FH Salzburg

VO Web-Technologien

04.10.2023, Oliver Jung

Technik
Gesundheit
Medien

http://www.csszengarden.com/

Erinnerung

- HTML folgt der grundlegenden Idee von SGML:
  Trennung von Dokumentstruktur und Dokumentpräsentation
  - SGML beschreibt die Dokumentstruktur
  - Formatierungsbeschreibungssprache DSSSL (Document Style Semantics and Specification Language) definiert das Layout
- Da Browser zu Beginn keine grafische Anzeige besaßen, wurde der Layout-Präsentation bei Webdokumenten kaum Beachtung geschenkt und zu HTML anfänglich keine separate Formatierungs-Beschreibungssprache eingeführt
- Mit dem Aufkommen von grafischen Browseroberflächen – 1993 Mosaic Browser – erfolgten die zunehmend erforderlichen Formatierungsanweisungen direkt in HTML

Erinnerung

- Rapides Wachstum des WWW führte zum Wunsch, das Dokumenten-Layout so gestalten zu können, das es sicher reproduzierbar und auch an das jeweilige Ausgabemedium angepasst sein sollte
- Mit **Cascading Stylesheets** (**CSS)** wurde eine unabhängige Formatierungsbeschreibungssprache entwickelt, um **Stylesheets** für die Layout-Präsentation von HTML-Dokumenten beschreiben zu können
- Dabei beschreibt HTML die Dokumentstruktur und Stylesheets legen die Layout-Präsentation der einzelnen Strukturelemente fest für die jeweiligen Ausgabemedien

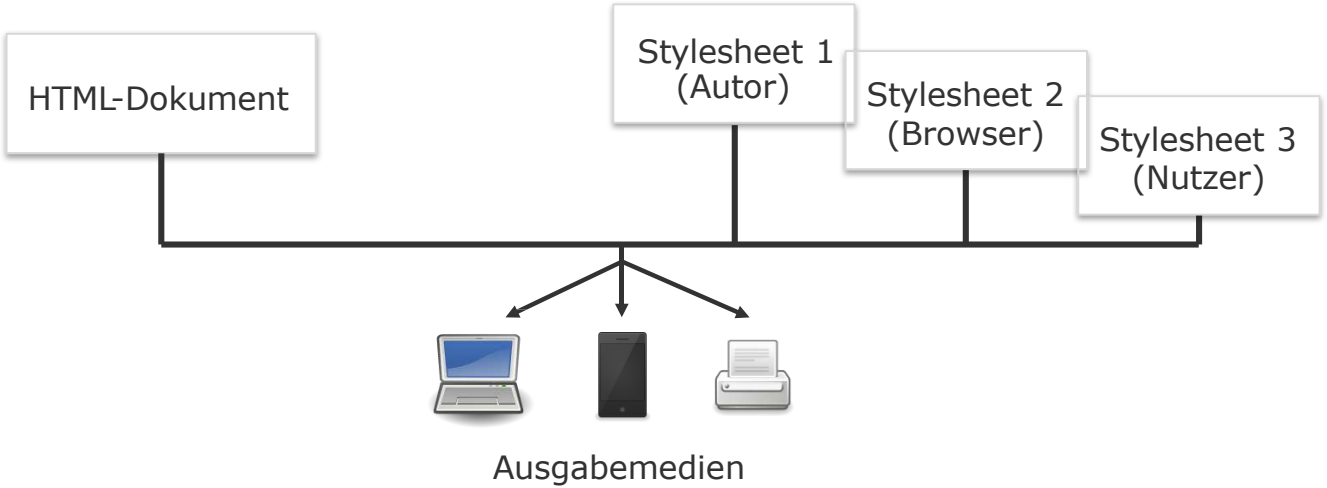
CSS – Cascading Stylesheets

Mit CSS erstellte Stylesheets ermöglichen:

- Einheitliches Layout in zugehörigen HTML-Dokumenten
- Entwurf eines Dokuments nach den Wünschen und Bedürfnissen des Autors oder Nutzers
- Nutzung und Koexistenz von kompletten Stylesheet-Hierarchien, welche vom Autor, Browser-Hersteller oder Nutzer erstellt werden können
- Dokumentenübergreifende Wiederverwendung von Layouts

CSS – Cascading Stylesheets

Verwendung von Stylesheets



CSS – Cascading Stylesheets

- CSS bietet direkte Ergänzung zu HTML und weist individuellen HTML-Kommandos Ausgabemedien-spezifische Formatierungen zu
- Haupteigenschaften von CSS sind:
  - exakte Definition des Layouts eines HTML-Dokuments
  - Anpassung an verschiedene Ausgabemedien, z.B.
    - Anweisungen für Druck-Layout,
    - künstliche Sprachausgabe,
    - Smartphones, ...
  - zentrales Layout-Management – Layout-Definition und Aktualisierung kann an zentraler Stelle erfolgen

CSS – Kurze Geschichte

- Erste Spezifikation von CSS (1.0) wurde 1996 vom W3C veröffentlicht zusammen mit Anpassung des HTML 4.0-Standards
- **CSS 2.0** (1998) bietet Ausgabe verschiedener Medientypen – Druck- und Sprachausgabe – und Pixel-genaue Positionierung von integrierten Objekten
- Browser arbeiten (bisher) nicht immer Standard-konform
  - 2009 nahm W3C die Zwischenversion CSS 2.1 als „empfohlenen Kandidaten" an, um Unregelmäßigkeiten zu korrigieren
- Gleichzeitig begann Entwicklung von CCS 3, eine Version mit modularer Struktur
  - Die Spezifikationsphase für die meisten Teile ist abgeschlossen und Browserhersteller setzten die meisten der empfohlenen Module der W3C um

CSS – Verbindung mit HTML-Dokumenten

Stylesheets können auf verschiedene Arten mit HTML-Dokumenten verbunden werden:

- **Inline-Definition** in HTML-Elementen mit **style**-Attribut, z.B.
  - `<span style="font-weight: bold">Fetter Text</span>`
- **(Interne) Stylesheets** werden in einem HTML-Dokument definiert mit dem **<style>**-Tag (sollte im Header stehen)
  - **Anmerkung**: Hierbei sind Dokumentenstruktur und Präsentation noch nicht strikt getrennt
- **(Externe) Stylesheets** aus separaten Dateien werden eingebunden mithilfe der **<link>**-Deklaration im Header des HTML-Dokuments
- Wenn kombiniert wird, erhalten Inline-Styles Vorrang vor internen Stylesheets (im Header des Dokuments) und externen Stylesheets (lokale Definition vor globaler Definition)

CSS Syntax – Regeln

- **CSS Stylesheet** ist Ansammlung von Regeln die auf ein HTML-Dokument angewandt werden
- Beispiel einer **CSS Regel**:

```
h1 {
  color: blue;
  font-family: Arial, Helvetica, sans-
  serif; font-size: 14px;
}
```

- **CSS Regel** besteht aus:
  - **Selektor** (`h1`) und
  - **Deklaration(en)** (`{ color: blue; font-family: […] }`)

CSS Syntax – Deklarationen

- **Deklaration** beinhaltet die eigentliche Formatierungsanweisung und weist einer **Eigenschaft** einen oder mehrere **Werte** zu
- Jede CSS-Regel kann **mehrere Deklarationen** enthalten
- Mögliche Formatierungsanweisung beziehen sich z.B. auf die folgenden Elemente:
  - **Schrift** – Schriftart, Schriftgröße, Schriftfarbe
  - **Abstände und Rahmen** – für alle Arten von HTML-Elementen
  - **Position und Größe** – für alle Arten von HTML-Elementen
  - **Farben und Hintergründe** – für alle Arten von HTML-Elementen
  - **Sichtbarkeit und Art der Anzeige** – für alle Arten von HTML-Elementen
  - **Umfließung** – für alle Arten von Block-Elementen
  - ...

CSS Syntax – Selektoren

Selektor begrenzt potenziellen Anwendungsbereich von Deklarationen (Formatierungsanweisungen), kurz,

> ein Selektor **wählt HTML-Elemente aus**, auf die Deklarationen angewendet werden

Es gibt:

- **Klassen- und ID-Selektoren** – begrenzen den gültigen Bereich auf bestimmtes Element bzw. Gruppe von Elementen gleicher Klasse
- **Attribut-Selektoren** – Elemente mit einem bestimmten Attribut bzw. Attribut-Wert
- **Context-Selektoren** – begrenzen Vererbung von Formatierungsanweisungen
- **Externe Selektoren** (Pseudo-Elemente) – trifft auf Elemente zu, die nicht Teil des HTML-Dokuments sind, aber Teil des Kontexts

CSS Syntax – Selektorem

Beispiele für verschiedene Selektor-Typen

- Standard-Selektor
  - Beispiel:

```
p { color: red; }
```

```
<p>roter Text</p>
```

- Attribut-Selektor
  - Beispiel:
  - oder:

```
p[small] { font-size: 8px; }
```

```
<p small>Das ist eine Fußnote …</p>
```

```
a[target="_blank"] { color:red; }
```

```
<a target="_blank">roter Link</p>
```

- Kontext-Selektor
  - Beispiel:
  - Erläuterung:

```
td p { color:#0000FF; }
```

```
<td><p>blauer Text in Tabelle</p></td>
```

Absätze (<p>) in Tabellen-Zelle (<td>) werden blau dargestellt

CSS Syntax – Selektoren

Beispiele für verschiedene Selektor-Typen

- Klassen-Selektor `p.footnote { font-size: 8px; }`
  - Beispiel: `<p class="footnote">Dies ist eine Notiz</p>`
  - Hinweis: Ein Element kann mehrere Klassen haben, bspw.
    `<h1 class="blue center">...</h1>`
- ID-Selektor `a#imprint { font-weight: bold; }`
  - Beispiel: `<a id="imprint" href="...">Impressum</a>`
- Hinweis: **class**-Attribut können für **mehrere Elemente** identisch sein, **id** muss in einem Dokument **einzigartig** sein
- Klassen- und ID-Selektoren können ohne Element definiert (und auf verschiedene Elemente angewandt) werden:
  - Beispiele:
    ```
    .red { color: #FF0000; }
    #invisible { visibility: hidden; }
    ```

CSS Syntax – Selektoren

Gruppieren von Selektoren

- Will man identische Deklarationen für mehrere Selektoren anwenden, kann man diese zusammenfassen
- ```
  h1        { font-weight: bold; }
  #headline { font-weight: bold; }
  ```
  wird zu:
  ```
  h1, #headline { font-weight: bold; }
  ```

Manipulation von Element-Inhalten

- Mit den Selektoren `::before` und `::after` kann vor oder nach dem Inhalt eines Elements Text und/oder CSS ergänzt werden
- ```
  p.wichtig::before {
  content: "Wichtig: ";
  color: red;
  font-weight: bold;
  }
  ```

```
<p class="wichtig">CSS lernen</p>
```

Wichtig: CSS lernen

CSS Syntax – Selektoren

Pseudo-Klassen und Pseudo-Elemente

Einige Selektoren können sich auf bestimmte Zustände von Elementen oder auf bestimmte Untermengen beziehen

- **:hover** – für „Mouseover"-Zustand von Elementen, z.B.
  ```
  a:hover { text-decoration: underline; }
  ```
  (Links werden unterstrichen, wenn Maus darauf zeigt)
- **:focus** – für aktives Eingabefeld in Formularen, z.B.
  ```
  input:focus { border: 1px solid blue; }
  ```
  (blauer Rahmen um aktives Eingabefeld)
- **::first-letter** – für den ersten Buchstaben in einem Element, z.B.
  ```
  p::first-letter { font-size: 24px; font-weight: bold; }
  ```
  („Initial": erster Buchstabe eines Absatzes groß und fett)
- Gibt eine Reihe weitere, z.B. ::first-line, :first-child, :active, :required, ::selection, ...

Dynamische Selektoren in CSS 3

CSS 3 bietet Reihe neuer **dynamischer Selektoren**, z.B.:

- E:nth-child(*n*) — • *n*-te Kind-Element vom Element *E*
- E:nth-last-child(*n*) — • *n*-te Kind-Element gezählt vom letzten
- E:nth-of-type(*n*) — • *n*-te Kind-Element vom Typ
- E:nth-last-of-type(*n*) — • *n*-te Kind-Element gezählt vom letzten
- Anstelle von Zahlenwert *n* sind auch Formeln möglich, z.B.:

```
li:nth-of-type(2n) { color: red; }
```

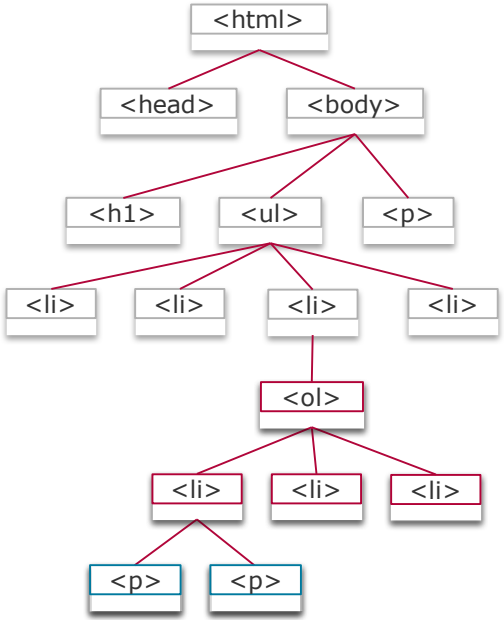
(Färbt jedes zweite <li> rot)

- Außerdem gibt es in CSS 3 den **Negations-Selektor**
  - :not(E) alle Elemente außer Element E
  - `:not(p)` wählt alle Elemente außer `<p>`

CSS Syntax – Vererbung

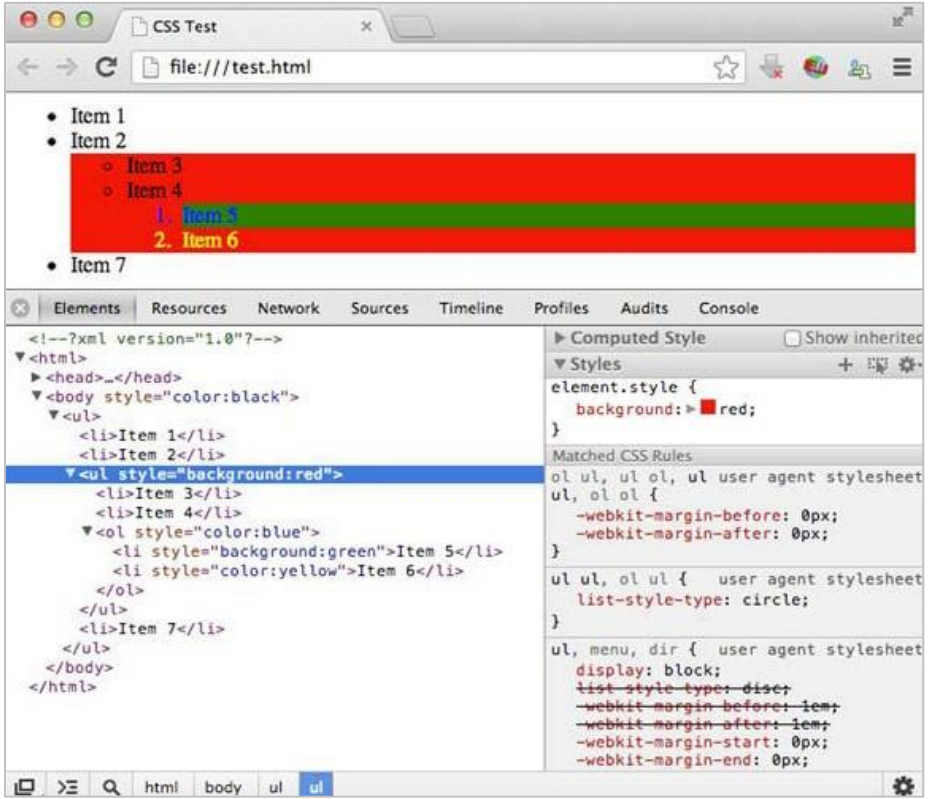
CSS-Formatierungsanweisungen werden im HTML-Dokumentenbaum an alle innerhalb eines Elements liegenden Unterelemente **vererbt**

- Beispiel:
  - `ol { background-color: red; }`
  - Unterbaum von `<ol>` erbt rote Hintergrundfarbe
- Spezielle Selektoren können Elemente **im Unterbaum** eines Elements auswählen:
  - `ol p { background-color: blue; }`
  - Alle `<p>`-Elemente unterhalb von `<ol>` werden blau gefärbt
- Spezifischere Selektoren überschreiben ererbte Eigenschaften (`ol p` ist spezifischer als `ol` → Hintergrund von `<p>` wird blau)



CSS Syntax – Vererbung

Beispiel:
CSS Vererbungs-konzept



CSS – Mehrere Stylesheets

Eine grundlegende Idee von CSS ist gleichzeitige Nutzung von mehreren Stylesheets. **Mögliche Kaskadierung von Stylesheets** umfasst:

- **Browser-Stylesheet**s – in jeder Dokumentenrepräsentation wird ein Browser-spezifisches Stylesheet genutzt
- **Nutzer-Stylesheets** – Browser bieten Nutzern (begrenzte) Möglichkeiten an, die Dokumentenrepräsentation anzupassen (Schrift, Farbe, etc.). Priorität ist höher als Browser-Stylesheets
- **Autoren-Stylesheets** – Autoren haben großen Freiraum in der Layout-Gestaltung; sie können mehrere Stylesheets definieren:
  - modulare Stylesheet Organisation
  - hierarchische Stylesheet Organisation
  - alternative Stylesheets, z.B. für verschiedene Präsentationsformen

CSS – Mehrere Stylesheets

Nutzung von mehreren Stylesheets kann zu **Darstellungsfehlern** führen, wenn einem HTML-Element gegensätzliche Formatierungsanweisungen zugewiesen werden – was auch innerhalb eines Stylesheets möglich ist

CSS definiert Reihe von Regeln zur Konfliktlösung:

1. Finde alle relevanten Regeln
2. Ordne Regeln nach ihrer Wichtigkeit – Autoren können Regeln in CSS-Definitionen das Attribut `!important` geben
3. Ordne Regeln nach ihrem Ursprung: Autoren- vor Nutzer- vor Browser-Stylesheets
4. Ordne Regeln nach ihrem Spezialisierungsgrad: Anzahl von ID- und Klassen-Attributen
5. Ordne verbleibende, konkurrierende Regeln chronologisch nach ihrem Auftreten

CSS Eigenschaften - Übersicht

Eigenschaft	
Schriftbild	font-family, font-size, font-weight, ….
Textformatierung	color, word-spacing, text-decoration, word-wrap, …
Textausrichtung	text-align, white-space, text-indent, …
Listen	list-style-type, list-style-position, list-style-image
Tabellen	table-layout, empty-cells, border-spacing, border-collapse, ..
Hintergrund	background, background-image, …

Eigenschaft	
Äußere Gestaltung	border, outlinie, box-shadow
Schreib & Leserichtung	direction (ltr, rtl)
Abstand	padding, margin
Anzeigeart	visibility, opacity, display, …
Positionierung	position, float, shape-outside
Layout-Modelle	width, height, flexbox, box-sizing, …
Animationen	transition, animation
Media Queries	@media

CSS Positionierung

- Mittels **Positionierung** können Elemente innerhalb eines HTML-Dokumentes angeordnet werden
- Auch festgelegt wird, wie Elemente von Text umflossen werden
- Mit `position` können Elemente beliebig positioniert und aus dem regulären Elementfluss entfernt werden:
  - `absolute`
    - Position bezieht sich auf Vorfahren-Element oder Wurzelelement
    - losgelöst vom Textfluss
    - Position per `left`, `right`, `top` und `bottom`
  - …

CSS Positionierung

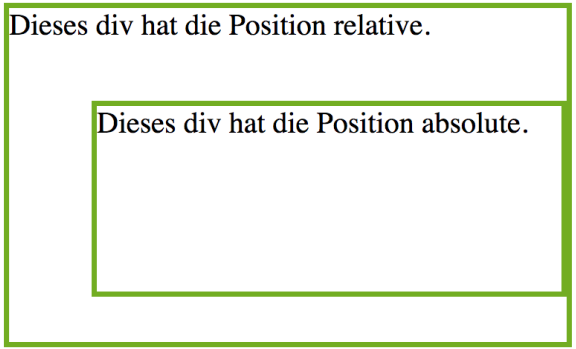
- **`fixed`**
  - Starre Ausrichtung ohne Abhängigkeiten
  - Losgelöst vom Textfluss
  - Position per **`left`**, **`right`**, **`top`** und **`bottom`**
- **`relative`**
  - Bezugspunkt für absolut positionierte Kind-Elemente
  - Bleibt im Textfluss
  - Position per **`left`**, **`right`**, **`top`** und **`bottom`**
- **`static`**
  - Standardwert der Eigenschaft **`position`**
  - Bleibt im Textfluss
  - **`left`**, **`right`**, **`top`** und **`bottom`** werden ignoriert

CSS Positionierung – Beispiel

- **CSS Stylesheet**:

```
div.relative {
  position: relative;
}
div.absolute {
  position: absolute;
  top: 50px;
  right: 0;
}
```

- **Ausgabe**:



- **HTML**:

```
<div class="relative">Dieses div hat die Position relative.
  <div class="absolute">Dieses div hat die Position
  absolute.</div>
</div>
```

CSS Positionierung – float

Die Eigenschaft **float** legt fest, in welche Richtung ein Element „**fließt**“:

1. Element kann **links** (float: left) bzw. **rechts** (float: right) fließen.

   => an die **linke** bzw. **rechte** **Innenkante**, sowie **Oberkante** des **Elternelements**
2. Ist in diesem „Floating-Weg“ zum Elternelement ein **anderes gefloatetes Element** „**stößt**“ es **dort an –** block Elemente werden ignoriert
3. **Text,** sowie **inline** & **inline-block** – Elemente **legen sich um** das gefloatete **Element** herum
4. Das Element nimmt nur mehr die **Breite** ein die der **Inhalt** benötigt.
5. Die gesamte „Box“ **inkl. margins** wird beim Floaten berücksichtigt (kein margin collapsing).

Hilfe: https://developer.mozilla.org/en-US/docs/Web/CSS/float

CSS Positionierung – float

```
img {
   float: right;
   margin: 0 0 10px 10px;
}
```

```
<p>In this example, the image ...</p>

<p>
  <img src="w3css.gif" alt="W3Schools.com"
       width="100" height="140">
  Lorem ipsum dolor sit amet, ...
</p>
```

Elternelement

Float-Element

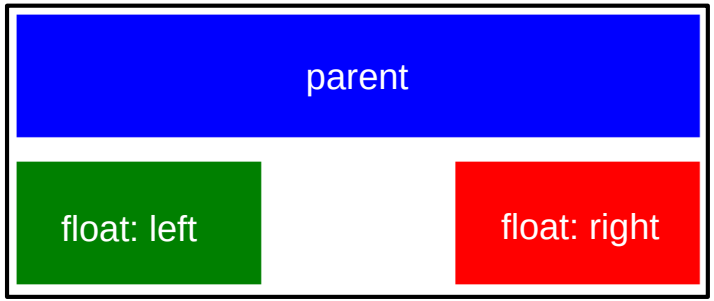
In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.

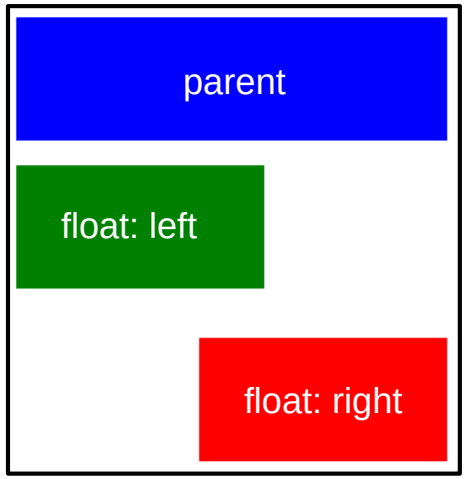
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

CSS Positionierung – float

```
<div class="parent"></div>
<p class="left"></p>
<p class="right"></p>
```

```
.parent {
    height: 5rem;
    background-color: blue;
}
p {
    height: 5rem;
    width: 10rem;
}
p.left {
    float: left;
    background-color: green;
}
p.right {
    float: right;
    background-color: red;
}
```





CSS Positionierung – float

Ohne clear

div Typ A | div Typ A | div Typ B Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Using clear

div Typ C | div Typ C

div Typ D - Using clear moves div4 down below the floated div3. The value "left" clears elements floated to the left. You can also clear "right" and "both".

```
<h2>Ohne clear</h2>

<div class="divA">div Typ A</div>
<div class="divA">div Typ A</div>
<div class="divB">div Typ B Lorem ipsum ...</div>

<h2>Using clear</h2>
<div class="divC">div Typ C</div>
<div class="divC">div Typ C</div>
<div class="divD">div Typ D - Using clear ..</div>
```

```
.divA {
    float: left;
    width: 100px; height: 50px;
    border: 3px solid #73AD21;
}
.divB {
    border: 1px solid red;
}
.divC {
    float: left;
    width: 100px; height: 50px;
    border: 3px solid #73AD21;
}
.divD {
    border: 1px solid red;
    clear: left;
}
```

Hilfe: https://www.youtube.com/watch?v=xIJvkm-CgFQ

CSS Positionierung – display

```css
/* verstecken */
p.hidden {
        display: none;
}

/* "umwandeln" in ein block-Element */
img.layouted {
        display: block;
}

/* als Listenelement  darstellen */
p.item {
        display: list-item;
}
```

```css
/* im Text – keine Größenangaben möglich */
figure.in-text {
        display: inline;
}

/* wie inline, aber mit Breite und Höhe */
figure.in-text {
        display: inline-block;
}

/* als Tabelle  rendern */
div.tabular-layout {
        /* || display: table-row || display: table-cell
*/
        display: table;
}
```

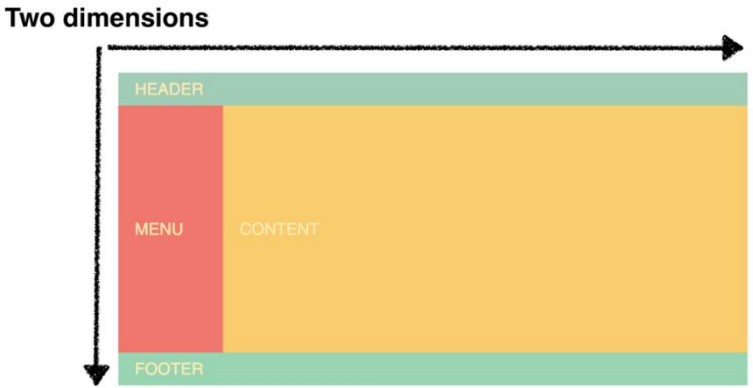
CSS Positionierung – flex & grid

- flex

Geeignet für
„Ein-dimensionale“ Layouts

- Grid

- Geeignet für
„Zwei-dimensionale“ Layouts



CSS Positionierung – flex

```
<header>
    <div>Home</div>
    <div>Search</div>
    <div>Logout</div>
</header>
```

HOME

SEARCH

LOGOUT

CSS Positionierung – flex

```
header {
    display: flex;
}
```

CSS Positionierung – flex

Container

- **flex-direction**: row | row-reverse | column | column-reverse;
- **flex-wrap**: nowrap | wrap | wrap-reverse
- **justify-content**: flex-start | flex-end | center | space-between | space-around | space-evenly;
- …

Element

- **order**: <integer>
- **flex-shrink / flex-grow**: <number>
- **flex-basis**: <length> | auto
- **align-self**: auto | flex-start | flex-end | center | baseline | stretch;
- …

Hilfestellung:

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

CSS Positionierung – grid

```
<div class="container">
  <header>HEADER</header>
  <aside>MENU</aside>
  <main>CONTENT</main>
  <footer>FOOTER</footer>
</div>
```

CSS Positionierung – grid

```
.container {
    display: grid;
    grid-template-rows:    50px 350px 50px;
    grid-template-columns: repeat(12, 1fr);
}
header {
    grid-column: span 12;
}
aside {
    grid-column: span 2;
}
main {
    grid-column: span 10;
}
footer {
    grid-column: span 12;
}
```

CSS Positionierung – grid

Container

- **grid-column-gap**: <line-size>;
- **grid-row-gap**: <line-size>;
- **grid-template-columns**: <track-size> ... | <line-name> <track-size> ...;
- **grid-template-rows**: <track-size> ... | <line-name> <track-size> ...;
- …

Element

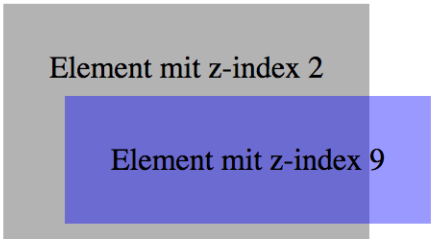
- **justify-self**: start | end | center | stretch;
- …

Hilfestellung:
https://css-tricks.com/snippets/css/complete-guide-grid/

CSS Positionierung – z-index

- Die `z-index` Eigenschaft kommt zur Anwendung, wenn sich Elemente überlappen
- Je größer der Wert, desto „höher" liegt das Element und überdeckt andere Elemente
- Kann nur für Elemente angegeben werden, die eine `position` definiert haben und nicht `static` sind
- **Beispiel** (CSS):

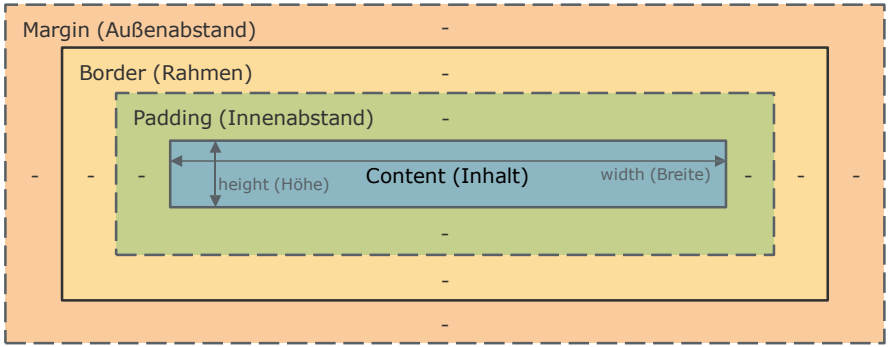
```
#element1 {
  z-index: 2;
}
#element2 {
  z-index: 9;
}
```



Box-Modell – Einführung

- HTML-Elemente, wie z.B. , werden durch innerhalb von Rechtecken dargestellt
- Diese Rechtecke werden durch das **Box-Modell** beschrieben
- Bestandteile einer Box sind:
  - **Inhalt** – Texte und Bilder
  - **Innenabstand** (padding)
  - **Rahmen** (border)
  - **Außenabstand** (margin)
- Die Innen- und Außenabstände sowie der Rahmen können für alle vier Seiten der Box individuell festgelegt werden

Box-Modell – Einführung



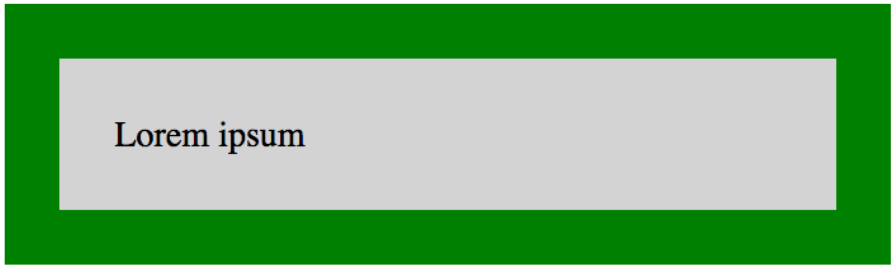
- Die einzelnen Bereiche werden bezeichnet als:
  - Content-Box
  - Padding-Box
  - Border-Box
  - Margin-Box

Box-Modell – Beispiel

- **CSS Stylesheet**:

```
div {
  background-color: lightgrey;
  width: 300px;
  padding: 25px;
  border: 25px solid green;
  margin: 25px;
}
```

- Ausgabe:



Box-Modell – Innen- und Außenabstand

- Abstände einzeln definierbar mit:

```
padding-top          margin-top
padding-right        margin-right
padding-bottom       margin-bottom
padding-left         margin-left
```

- Oder als zusammenfassende Eigenschaft:
  - Bei einer Angabe gilt der Wert für alle vier Seiten

```
padding: 10px;
margin: 10px;
```

  - Zwei bis vier Angaben: 1. Wert für top, 2. Wert für right, 3. Wert für bottom, 4. Wert für left

```
padding: 10px 5px;
margin: 10px 5px 10px 5px;
```

- Als Werte sind nicht negative numerische Längenmaße erlaubt, z.B. Pixel (`px`), Punkt (`pt`), Zentimeter (`cm`), Millimeter (`mm`), ...

Box-Modell – Einheiten, absolut

Einheit	Name CSS	Beschreibung	Beispiel
Zoll	in	Ein Zoll ist genau 2,54 cm lang.	margin-left: 1in;
Pica	pc	Ein Pica ist der sechste Teil eines Zolls. Es ist daher 12 Punkt lang. Pica ist eine in der Typografie verbreitete Maßeinheit.	font-size: 1pc;
Punkt	pt	Ein Punkt ist der 72ste Teil eines Zolls. Auch der Punkt ist eine in der Typografie verbreitete Maßeinheit.	font-size: 12pt;
Zentimeter	cm	Ein Zentimeter ist der hundertste Teil eines Meters und somit 10 Millimeter lang.	margin-top: 1.5cm;
Millimeter	mm	Ein Millimeter ist der tausendste Teil eines Meters bzw. der zehnte Teil eines Zentimeters.	padding: 1mm;

Einheit	Physische Größe	Größe in Pixel
1in	2,54 cm	96 Pixel
1pc	1/6 Zoll	16 Pixel
1pt	1/72 Zoll	1,33 Pixel (gerundet)
1cm	1 cm	37,8 Pixel (gerundet)
1mm	0,1 cm	3,78 Pixel (gerundet)
1px	***variabel***	**0,75 Punkt**

Box-Modell – Einheiten, relativ

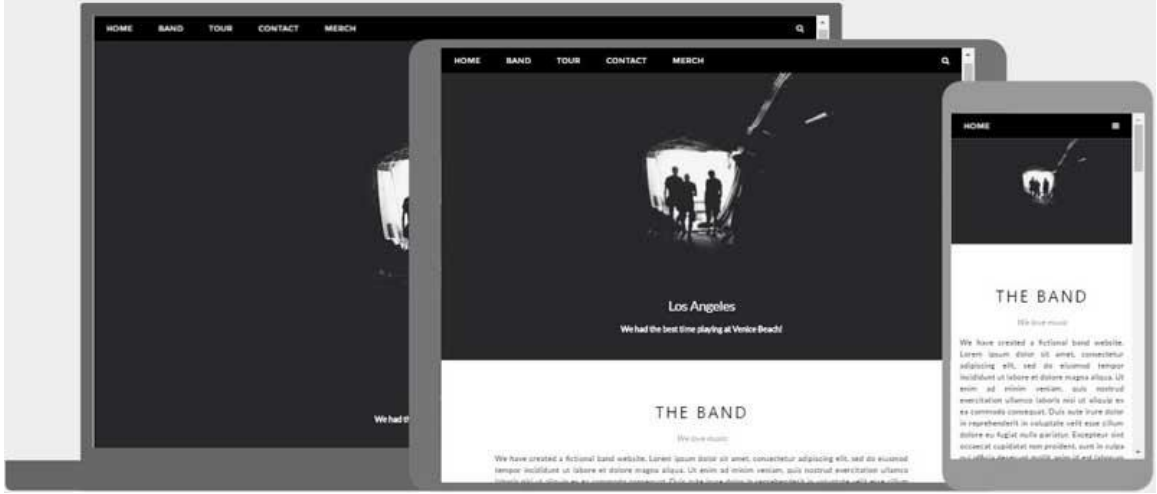
Einheit	Name in CSS	Beschreibung	Beispiel
em	em	Relativ zur Elternelement Schriftgröße	margin: 1em; font-size: 1.5em;
ex	ex	Relativ zur Schriftgröße vom Buchstaben x, sonst wie em	font-size-adjust: .5ex;
Null-Breite	ch	Relativ zur Breite von „0“, sonst wie em	font-size-adjust: .5ch;
Wurzel em	rem	Relativ zur Schriftgröße des Wurzelelements (html)	font-size: 1.5rem;
Prozent	%	Relativ zum Elternelement	height: 10%;
Viewport Breite	vw	Prozentuale Breite zum Viewport	width: 25vw;
Viewport Höhe	vh	Prozentuale Höhe zum Viewport	height: 50vh;
Viewport Minimalabmessung	vmin	min(w, h)/100 * vmin	max-width: 100vmin;
Viewport Maximalabmessung	vmax	max(w, h)/100 * vmax	height: 50vmax;

Box-Modell – Rahmen

- Eigenschaft **`border`** bestimmt Rahmendicke, Rahmentyp und Rahmenfarbe, **`z.B. 25px solid green;`**
- Alle vier Seiten sind ebenfalls einzeln definierbar, **`z.B.`**
  - **`border-top`** **`border-right`**
    `border-bottom` **`border-left`**
- Rahmendicke: **`border-width`**
  - Numerischer Wert (z.B. **`25px`**) oder
    `thin` (dünn), **`medium`** (mittel), **`thick`** (dick)
- Rahmentyp: **`border-style`**
  - **`none`** (transparent), **`dotted`** (gepunktet), **`dashed`** (gestrichelt), **`solid`** (durchgezogen), **`double`** (doppelt durchgezogen), ...
- Rahmenfarbe: **`border-color`**
  - **`transparent`** oder Farbangabe (z.B. **`#FF0000`**)

Responsive Webdesign

Responsives („reagierendes“) **Webdesign** (RWD) ist **Paradigma** bei Erstellung von Webseiten → Webdesign soll auf Anforderungen verschiedener Endgeräte (Desktop, Tablet, Smartphone) reagieren können



Responsive Webdesign

- **Ziel beim Responsive Design**: Webseite erstellen, die für alle Nutzer auf allen Geräten gut aussieht und gut zu bedienen ist
- RWD sollte technisch nur mit HTML und CSS realisiert werden – JavaScript ist in der Regel nicht nötig
- RWD sollte nicht darauf beruhen, auf kleineren Bildschirmen einfach Inhalte wegzulassen, sondern diese anders anzuordnen
- **Beispiel**: Liste von Inhalten
  - Tablets: Bilder werden zunächst schmaler, bei kleineren Bildschirmen werden 3 statt 4 Einträge pro Zeile angezeigt
  - Smartphones:
    - Navigation hinter „Hamburger-Icon“ verborgen 
    - Einträge untereinander angeordnet
    - Links in Fußleiste werden untereinander angeordnet

Viewport

- **Früher**: Webseiten wurden für Desktop-Browser mit fester Breite gestaltet („Dieses Seite ist optimiert für 1024x768 Pixel“)
  - Seiten waren für mobile Endgeräte zu breit
  - Mobile Browser verkleinern gesamte Seite → unlesbare Schrift
- **Einfache Maßnahme**: Angabe des **Viewports**
  - Viewport ist der für Nutzer sichtbare Bereich einer Webseite
  - variiert mit Breite des Bildschirms bzw. Fensters
- HTML 5 erlaubt Autoren, den Viewport selbst festzulegen:
  `<meta name="viewport" content="width=device-width">`
  - **device-width** ist die Breite des genutzten Gerätes, Elemente werden in der Breite auf Bildschirmbreite begrenzt
- Zusätzlich beachten: Keine Elemente mit großer fester Breite verwenden (also z.B. width="1000px"), diese ragen immer über Viewport hinaus

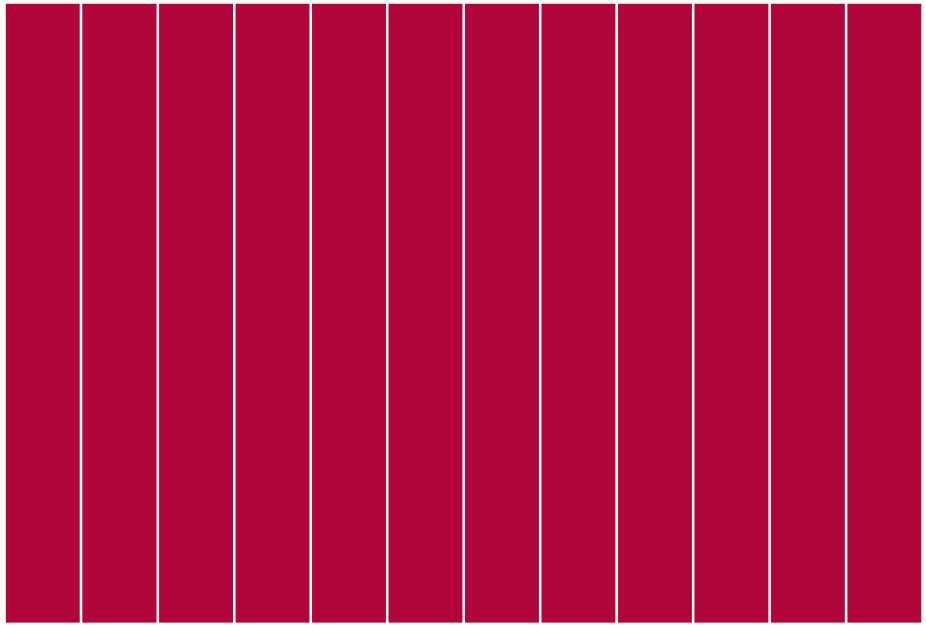
Viewport

Ohne Viewport-Angabe

Mit Viewport-Angabe

CSS Raster Layout (1/2)

- Webseiten werden häufig anhand eines horizontalen Rasters (Grid) gestaltet, die für verschiedene Bildschirmbreiten skaliert werden



12-spaltiges Raster

CSS Raster Layout

Code-Beispiel für CSS Raster

```
.col-1  {width: 8.33%;}
.col-2  {width: 16.66%;}
.col-3  {width: 25%;}
.col-4  {width: 33.33%;}
.col-5  {width: 41.66%;}
.col-6  {width: 50%;}
.col-7  {width: 58.33%;}
.col-8  {width: 66.66%;}
.col-9  {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

Spalten des Rasters (je 8,33% breit)

```
[class*="col-"] {
    float: left;
}
```

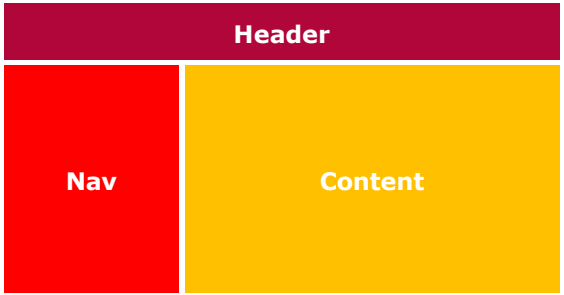
Spalten sollen nebeneinander stehen (**float**)

```
 .row::after {
    content: "";
    clear: both;
}
```

Neue Zeile: **clear** hebt **float** auf

```
...
<div class="row">
  <div class="col-12">Header</div>
</div>
<div class="row">
  <div class="col-3">Navigation</div>
  <div class="col-9">Content</div>
</div>
```

Raster bewirkt, dass alle Elemente der Webseite auf **schmalen** Geräten in der Breite **gleichmäßig skaliert** werden



CSS Media Queries

- CSS 3 führt erstmals **Media Queries** ein
- **@media**-Blöcke erlauben Definition von CSS-Regeln, die nur gelten, wenn bestimmte Bedingung gelten
- **Beispiele**:
  - nur für Bildschirme bis maximale Smartphone-Breite

```
@media only screen and (max-width: 768px) {
   img { width: 100% }
}
```

  - nur für Bildschirme im Querformat

```
@media only screen and (orientation: landscape) { ... }
```

  - nur für Druck

```
@media only print { ... }
```

- Media Queries eignen sich auch für Einbinden externer CSS-Dateien:

```
<link rel="stylesheet" href="druck.css" media="print">
```

CSS Media Queries

Kombination von Media Queries und CSS Raster Layout erlaubt tatsächliches reagierendes Design, z.B. Spalten neben- / untereinander

```
/* Für Desktop / Tablet */
.col-1  {width: 8.33%;}
.col-2  {width: 16.66%;}
.col-3  {width: 25%;}
[...]
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

Spalten nebeneinander

```
/* Für Smartphones */
@media only screen and (max-width: 768px) {
  [class*="col-"] {
      width: 100%;
  }
}
```

Spalten untereinander

- Bildschirmbreiten, an denen sich Gestaltung einer Webseite ändert, nennt man **Breakpoints**

CSS Frameworks

- Für Responsive Design und andere wiederkehrende CSS-Anwendungsfälle gibt es **Frameworks** – Sammlungen von vorgefertigten Gestaltungselementen und Hilfsmitteln
- Bekanntes Framework: **Bootstrap** (von Twitter)
  - enthält neben CSS-Elementen auch eine Reihe nützlicher Javascript-Komponenten
  - Tools für Responsive Design:
    - umfassendes Grid-System mit Media Queries
    - erlaubt Ausblenden von Seiten-Elementen bei bestimmten Bildschirmbreiten
    - Hilfsmittel für responsive Images und Videos
  - bietet Unterstützung für → **CSS Präprozessoren**

Exkurs XML - Einführung

- Eine große Kritik an HTML war immer ein Mangel an Flexibilität
- Der Grund dafür ist die in der **HTML Document Type Definition** (DTD) festgelegte Syntax von HTML
- Autoren von HTML-Dokumenten können HTML-Elementen keine eigene Bedeutung zuweisen oder selbst neue Elemente definieren
  - ist großes Hindernis für die Interaktion mit externen Anwendungen
- Genau hier kommt die **Extensible Markup Language** (**XML)** ins Spiel
  - erlaubt Einführung von neuen und aussagekräftigen Elementen durch die Definition von eigenen DTDs

Exkurs XML - Einführung

- Vorlage für die Entwicklung von XML war, genau wie bei HTML, die komplexe Meta-Markupsprache SGML (Standard Generalized Markup Language), mit der jedes Dokument unterschieden und strukturiert werden kann
- Da XML primär für Webanwendungen entwickelt, wurde nur Teilmenge von SGML zur Definition von XML genutzt, um Interpretation von XML-Dokumenten, Autorenwerkzeuge, Browser, usw. so einfach wie möglich zu halten
- In XML wurden nur absolut notwendige Bestimmungen von SGML umgesetzt, um die vollständige Flexibilität von SGML auch für das WWW zu nutzen

XML – Extensible Markup Language

- **XML-Spezifikation** beschreibt nur die Struktur von Dokumenten
- **XML-Dokument** das den XML-Erzeugungsregeln [1] entspricht, ist auch syntaktisch korrekt, z.B.:
  - Dokument hat genau ein Wurzelelement
  - Elemente haben Start- und End-Tag ("<xy></xy>", Kurzform "<xy />")
  - …
- XML-Dokument das der allgemeinen XML-Spezifikation entspricht und die Regeln und Definitionen in der Grammatik einhält, wird **valide** genannt
- Definition der Grammatik kann z.B. durch ein **XML-Schema** oder eine **DTD** festgelegt werden

[1] http://www.w3.org/TR/REC-xml/#sec-well-formed

XML – Extensible Markup Language

XML

- Ersetzt HTML nicht, aber ergänzt es in Bereichen, wo HTML Schwächen hat, wie z.B. in der semantischen Unterstützung von Interaktionen mit externen Anwendungen
- Allgemeines Objektmodell erlaubt die Entwicklung von Anwendungen mit maschinellem Zugang und zur Manipulation der Daten
  - **Document Object Model – DOM**
- Bietet offenen Standard zur Beschreibung beliebiger Datenstrukturen, was den Austausch über das WWW vereinfacht und eine integrierte Verbindung zwischen Dokument und Anwendung schafft

XSL – Extensible Stylesheet Language

- XML beschreibt nur die Struktur; die Formatierung eines Dokuments wird durch eine Dokument-Formatbeschreibungssprache in Form von Stylesheets bestimmt ähnlich wie bei HTML
- **Extensible Stylesheet Language – XSL** (eigentlich XSL-FO Formatting Objects) dient zur Beschreibung des Layouts von XML-Dokumenten
- XSL wird oft in enger Verbindung mit **XSL-Transformation** – **XSLT** genutzt
- Typische Verwendung:
  - XML-Dokument wird umgewandelt durch XSLT in ein XSL-Dokument, um vom FO-Prozessor als lesbares/druckbares Dokument dargestellt werden zu können

DTD – Document Type Definition

- XML bietet Autoren Möglichkeit zur maßgeschneiderten Definition ihrer Dokumente vermittels eigener Dokumenttypbeschreibung- **Document Type Definitions** – **DTDs**
- Um die **Validität** eines XML-Dokuments überprüfen zu können, wird eine Grammatik benötigt, z.B. definiert als DTD oder XML-Schema
- **XML-Parser** wird benötigt, um XML-Dokumente gemäß zugehöriger DTD zu interpretieren – Deklaration und Regeln der XML-Syntax
- **XML-Schemas** sind komplexer, bieten aber zusätzlich
  - Vererbung,
  - XML-Parsing,
  - ausführlichere Beschreibungen, ...

DTD – Document Type Definition

Beispiel für eine DTD – Zeitung

```
<!DOCTYPE newspaper [
        <!ELEMENT newspaper (article+)>
        <!ELEMENT article (title, introduction, text)>
        <!ELEMENT title (#PCDATA)>
        <!ELEMENT introduction (#PCDATA)>
        <!ELEMENT text (#PCDATA)>
        <!ATTLIST article author CDATA #REQUIRED>
        <!ATTLIST article lecturer CDATA #IMPLIED>
        <!ATTLIST article date CDATA #IMPLIED>
        <!ENTITY newspaper "New York Times">
        <!ENTITY publisher "Arthur Ochs Sulzberger Jr.">
]>
```

Hinweis: Entitäten erlauben die Definition von Konstanten (Sonderzeichen oder häufig verwendetet Textteile)

DTD – Document Type Definition

Beispiel eines XML-Dokuments entsprechend der newspaper.dtd

```
<?xml version="1.0" ?>
<!DOCTYPE newspaper SYSTEM "./newspaper.dtd">
<newspaper>
   <article author="Author 1" date="2023-01-01">
     <title>Title 1</title>
     <introduction>Introduction text …</introduction>
     <text>Main article text</text>
   </article>
   <article author="Author 2" date="2023-01-01">
     <title>Title 2</title>
     <introduction>Introduction text …</introduction>
     <text>Main article text</text>
   </article>
</newspaper>
```

Ist valides XML-Dokument in Bezug auf newspaper.dtd

XML – Verknüpfungskonzept

Erinnerung: HTML Verknüpfungskonzept mit dem Hyperlink-Element <a> ist sehr eingeschränkt:

- HTML-Hyperlink verlinkt nur zwei Ressourcen
- HTML-Hyperlink ist immer unidirektional
- HTML-Hyperlink ist Teil der HTML-Form wo der Link herstammt
- HTML-Hyperlink beeinflusst nicht die Darstellung des Link-Ziels im Browser

XML – Verknüpfungskonzept

Für XML wurden zwei separate Sprachstandards entwickelt, um HTML-Verknüpfungskonzept zu verallgemeinern und für die Definition von weiteren Verbindungen zwischen Informationsressourcen:

- **XML Linking Language** – **Xlink**
  - definiert auch multidirektionale Verbindungen
- **XML Pointer Language** – **XPointer**
  - erweitert die XPath-Spezifikation um auf Teile von XML-Dokumenten zu verweisen
  - Beispiel:
    - xlink:href="music.xml#xpointer(/interpret/album)"

Bereichsspezifische Beschreibungssprachen

Mit dem XML-Metasprachkonzept zur Definition eigener DTD ist der Weg für Entwicklung eigener bereichsspezifischer Beschreibungssprachen frei, z.B.:

- Mathematical Markup Language – **MathML**
- Chemical Markup Language – **CML**
- Synchronized Multimedia Integration Language – **SMIL**
- Scalable Vector Graphics – **SVG**
- …

Da XML mehr ein Werkzeug für Spezialisten ist, kommt der Endnutzer nur in Kontakt mit XML bei verschiedenen Anwendungen oder mit XHTML als XML-konformem HTML

FH Salzburg

VO Web-Technologien

04.10.2023, Oliver Jung

Technik
Gesundheit
Medien