalten werden nicht angegeben. Stattdessen werden pro Reihe mehrere Datenfelder ("table data", `td`) angegeben, die implizit zu einer gewissen Spaltenanzahl führen.

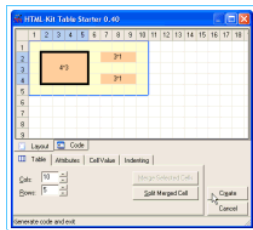
Tabellen(2)

```
<table border='0' cellspacing='0' cellpadding='0'
  summary="European Curriculum Vitae">
  <tr>
    <td class="heading3">Personal information</td>
  </tr>
  <tr>
    <td class="leftHalf">
      <ul><li>Name</li></ul>
    </td>
    <td>Christoph Peter Neumann</td>
  </tr>
</table>
```

- 1 Eine "blinde Tabelle" (ohne Gitternetzlinien)
- 2 Auch Listen, oder Bilder et al., sind als Inhalt erlaubt.
- 3 Zwei Datenfelder sind auf CSS-Formatierung vorbereitet!

Übung

- Nvu bietet einen sehr gelungenen Mechanismus um Tabellen graphisch anzulegen und zu erweitern:
“Table” >> “Insert” >> “Table...”
- Aber auch für HTML-Kit gibt es ein sehr gutes Plug-In, das die Erstellung von Tabellen sehr erleichtert:
[hkTableStarter](#).
Nach der Installation (“Tools” >> “Install” >> “Install Plugin...”), findet man es unter “Actions” >> “Tables” >> “hkTableStarter”.



Das DIV Tag

Das `<div>`-Tag erlaubt rechteckige Bereiche zu definieren. Diese dienen dazu spezielle Inhalte zu kapseln:

```
<body>
  <div id="header">
  </div>

  <div id="navbarContainerHack">
    <div id="navbarContainer">
    </div>
  </div>

  <div id="content">
  </div>

  <div id="footer">
  </div>
</body>
```

DIV(2)

Für DIVs gibt es praktisch **immer** durch **CSS** eine Definition, die Größe, Position und Ausrichtungsverhalten der DIVs bestimmen.

Bsp.:

```
#header{  
    background-color: transparent;  
    margin: 0;  
    padding: 10px 0px;  
    width: 100%;  
    height: 50px;  
    position: absolute;  
}
```

DIV(3)

Offline Information

Eine genaue Diskussion der CSS-Definitionen für die DIVs findet ausserhalb der Folien, auf Basis der Bsp.-Seiten statt!

CSS für Fortgeschrittene

Durch CSS kann man für verschiedene Ausgabemedien die Anzeigeeigenschaften einstellen.

Die [w3c](#) hat verschiedene [Medientypen](#) definiert. Das wichtigste (und mit das einzige, dass im Moment schon von Browsern zusätzlich verstanden wird) ist der Typ **“print”**. Es bestimmt die [Druckansicht](#), wenn man die HTML-Seite aus dem Browser heraus druckt.

```
@media print {  
    body {  
        color: Black;  
        background-color: White;  
        padding: 0em 0em 0em 0em;  
    }  
}
```

CSS für Fortgeschrittene(2)

Man kann auch die Ausgabe/Ausdruck von Elementen, wie z.B. Hilfs-DIVs (Header, Footer, NavBar), durch eine “`display: none;`” Anweisung komplett unterdrücken.

```
@media print {
  body {
    color: Black;
    background-color: White;
    padding: 0em 0em 0em 0em;
  }
  h1,h2,h3,h4,h5,h6 { color: Black; }
  a:link { color: Black; }

  #header{ display: none; }
  .navbar { display: none; }
  #content{ padding: 0 0 0 0; }
  #footer { display: none; }
}
```

Übung

Vorschlag

Eine Möglichkeit wäre an dieser Stelle die beispielhaften CSS-Definitionen zu übernehmen – und sie zu variieren.
Oder: eigene Ideen? Fragen stellen!

Editor Side Includes

Editor Side Includes bedeutet, dass HTML-Schnipsel in einer zentralen Datei (XML !) verwaltet werden, und in den XHTML-Dateien über Kommentar-Referenzierungen eingebunden werden.

Die genaue Syntax und Verwendung von Editor Side Includes sind **nicht standardisiert**, sondern abhängig vom Editor!

Ein Plug-In für HTML-Kit, das Editor Side Includes ermöglicht, ist **includeHTML** → **p.t.o.**

includeHTML

Bei `includeHTML` muss ein XML-Dokument angelegt werden:

```
<?xml version="1.0"?>
<snippetcollection>

  <snippet key="header">
    <![CDATA[
      <div id="header">
        <a id="headerAnchor"></a>
        <h1 id="siteName">Christoph Neumann</h1>
      </div>
    ]]>
  </snippet>

</snippetcollection>
```

Ein solches Snippet kann dann durch eine Kommentar-Referenz in einer XHTML-Datei verwendet werden → [p.t.o.](#)

includeHTML(2)

An der passenden Stelle im XHTML-Dokument wird folgender Kommentar geschrieben:

```
<body>
...
  <!--ih:includeHTML
    file="_include/includeHTML.xml"
    key="header"-->
  <!--/ih:includeHTML-->
...
```

includeHTML(3)

Wird jetzt includeHTML als Tool auf diese Datei angewendet, dann wird der Kommentar wie folgt ersetzt:

```
<body>
...
  <!--ih:includeHTML
    file="_include/includeHTML.xml"
    key="header"-->
  <div id="header">
    <a id="headerAnchor"></a>
    <h1 id="siteName">Christoph Neumann</h1>
  </div>
  <!--/ih:includeHTML-->
...
```

Alternative Möglichkeiten

- 1 Server Side Includes (SSI)
 - Ähnlicher Mechanismus wie Editor Side Includes
 - Standardisiert!
 - Ersetzung durch den WebServer (bei Anfrage der HTML-Seite)
- 2 Dynamic Content Creation
 - Aus Datenbanken erzeugte Inhalte
 - Zum Beispiel: WebShops
 - (Dies ist die anspruchvollste Art HTML zu erzeugen ...)
 - (Auch Content Management Systeme (CMS) wie das Open-Source Projekt [Mambo](#) basieren hierauf.)

Übung

- 1 **Download:** `includeHTML`
- 2 **Installation:** “Next”, “Yes”, “Next”, wähle “Full installation”, “Next”, “Next”, “Next”, “Install” ... (dann: deaktiviere “run” und “open help” Auswahl) “Finish”

Plug-in versus Stand-alone

V.a. die Funktion “rekursiv alle html-Dateien einer ganzen Verzeichnisstruktur abzuarbeiten”, ist aus HTML-Kit nicht verfügbar, sondern nur über die eigenständige `includeHTML`-Programmoberfläche!

Zusammenfassung

Im diesem zweiten Teil des Seminars haben wir kennengelernt:

- 1 Listen
- 2 Tabellen
- 3 DIVs
- 4 CSS für Fortgeschrittene
- 5 Editor Side Includes (includeHTML)

Noch Fragen?

- 1 Folien erstellt mit \LaTeX (MiKTeX), TeXnicCenter Editor und den latex-beamer Klassen.
- 2 Folien erhältlich unter www.chr15t0ph.de/webPublishing/_pdf/fitAtWebdesign.pdf
- 3 © Christoph Peter Neumann – Quellcode der Folien auf Anfrage: c.p.neumann@web.de