The TYPOlight CSS framework

Functions of a CSS framework

- CSS reset
  - Consistent element rendering in all browsers
  - Resetting of proprietary margins and formatting
  - Cross-browser formatting of the basic elements
- CSS layout
  - Division of a page into several layout sections
  - Combination of HTML and CSS code
  - Includes browser-specific CSS fixes
- CSS grid
  - Implementation of a grid system to arrange elements
  - Division of the layout sections into columns (cp. print media)

Functions of a CSS framework

- CSS reset
  - Rudimentary available in the typolight.css file
  - A full CSS reset style sheet that is not only suitable for TYPOlight will be released soon (cp. iNet Robots templates)
- CSS layout
  - Fully integrated in TYPOlight
  - Compatible with all major browsers and IE from version 6
  - Configurable in the back end: Layout → Page layouts
- CSS grid
  - Typically specific requirements, therefore not integrated
  - Implementation in TYPOlight is subject of the workshop

Functions of a CSS framework

- Up to 5 default layout sections
  - Header and footer (#header, #footer)
  - One to three columns (#left, #main, #right)
- Unlimited custom layout sections possible
  - Three positions available
  - Individual positions can be added in the template
- Corresponding style sheets
  - Inline CSS
  - system/typolight.css
  - system/iefixes.css (IE until version 7)

HTML markup

```
<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;
}
```

Percent grid system

- Advantages
  - No static width required
  - Suitable for “liquid layouts”
- Disadvantages
  - Does not allow three-column layouts (rounding error at 33.333% in some browsers)
  - Additional borders or padding cause box-model issues
- TYPOlight back end
  - Uses a grid system with two widths (50% and 100%)

- 10% (10 columns)
- 20% (5 columns)
- 25% (4 columns)
- 30% (combined with 70%)
- 36% (golden section)
- 40% (combined with 60%)
- 50% (2 columns)
- 60% (combined with 40%)
- 64% (golden section)
- 70% (combined with 30%)
- …

Grid system with a fixed width

- Advantages
  - More combinations than the percent grid system
  - Supports three-column layouts
- Disadvantages
  - Static width required
  - Recalculation needed if the overall width changes

- 80px (12 columns)
- 160px (6 columns)
- 240px (4 columns)
- 320px (3 columns)
- 400px (golden section)
- 480px (2 columns)
- 560px (golden section)
- 640px
- 720px
- 800px
- …

960 pixel grid

- Mathematically feasible
  - 960 can be devided by 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 and 480
  - Typically division into 12 or 16 columns
- Very popular
  - Comprehensive online documentation
  - Many practical examples
  - Online calculator for different widths (960.gs)
- Restrictions
  - Applicable for screen resolutions from 1024x768 pixels

960 pixel grid in TYPOlight

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

/* Define widths */
.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; }
```

Additional comfort in TYPOlight

```
/**
 * Give content elements inside articles a default margin so we
 * do not have to assign class “g12” to every one-column element.
 */
.mod_article>.block {
    margin-left:10px;
    margin-right:10px;
}

/**
 * However, exclude content elements inside articles which are
 * part of a grid system themselves, otherwise there would be a
 * 20px margin.
 */
.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;
}
```

Additional comfort in TYPOlight

```
/**
 * The only exception are content elements within a grid article
 * which additionally get a grid class. Thanks to the comfort
 * CSS definitions, these elements do not have a margin, which
 * can be corrected by the following definition.
 */
.gr {
    margin-right:20px !important;
}


/**
 * NOTE: Most grid systems do it the other way round and subtract
 * the double margin from the first and last element in each row
 * (often referred to as “alpha” and “omega”). The “block” class
 * in TYPOlight, however, allows for a more comfortable approach.
 */
```

Fundamental thoughts

- Floating articles
  - E.g. suitable for pages with two or three continuous columns
  - The number and type of content elements inside the articles does not matter at all
  - No extra clearing markup required
- Floating content elements
  - Absolute flexibility in arranging elements
  - Additional clearing markup required (Microsoft only solved this problem in IE8)
  - Possibly there already is a separator (e.g. <hr />) that can be used as clearing element