Das TYPOlight CSS-Framework

Aufgaben eines CSS-Frameworks

- CSS-Reset
  - Vereinheitlichung der Darstellung in allen Browsern
  - Zurücksetzen der proprietären Abstände und Formatierungen
  - Cross-Browser-Formatierung der grundlegenden Elemente
- CSS-Layout
  - Aufteilung einer Seite in verschiedene Layoutbereiche
  - Kombination aus HTML- und CSS-Code
  - Browser-spezifische CSS-Fixes sind enthalten
- CSS-Grid
  - Implementierung eines Rasters zur Anordnung von Elementen
  - Aufteilung der Layoutbereiche in Spalten (ähnlich Printmedium)

Das TYPOlight CSS-Framework

- CSS-Reset
  - Rudimentär vorhanden in der typolight.css
  - Demnächst Veröffentlichung eines nicht nur für TYPOlight geeigneten vollständigen CSS-Resets (vgl. Templates)
- CSS-Layout
  - In TYPOlight vollständig implementiert
  - Kompatibel mit allen Browser und dem IE ab Version 6
  - Konfiguration im Backend: Layout → Seitenlayouts
- CSS-Grid
  - Nicht vorhanden, da je nach Anforderung unterschiedlich
  - Implementierung in TYPOlight ist Gegenstand dieses Workshops

CSS-Layouts in TYPOlight

- Standardmäßig bis zu 5 Layoutbereiche
  - Kopf- und Fußzeile (#header, #footer)
  - Ein bis drei Spalten (#left, #main, #right)
- Erweiterung um beliebig viele eigene Layoutbereiche
  - Drei vorgegebene Positionen
  - Individuelle Positionierung durch Templateanpassung
- Zuständige Stylesheets
  - Inline-CSS pro Seite
  - system/typolight.css
  - system/iefixes.css (IE bis einschließlich Version 7)

HTML-Grundgerüst

```
<body>
<div id="wrapper">

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

    <div id="container">
            <div id="left"></div>
            <div id="right"></div>
            <div id="main">
                <div id="clear"></div>
            </div>
    </div>

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

</div>
</body>
```

CSS-Reset

```
body {
    margin:0;
    padding:0;
    font-size:100.01%;
    text-align:left;
    position:relative;
}
select,input,textarea {
    font-size:99%;
}
form {
    margin:0;
    padding:0;
}
img {
    border:0;
}
```

Inline-CSS

```
#wrapper {
    width:960px;
    margin:0 auto;
}
#header {
    height:330px;
}
#left {
    width:240px;
}
#main {
    margin-left:240px;
}
#footer {
    height:110px;
}
```

system/typolight.css

```
#left {
    float:left;
}
#right {
    float:right;
}
#main {
    width:auto;
    position:relative;
}
.inside {
    position:relative;
    text-align:left;
}
.block {
    overflow:hidden;
}
```

system/typolight.css

```
.clear,#clear {
    height:0.1px;
    font-size:0.1px;
    line-height:0.1px;
    clear:both;
}
.invisible {
    width:0px;
    height:0px;
    left:-1000px;
    top:-1000px;
    position:absolute;
    overflow:hidden;
    display:inline;
}
```

system/iefixes.css

```
* html .block {
    overflow:auto;
    zoom:1;
}
* html #container,* html .mod_article {
    zoom:1;
}
* html #left,* html #right {
    display:inline;
}
* html a,* html a:hover {
    background-color:transparent;
}
* html i,* html em {
    overflow:visible;
    display:inline-block;
}
*:first-child+html #main {
    position:static;
}
```

Prozentuales Grid-System

- Vorteile
  - Keine feste Breite notwendig
  - Geeignet für „Liquid Layouts“
- Nachteile
  - Kein dreispaltiges Layout möglich (Rundungsfehler bei 33.333% möglich)
  - Box-Model-Problem bei z.B. 2 Pixel Randabstand
- TYPOlight Backend
  - Nutzt Grid-System mit zwei Weiten (50% und 100%)

- 10% (10 Spalten)
- 20% (5 Spalten)
- 25% (4 Spalten)
- 30% (Kombination mit 70%)
- 36% (goldener Schnitt)
- 40% (Kombination mit 60%)
- 50% (2 Spalten)
- 60% (Kombination mit 40%)
- 64% (goldener Schnitt)
- 70% (Kombination mit 30%)
- …

Grid-System mit fester Breite

- Vorteile
  - Mehr Kombinationen als beim prozentualen Grid-System
  - Dreispaltiges Layout möglich
- Nachteile
  - Feste Breite notwendig
  - Spalten müssen für jede Breite neu berechnet werden

- 80px (12 Spalten)
- 160px (6 Spalten)
- 240px (4 Spalten)
- 320px (3 Spalten)
- 400px (goldener Schnitt)
- 480px (2 Spalten)
- 560px (goldener Schnitt)
- 640px
- 720px
- 800px
- …

960-Pixel-Grid

- Mathematisch sinnvoll
  - 960 lässt sich durch 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 32, 40, 48, 60, 64, 80, 96, 120, 160, 192, 240, 320 und 480 teilen
  - Typischerweise Aufteilung in 12 oder 16 Spalten
- Sehr populär
  - Umfassende Online-Dokumentation
  - Viele Beispiele aus der Praxis
  - Onlinerechner für abweichende Breiten (960.gs)
- Einschränkungen
  - Geeignet ab einer Auflösung von 1024x768 Pixel

960-Pixel-Grid in TYPOlight

```
/* Elemente floaten */
.g1,.g2,.g3,.g4,.g5,.g6,.g7,.g8,.g9,.g10,.g11,.g12 {
    float:left;
    margin-left:10px;
    margin-right:10px;
    display:inline;
}

/* Breiten festlegen */
.g1 { width:60px; }
.g2 { width:140px; }
.g3 { width:220px; }
.g4 { width:300px; }
.g5 { width:380px; }
.g6 { width:460px; }
.g7 { width:540px; }
.g8 { width:620px; }
.g9 { width:700px; }
.g10 { width:780px; }
.g11 { width:860px; }
.g12 { width:940px; }
```

Zusätzlicher Komfort in TYPOlight

```
/**
 * Inhaltselemente in einem Artikel erhalten grundsätzlich einen
 * Randabstand, damit wir nicht jedem einspaltigen Element extra
 * die Klasse „g12“ zuweisen müssen.
 */
.mod_article>.block {
    margin-left:10px;
    margin-right:10px;
}

/**
 * Das gilt jedoch nicht für Inhaltselemente innerhalb eines
 * Artikels, der selbst ein Teil des Grid-Systems ist, da sonst
 * insgesamt 20px Abstand vorhanden wären.
 */
.g1 .block,.g2 .block,.g3 .block,.g4 .block,
.g5 .block,.g6 .block,.g7 .block,.g8 .block,
.g9 .block,.g10 .block,.g11 .block,.g12 .block {
    margin-left:0;
    margin-right:0;
}
```

Zusätzlicher Komfort in TYPOlight

```
/**
 * Die einzige Ausnahme sind Inhaltselemente, die innerhalb
 * eines Grid-Artikel selbst wieder eine Grid-Klasse erhalten.
 * Dank der „Komfort-Formatdefinitionen“ haben diese Elemente
 * standardmäßig keinen Randabstand, was durch die folgenden
 * Anweisungen ausgeglichen werden kann.
 */
.gr {
    margin-right:20px !important;
}


/**
 * HINWEIS: Viele Grid-Systeme gehen hier den umgekehrten Weg
 * und ziehen den doppelten Randabstand vom ersten und letzten
 * verschachtelten Element jeder Zeile (oft „alpha“ und „omega“
 * genannt) ab. Dank der „block“-Klassen ist in TYPOlight ein
 * komfortablerer Ansatz möglich.
 */
```

Grundsätzliche Überlegungen

- Artikel floaten
  - Sinnvoll z.B. für Seiten mit durchgehend 2 oder 3 Spalten
  - Es spielt keine Rolle, welche und wie viele Inhaltselemente in dem Artikel enthalten sind
  - Kein extra Markup für das Clearing der Floats notwendig
- Inhaltselemente floaten
  - Absolute Flexibilität bei der Ausrichtung der Elemente
  - Allerdings zusätzliches Clearing-Markup notwendig (erst mit dem IE8 ist dieses Problem auch bei Microsoft behoben worden)
  - Unter Umständen wird jedoch sowieso ein Trennelement verwendet (z.B. <hr />), das das Clearing übernehmen kann