Entwicklung webbasierter Anwendungen

4. Kapitel: CSS

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Beispiele

Quelle: http://www.csszengarden.com

Potential der CSS

- exakte Bestimmung des Erscheinungsbilds
  - ⇨ wichtiger Schritt in Richtung WYSIWYG
  - ⇨ die variable Bildschirmauflösung bleibt ein Problem
- verbesserte Exportierbarkeit aus anderen Tools
  - ⇨ Zukunftsvision: Erstellung von Webseiten mit gewohnten Tools
  - ⇨ kein Fachwissen und keine Tricks mehr erforderlich;
    HTML-Programmierer werden arbeitslos
- Optimierung für verschiedene Ausgabemedien
  - ⇨ Bildschirm, Drucker, TV, Palmtop, ...
- Grundlage für Barrierefreies Webdesign

Arbeitsteilung mit CSS

- saubere Trennung zwischen Inhalt und Form
  - ⇨ Inhalt logisch formatiert in HTML
  - ⇨ Physisches Format und Fein-Layout separat in CSS

?

- Arbeitsteilung Autor / Designer wird möglich
  - ⇨ einheitliche Layouts für große Projekte

?

- Corporate Design kann übernommen werden
  - ⇨ hierarchischer Aufbau (Kaskadierung)
  - ⇨ Übernahme und Abwandlung einer Designvorgabe

Für verschiedene Ausgabemedien

- verschiedene CSS-Dateien in HTML einbinden

```
<link rel="stylesheet" media="screen"
                        href="Bildschirm.css">
<link rel="stylesheet" media="print"
                        href="Drucker.css">
```

- verschiedene Bereiche innerhalb eines CSS

```
@media screen {
  /* Style-Sheet-Definitionen für den Bildschirm */
}
@media print {
  /* Style-Sheet-Definitionen zum Drucken */
}
```

Einbindung von CSS in HTML (1)

- "extern" in eigener CSS-Datei
  - ⇨ kann von mehreren HTML-Dateien genutzt werden

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

Normalfall

gehört in den HTML-Header

- "eingebettet" im HTML-Code
  - ⇨ gilt nur für diese eine HTML-Datei

```
<style type="text/css">
  <!--
     /* ... Style-Sheet-Definitionen ... */
  -->
</style>
```

HTML-Kommentar

CSS-Kommentar

Einbindung von CSS in HTML (2)

- "inline" in jedem HTML-Tag
  - ⇨ gilt nur für dieses eine Objekt

```
<p style="color:red; font-size:36pt;">
      großer roter Text</p>
```

erfordert unbedingt ein schliessendes Tag

  - ⇨ neues Inline-Tag `<span>` zur Markierung eines Teilbereich eines Objekts

einziger Zweck

```
<p>Normaler Textabsatz mit
   <span style="font-style:italic; color:red;">
      rot-kursivem Text
   </span> und wieder normal
</p>
```

Standard-Formate modifizieren

- Definition in CSS

vorzugsweise für Ausgestaltung logischer Formate

```
h3 { font-size:48pt; color:#33FF00; }
p  { font-size:12pt; line-height:16pt;
     font-family:Arial, Helvetica; }
```

- Anwendung in HTML

kein Attribut in HTML

```
<h3>Überschrift 3. Ebene</h3>
<p>einfacher Fließtext in einem Absatz</p>
```

⇨ ohne CSS zeigt der Browser die "schlichte" Version

Browser-Default-Formatierung im Vergleich - ohne CSS

Standard-Formate kontextabhängig

- Definition in CSS

```
h1     { color:red; }
h1 i   { color:blue; font-weight:normal; }
```

d.h. Italic geschachtelt in Header 1

dieses Format gilt nur dort

- Anwendung in HTML

```
<h1>Eine Überschrift mit <i>Style-Sheets</i></h1>
<p>Ein Fließtext mit <i>Style-Sheets</i></p>
```

nicht hier

Eigene Format-Klassen

- Definition in CSS

  aber keine Ableitung wie in OO

  ⇨ Unterklassen für Standard Formate

```
p.Hinweis  { font-size:12pt; color:black; }
p.Fussnote { font-size:8pt; color:black; }
```

  ⇨ allgemein verwendbar

```
.Warnung   { color:#DC0081 }
.Zitat     { color:#00DFCA }
```

- Anwendung in HTML

```
<p class="Hinweis">beachten Sie bitte</p>
<p class="Fussnote">das nur am Rande</p>
<p class="Warnung">Achtung! Aufpassen!</p>
<blockquote class="Zitat">des Pudels Kern
    </blockquote>
```

HTML Attribut *class* stellt den Bezug her

Individuelle Objekt Formate

- Definition in CSS

```
#Block1    { font-weight:bold; font-style:italic; }
#Hotw3     { text-decoration:underline; }
```

- Anwendung in HTML
  - ⇨ jedes Format und jede `id` nur einmal !

Eindeutige Block-IDs kann man auch für JavaScript brauchen

```
<p id="Block1">Extra-Formatierung</p>
<p>Einfacher Text mit <em id="Hotw3">Hotword</em></p>
```

"Hotw3" ergänzt das Format von ; im Konfliktfall mit Vorrang

Pseudo-Formate

- Sonderfall:

  definieren Eigenschaften, die keine Attribute von HTML-Blöcken sind

- Definition in CSS

```
a:link    { color:#FF0000; font-weight:bold; }
a:visited { color:#990000; }
a:active  { color:#0000FF; font-style:italic;}

p:first-line   { font-weight:bold; }
p:first-letter { font-size:36pt; color:red;}
```

Darstellung von Hyperlinks

Man kann nur Brücken schlagen zwischen Ufern die man auseinanderhält. Denn wo es keine Gräben gibt, da gibt es auch keine Unterschiede, und wo es keine Unterschiede gibt, da ist kein Leben.

Lesbarkeit ohne CSS bedenken

- für Browser, die noch keine CSS verstehen
  - ⇨ hilft aber nicht für Netscape 4.x, der CSS falsch interpretiert
  - ⇨ barrierefreies Design: für Screen Reader
- nur logische Standard-Formate verwenden
  - ⇨ damit "schlicht", aber logisch formatieren

    `<p>, <h1>, <h5>, <hr>`
- externe CSS-Datei anbinden
  - ⇨ dort "schönes" Format definieren
    - schlichtes Format bei Bedarf redefinieren
  - ⇨ Hilfs-Objekte des schlichten Formats verbergen
    - z.B. waagrechte Linien `<hr>` mit `display:none;`

Farben und Hintergrundbilder

- Farbattribute

`background-color`	Hintergrundfarbe
`color`	Textfarbe
`border-color`	Rahmenfarbe
`text-shadow`	bisher nicht unterstützt

- Notationen für Farbwerte

`rgb(255,140,0)`	Farbanteile für rot, grün, blau im Bereich 0..255
`rgb(100%,55%,0%)`	Farbanteile im Bereich 0%..100%
`#FF8C00`	Farbanteile hexadezimal
`darkorange`	diverse Farben mit Namen

- Hintergrundbild

  nicht nur für gesamte Seite, sondern auch für einzelne Blöcke

```
background-image:url(bild.gif)
```

Netscape Palette

- 216 "websafe" Farben
  - ⇨ aus Rücksichtnahme auf Surfer mit einfacher Graphikkarte
  - ⇨ wird vom Browser auf allen Plattformen als Systempalette in den RAMDAC geladen
- schlechte Farbabstufung
  - ⇨ "mathematisches" Bildungsgesetz: RGB = (n_R*0x33, n_G*0x33, n_B*0x33)
  - ⇨ 6 Abstufungen für jeden Farbkanal: 0x00, 0x33, 0x66, 0x99, 0xCC, 0xFF

#000000	#000033	#000066	#000099	#0000CC	#0000FF
#330000	#330033	#330066	#330099	#3300CC	#3300FF
#660000	#660033	#660066	#660099	#6600CC	#6600FF
#990000	#990033	#990066	#990099	#9900CC	#9900FF
#CC0000	#CC0033	#CC0066	#CC0099	#CC00CC	#CC00FF
#FF0000	#FF0033	#FF0066	#FF0099	#FF00CC	#FF00FF
#003300	#003333	#003366	#003399	#0033CC	#0033FF
#333300	#333333	#333366	#333399	#3333CC	#3333FF
#663300	#663333	#663366	#663399	#6633CC	#6633FF
#993300	#993333	#993366	#993399	#9933CC	#9933FF
#CC3300	#CC3333	#CC3366	#CC3399	#CC33CC	#CC33FF
#FF3300	#FF3333	#FF3366	#FF3399	#FF33CC	#FF33FF
#006600	#006633	#006666	#006699	#0066CC	#0066FF
#336600	#336633	#336666	#336699	#3366CC	#3366FF
#666600	#666633	#666666	#666699	#6666CC	#6666FF
#996600	#996633	#996666	#996699	#9966CC	#9966FF
#CC6600	#CC6633	#CC6666	#CC6699	#CC66CC	#CC66FF
#FF6600	#FF6633	#FF6666	#FF6699	#FF66CC	#FF66FF
#009900	#009933	#009966	#009999	#0099CC	#0099FF
#339900	#339933	#339966	#339999	#3399CC	#3399FF
#669900	#669933	#669966	#669999	#6699CC	#6699FF
#999900	#999933	#999966	#999999	#9999CC	#9999FF
#CC9900	#CC9933	#CC9966	#CC9999	#CC99CC	#CC99FF
#FF9900	#FF9933	#FF9966	#FF9999	#FF99CC	#FF99FF
#00CC00	#00CC33	#00CC66	#00CC99	#00CCCC	#00CCFF
#33CC00	#33CC33	#33CC66	#33CC99	#33CCCC	#33CCFF
#66CC00	#66CC33	#66CC66	#66CC99	#66CCCC	#66CCFF
#99CC00	#99CC33	#99CC66	#99CC99	#99CCCC	#99CCFF
#CCCC00	#CCCC33	#CCCC66	#CCCC99	#CCCCCC	#CCCCFF
#FFCC00	#FFCC33	#FFCC66	#FFCC99	#FFCCCC	#FFCCFF
#00FF00	#00FF33	#00FF66	#00FF99	#00FFCC	#00FFFF
#33FF00	#33FF33	#33FF66	#33FF99	#33FFCC	#33FFFF
#66FF00	#66FF33	#66FF66	#66FF99	#66FFCC	#66FFFF
#99FF00	#99FF33	#99FF66	#99FF99	#99FFCC	#99FFFF
#CCFF00	#CCFF33	#CCFF66	#CCFF99	#CCFFCC	#CCFFFF
#FFFF00	#FFFF33	#FFFF66	#FFFF99	#FFFFCC	#FFFFFF

Schrift

- `font-family:`
  `Arial, Helvetica, "Times New Roman"`
  `serif, sans-serif, cursive, fantasy, monospace`
- `font-style:`
  `italic, normal`
- `font-size:`
  `12pt, 3em, 1.5cm, large`
- `font-weight:`
  `bold, bolder, lighter, 100 .. 900`
- `font:`
  kompakte Kombination o.g. Attributwerte

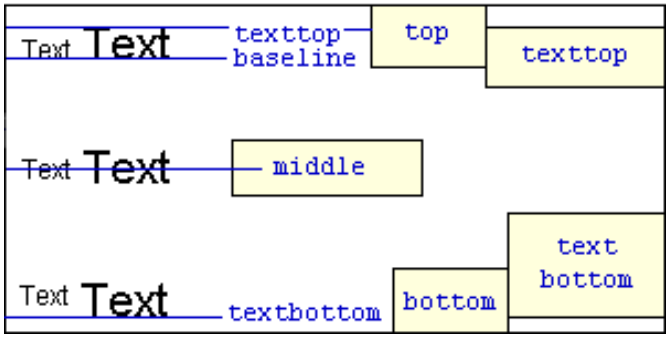
Ausrichtung und Rand

Ausrichtung

- `text-align:`
  `left, center, right, justify`
- `vertical-align:`
  `top, middle, bottom, text-top, text-bottom`
- `text-indent` Texteinrückung in Längenmaß
- `line-height` Zeilenhöhe in Längenmaß

Rand

- `border[-top, -left, -right, -bottom]-width`
  `(z.B. border-left-width, border-width)`
- `border[-top, -left, -right, -bottom]-style:`
  `hidden, dotted, dashed, solid, double, groove, ridge, inset, outset`

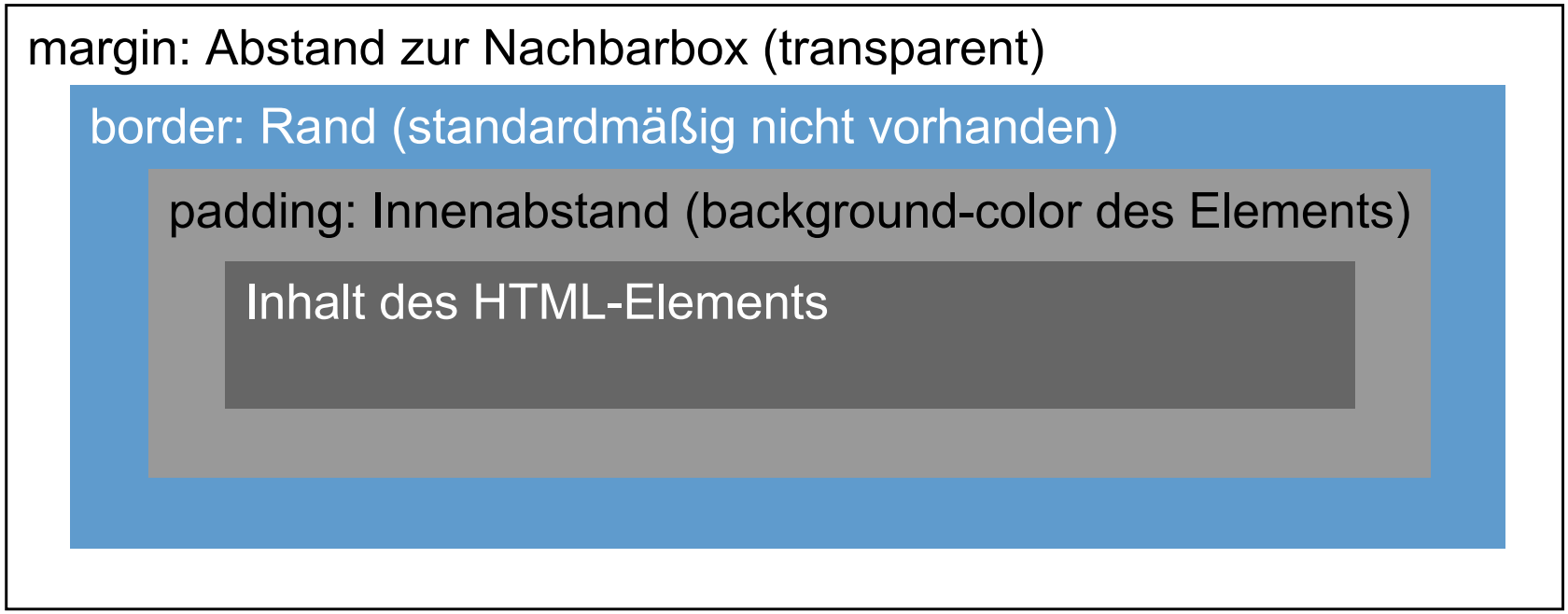


Quelle: SelfHTML

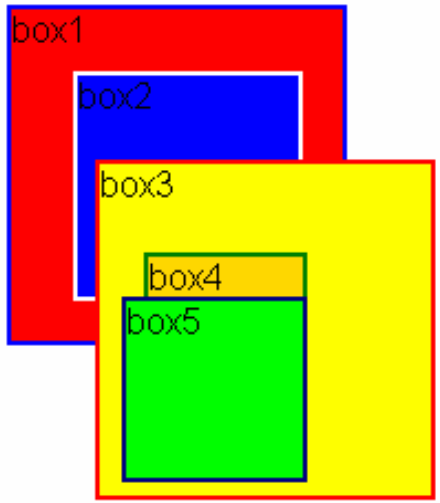
Aussen- und Innenabstand

die Standardwerte sind browserabhängig, deshalb vollständig spezifizieren!

- `margin, margin-top, margin-bottom, margin-left, margin-right` Aussenabstand in Längenmaß
- `padding, padding-top, padding-bottom, padding-left, padding-right` Innenabstand in Längenmaß



Platzierung und Tiefenstaffelung



- beliebige Platzierung mit Verdeckung

  `position`

  `absolute` (bezogen auf Elternelement),

  `relative` (bezogen auf ursprüngliche Position),

  `fixed` (absolute Positionierung, gemessen am Browserfenster. Bleibt beim Scrollen stehen

  `static` (normaler Elementfluss; Normaleinstellung))

  `top,` (bzw. `bottom`), `left,` (bzw. `right`), `width, height`

  ⇨ mit JavaScript dynamisch änderbar ⇨ Animation

nur von wenigen Browsern unterstützt

- Tiefenstaffelung explizit mit `z-index:3`
  - ⇨ je größer die Zahl, desto weiter vorne
  - ⇨ Elemente ohne `z-index` (entspricht z-index=0)

    entsprechend der Reihenfolge in der HTML-Datei
    - vor allen Elementen, die nicht absolut positioniert sind
    - oben in der Datei ⇨ im Hintergrund
    - unten in der Datei ⇨ im Vordergrund

Zeigen und Verbergen

- Anzeige(-art) bzw. Nichtanzeige ohne Platzhalter
  (folgende Blöcke verschieben sich)

```
display:
    block, inline, none
```

Auf- und Zuklappen von Unterpunkten im Inhaltsverzeichnis mit JavaScript

- Anzeige bzw. Nichtanzeige mit Platzhalter
  (folgende Blöcke bleiben stehen)

```
visibility:
    visible, hidden
```

Layouthilfsmittel

Bausteine für
Barrierefreies Webdesign

- Block aus dem Textfluss herausnehmen

  `float:`

  `left, right, none`

  verlangt Festlegung von `width`

- Fortsetzung unterhalb eines `float`-Blocks

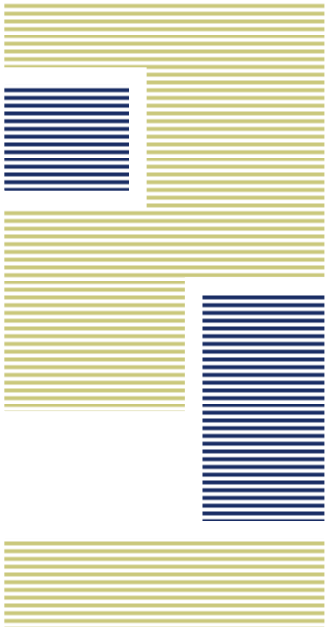
  `clear:`

`left`	unterhalb des letzten links ausgerückten Blocks
`right`	unterhalb des letzten rechts ausgerückten Blocks
`both`	unterhalb des letzten ausgerückten Blocks

- wenn der Inhalt größer ist als der Block

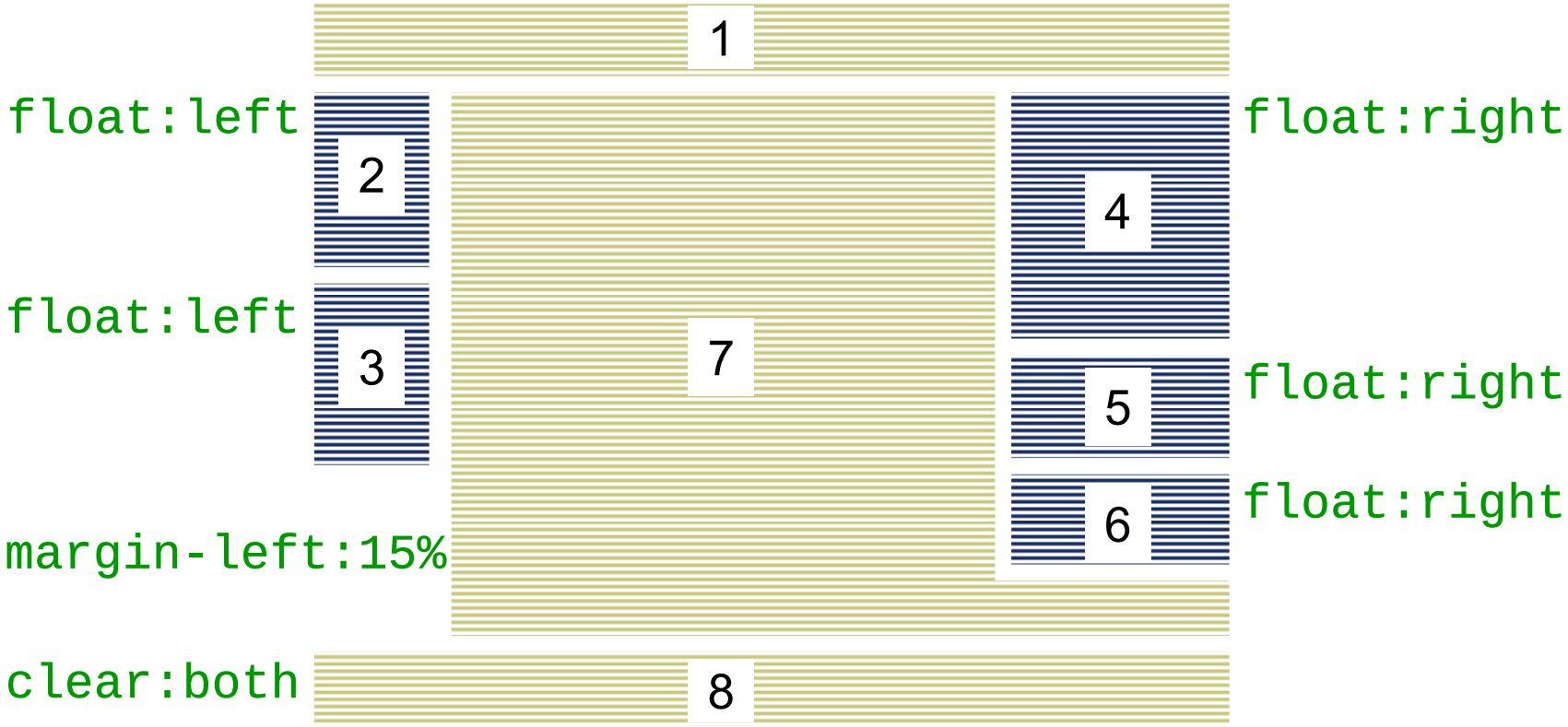
  `overflow:`

`visible`	Block vergrößern
`hidden`	Inhalt beschneiden
`scroll`	Inhalt verschiebbar mit Scroll-Balken



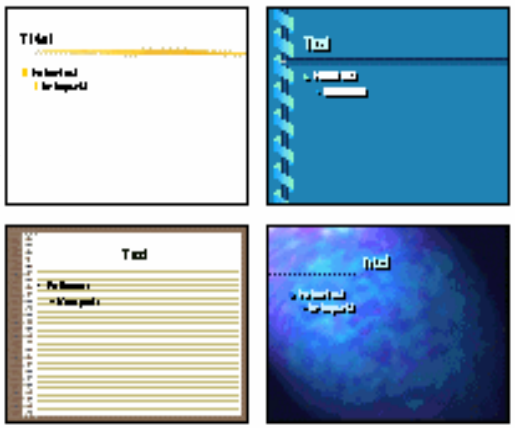
Layout-Beispiel

3-spaltig mit Kopf- und Fußzeile



Sinn einer Format-Hierarchie

1. Corporate Identity ⇨ Corporate Design
   - ⇨ Firmenlogo, Designrahmen
2. einheitl. Erscheinungsbild des Dokuments
   - ⇨ vgl. PowerPoint: Design übernehmen
   - ⇨ Farben, Schriften, Hintergrund, Ausrichtung
3. eine Auswahl von Layout-Typen für Seiten
   - ⇨ vgl. PowerPoint: Folienlayout
   - ⇨ 0/1/2 Textblöcke, mit/ohne Bild, hor./vert. geteilt
4. Besonderheiten der einzelnen Seite
   - ⇨ Abweichung vom Layout-Typ
5. individuelles Format einzelner Objekte



Realisierung einer Format-Hierarchie

vgl. Klassen-Hierarchie in OO

- CorporateDesign.css — kein Verweis

- DokumentDesign.css

```
@import ("CorporateDesign.css")
```

— extern

- SeitenLayouts.css

```
@import ("DokumentDesign.css")
```

— extern

- Seite3.html

```
<link href="SeitenLayouts.css" ...>
```

— extern

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

— embedded für diese Seite

```
<p style="color:red"> Block </p>
```

— inline für einzelne Objekte

Mehrfache Redefinition desselben Attributs für dasselbe Objekt ist möglich !

Auflösung von Konflikten

bei mehrfacher widersprüchlicher Definition

- Vereinigungsmenge aller Definitionen für ein Attribut eines Objekts bilden
  - ⇨ falls leer: vom umschließenden Objekt (parent) erben
  - ⇨ falls leer: Standardwert nehmen
- Sortieren nach
  - ⇨ Gewichtung (! important) — erstes Kriterium
  - ⇨ Herkunft (Autor vor Leser)
  - ⇨ Spezialisierungsgrad
    - Individuell (id)
    - vor kontextabh. Klasse (p.Hinweis)
    - vor allgem. Klasse (.Warnung)
    - vor redefiniertem Standard Format (p)
  - ⇨ Reihenfolge der Definition — letztes Kriterium
    - inline vor embedded vor extern