University of Glasgow

Cascading Style Sheets

Internet Technology

ITECH

XHTML
CSS
JavaScript
ORM
Django
SQL
Python
Separation of Concerns

Web Development Babel

- Whilst developing web applications, the **minimum useful set of technologies** for a sufficiently complex application is **five:**
  - **Server-side Language (PHP, Ruby, Python, Java, .Net etc)**
  - **Data Language (SQL)**
  - **Client-side Language (JavaScript)**
  - **Content Markup Language (XHTML)**
  - **Style Markup Language (CSS)**
- This turns out to be a headache when developing, especially when the languages mix together in one webpage/file.
- Maintenance becomes particulary difficult and codebase is especially fragile to change

Separation of Concerns

Separating the Presentation, Content and Processing on the client side

Style on the Web

- **Decisions of Style:** Most aspects about any element of a web page can be controlled:
  - position, color, size, font etc
- This can be achieved in any number of ways:
  - by describing the page in XML and then using XSL to generate formatted XHTML
  - by using Cascading Style Sheets in combination with XML
  - by using **Cascading Style Sheets** in combination with **XHTML**

Benefits of CSS

- Cascading Style Sheets (CSS) is a method of **separating a document's structure and content from its presentation**
- CSS allows for a much **richer document appearance** than (X)HTML alone
- CSS can save time as the appearance of the entire document can be **created and changed in just one place**
- CSS can improve load times as it **compactly stores the presentation concerns** of a document in one place instead of being repeated throughout the document

CSS Guide
&
Reference

or visit

w3schools

Cascading Style Sheets

- Stylesheets describe the rendering of html elements
  - they specify stylistic aspects of **individual elements** or **all elements** of a particular kind
  - A CSS consists of a set of **formatting rules,** which are specified in the following way:

```
selector {
    property1: value1;
    property2: value2;
    …
}
```

```
h3 {
    color: yellow;
    size: 18px;
    …
}
```

  - **selector** indicates the element (or set of), **property** refers to the stylistic aspect, and **value** is the specific configuration.

Find and Apply Pattern

```
p {
    font-size: 12pt;
    font-face: “Verdana”; }

h1, h2, h3 {
    color: red;
    font-size: 18px; }

*{ text-align: left; }

#menu a {
    padding: 45px 25px 0px 25px;
    border: none;
    height: 80px; }
```

Apply to all <p> elements

Apply to all <h1>, <h2>, <h3> elements

Apply to all elements

Apply to all <a> elements with id=“menu”

Value and Units

- **Units** affect the colors, distances, and sizes of a whole host of properties of an element's style
- Numbers
  - can be integers or real numbers
- Percentages
  - real number followed by %
  - generally relative to some other number
  - e.g. font-size:90% of the default or inherited value
- Color
  - named color (e.g. 'red'), functional rgb(0,0,255), or hexadecimal RGB codes (#FF0000)

Value and Units

- Length Units
  - inches (in), centimeters (cm), millimeters (mm), points (pt – 72pt to 1 inch), and picas (pi – 1pi = 12pt)
- Relative Length Units
  - em is relative to the given font-size value
  - e.g. font-size is 14px, 1em = 14px
  - ex is relative to the size of a lowercase x for the given font family
  - px is (should be) the size of a pixel on the monitor
  - px is generally the recommended unit to use

Using CSS with (X)HTML

Inline CSS Specification

- **Inline**: style information is added directly to one particular element using its **style** attribute
- CSS syntax is used after the **style** attribute in an HTML tag

```
<h3 style="color: yellow; font-size: 18pt">
```

- This only affects this element, and others of the same type are not effected.
- Useful to **override** existing style, but **breaks** the **separation** of **content and presentation**

Embedded CSS Specification

- **Embedded**: Style rules can be specified in the <head> section of the document
- These rules will be applied to the entire document

```
<html>
    <head>
        <style>
            h3 {  color: yellow;    font-size: 18pt;    }
        </style>
    </head>
    <body>
        … <h3> yellow font size 18 </h3> …
    </body>
</html>
```

External CSS Specification

- **External**: In a separate document which can be shared by several pages. The file extension is ".css"

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

- This is generally the best method in terms of:
  - Separation of concerns
  - Maintenance
  - Performance*

Specialization of Presentation

- **Class** and **ID selectors** can be used for finer control
- This involves more planning/effort with document markup
  - But can result in a better user experience
  - It is also very important for manipulating elements in Javascript
  - The effort also pays off if you use libraries like JQuery
- Class selectors work on a set of specified elements through the **class attribute**
- While ids, provide a way to stylize unique elements through the **id attribute**

Class Selectors

- Class selector allow you to style items with the same (X)HTML element differently
- They work when the **class attribute** of an HTML tag is given a **name**
- The **dot (.)** operator is used to define the class

```
<style>
    *.warning {font-weight: bold;}
</style>

<p class="warning">This text will be displayed in bold.</p>

<p>This text will NOT be displayed in bold.</p>
```

ID Selectors

- Similar to class but they define a special case for an element
  - IDs are meant to be unique and only used once
  - However, browsers are not particularly fussy about enforcing the uniqueness of identifiers
- The **hash symbol (#)** is used to specify a unique ID

```
<style>
    #first-para { font-weight: bold; }
</style>

<p id="first-para">This paragraph will be bold-faced</p>

<p>This will not be bold</p>
```

Restricted Class and ID Selectors

- All h2 elements within the class red should be coloured red

```
<style>   .red h2 { color : red; }    </style>
<body>   <div class="red"><h2>Heading</h2> </div>    </body>
```

- All h2 elements whose class is red should be coloured red

```
<style>   h2.red { color : red; }    </style>
<body>   <h2 class="red">Heading</h2>     </body>
```

- All h2 elements within an element with ID red should be coloured red

```
<style>   #red h2 { color : red; }    </style>
<body>   <div id="red"><h2>Heading</h2> </div>    </body>
```

Descendent Selectors

- Elements that are descended from a particular element are styled according to the rule of the **descendent selector**
  - This means that the rules will be applied to a set of elements in one context but not in another

```
<style>
      p em { color: red; font-weight: bold;  }
</style>
<body>
      <p>this will be the default color
      <em>this will be coloured red and bold</em>
       back to the default colour</p>
</body>
```

Inheritance of Style

- The order of application of styles is through inheritance
- Styles are applied not only to a specified element, but also to its descendants
  - For example, below will inherit the style of its parent <p>

```
<style>
    p { color: red;
        font-weight: bold  }
</style>
<body>
<p>happily grey <em>really emphasizing redness</em></p>
</body>
```

Specificity of Style

- Sometimes multiple rules apply to the same element
- CSS uses a weighting scheme to ensure that there are predictable outcomes with conflicts of style
  - For every ID attribute value given in the selector, add 1,0,0
  - For every class attribute value, attribute selection, or pseudo-class given in the selection, add 0,1,0
  - For every element and pseudo-element given in the selector, add 0,0,1

```
h1 {color: red;}                    /* specificity = 0,0,1 */
body h1 {color: green;}             /* specificity = 0,0,2 */
#content h2 {color: silver;}        /* specificity = 1,0,1 */
h2.grape {color: purple;}           /* specificity = 0,1,1 */
h2 {color: silver;}                 /* specificity = 0,0,1 */
```

  - Calculator: https://specificity.keegan.st/

The Cascade in CSS

- However, sometimes there is still a conflict between two or more rules.
  - i.e. if they all have the same weight
- CSS is based on a method of causing **styles to cascade together**, which is made possible by combining inheritance, specificity and order
- The purpose of "**cascading**" is to find one winning rule among a set of rules that apply to a given element

Page Layout

GOOGLE

Sponsored links live on the right margin of the page and above search results marked by a colored background.

FACEBOOK

Engagement ads live on the right side of page, footnoted by names of friends who have "liked" or interacted with the ad.

If a member "likes" the brand, futu messages from the company might appe inside the member' newsfeed at no extr cost to the advertise

TWITTER

Promoted tweets show up in Twitter search results and will appear in regular Twitter feeds based partly on whom a member follows. Users can retweet ads at no extra cost to the advertiser.

as their product's "service missi
cial K's page dispenses nutritio
Nature Valley ruminates about
parks and nature photography.

Page Layout

- Layout of major elements on a webpage (e.g. columns, navigation bars, sidebars, headers and footers) can be specified using CSS
- In bygone days, tables were heavily used for layout
  - simple to use for simple tasks
  - painful for complex layout
  - tables are meant for content, not layout
- The preferred solution is to divide a page into a collection of (division/section) elements
  - ** … **
  - …

Floating

CSS floating properties allow you to **float** elements horizontally.

Elements can be floated: **left** and **right**, but not up and down!

Elements after the floating element will flow around. So if screen size changes elements will move down.

```
<head><style type="text/css" media="screen">
        .thumbnail {
                float:left;
                width:110px;
                height:90px;
                margin:5px;       }
  </style> </head>

<body>

<img class="thumbnail" src="klematis_small.jpg”>
<img class="thumbnail" src="klematis2_small.jpg”>
<img class="thumbnail" src="klematis3_small.jpg">

</body>
```

Positioning

CSS positioning properties allow you to **position** an element.

Elements can be positioned using: **top**, **bottom**, **left** and **right** properties.

There are four different ways to position: static (default), **fixed**, **relative** and **absolute**.

```
<style type="text/css" media="screen">
        div {
                border: 1px solid #999;
                margin: 20px;    }

        div.fixed {
                position: fixed;
                top: 30px;
                right: 5px;         }

        div.Relative {
                position: relative;
                top: -50px;}

        div.Absolute {
                position: absolute;
                left: 100px;
                top: 150px;         }

 </style>
```

Floating & Positioning

```
<head>
 <style type="text/css" media="screen">
  div { border: 1px solid #999;
        margin: 20px;}
 </style>
</head>
<body>
 <div id="page">
  <div id="header"> <h1>header</h1> </div>
  <div id="navbar"> <p>navbar</p> </div>
  <div id="content"> <p>content</p> </div>
  <div id="sidebar"> <p>sidebar</p> </div>
  <div id="footer"> <p>footer</p> </div>
 </div> <!-- end of page -->
</body>
```

Floating and Positioning

```
#content {
    float: left;
    width: 67%;
    background: #eee;
    margin-top: 0;
    margin-right: 1.67em;
    border-right: 1px solid black;
    padding-top: 0;
    padding-right: 1em;
    padding-bottom: 20px;
  }
```

Floating and Positioning

Due to the natural flow layout, the footer block has appeared in the incorrect location.

Floating and Positioning

```
#footer {
    clear: both;
    padding-bottom: 1em;
    border-top: 1px solid #333;
    text-align: center;
  }
```

The Box Model

The Box Model

Every element generates one or more rectangular **element boxes** which houses the content. The element box is surrounded by optional amounts of **padding**, **borders**, and **margins**.

Box Model Properties

CSS Properties

- The full list of properties can be found: https://www.w3.org/TR/CSS/#css
- Compact CSS Cheatsheats are useful: http://www.lesliefranke.com/files/reference/csscheatsheet.html

background-color	**border-width**	**font-family**	**height**	**size**
text-align	**width**	**color**	**font-size**	**margin**
padding	**list-style**	**position**	**text-decorations**	**~100+ more!**

Summary

- Separation of concerns is a good principle to adopt
  - simple examples usually don’t benefit
  - effort is worth it as complexity increases
- CSS is a powerful method of specifying the style of web pages
  - separates presentation from structure and content