CIS 3.5, Spring 2010
Lecture I.2

Topics:

- CSS
  - Introduction
  - Motivation
  - Advantages
  - Implementation
  - Classes & IDs
  - and
  - General Model
  - Validation

CSS Introduction

- HTML was originally designed as a simple way of presenting information, with the aesthetics less important than the content. Of course as the web grew, presentation become much more important.

- New tags were created to allow greater presentation freedom (<font>). HTML coders quickly noticed that they were retyping the same old tags over and over leading to huge HTML files and above all, time consumption and frustration.

- Imagine you've just designed a two hundred page web site for a client, who at the last minute decides the font is a little two small or the typeface should be serif instead of sans-serif?

Motivation

- In 1996 (and 1998) CSS (Cascading StyleSheets) became a formal recommendation of the W3C. Stylesheets act as partners to HTML/XHTML documents; taking care of all layout, fonts, colors and the overall look of a page/site.

- With CSS the idea is to leave most of the formatting out of your HTML/XHTML files and use only nice structural elements (like headings, paragraphs and links). Thus separating structure and presentation.

- If you decide to change the look of a site, you modify the CSS file (style sheet) and all the HTML/XHTML pages reading from that file will display differently. This makes maintenance of your design much easier.

A Simple Table

Classic HTML

```
<table>
<tr><td bgcolor="#FFCC00" align="left"><font face="arial"
size="2" color="red"><b>this is line 1</b></font></td></tr>
<tr><td bgcolor="#FFCC00" align="left"><font face="arial"
size="2" color="red"><b>this is line 2</b></font></td></tr>
<tr><td bgcolor="#FFCC00" align="left"><font face="arial"
size="2" color="red"><b>this is line 3</b></font></td></tr>
</table>
```

With CSS (assuming "subtext" is defined)

```
<table>
<tr><td class="subtext">this is line 1</td></tr>
<tr><td class="subtext">this is line 2</td></tr>
<tr><td class="subtext">this is line 3</td></tr>
</table>
```

Advantages of CSS

1. Makes web site maintenance easier
    - Fewer lines to change.
    - Fewer pages to upload.
2. Improves page load time for a site
    - Style sheet is downloaded once and cached.
3. Insures page consistency within a site
    - Every page that uses your style sheet is derived from an identical style
4. Also helps improve accessibility
    - People can define own style sheets to override default settings (poor vision, colorblind, etc).
    - Mobile devices can have customized sheets.
5. There are dozens of extra formatting options and possibilities available through stylesheet commands that are not possible through normal HTML/XHTML.

Implementation

CSS files are termed “cascading” stylesheets for two reasons: one stylesheet can cascade, or have influence over, multiple pages. Similarly, many CSS files can define a single page.

<u>There are 3 ways to implement CSS into your site:</u>

1. Use one CSS file for all your pages. (Best Way!)
2. Integrate CSS commands into the head of your documents.
3. Use the style attribute to put CSS code directly into an element.

CSS allows you to use all three of these methods together, inheriting and overriding values as you go.

One Stylesheet (to rule them all)

You write just one .css file and have all pages reference it.
Example: meyer.css

Syntax in CSS is DIFFERENT than in XTHML:
selector {property: value; property: value; property: value; }

Examples:

```
body {background: blue; color: white; }
/* Previously we set the body element this way: */
/* <body bgcolor="green" text="white"> */

h1 {font-family: Verdana, sans-serif;
    color: red;
    font-size: 20px; }

p, div, h2 {color: #00DDFF; width: 80%; }  /* modifies 3 tags */
```

Syntax Rules

1. The selector is usually the name of a tag, without its surrounding angle-brackets.
    - div, span, h1 etc.
2. The braces are {curly}, not [square] or (round).
3. After the property name there is a colon, and between each individual part there is a semicolon. Each of these pairs of properties and values is a declaration.
    - You can put each separate declaration on a different line to make it easier to read.

Attaching your StyleSheet

- In order for your XHTML pages to use a CSS, you'll need to show them where the css file is.
- Put this line of code into the head part of any documents you want to read this file:

  <link rel="stylesheet" type="text/css" href="mystyles.css">
- This could be a new tag to you — rel stands for the file's 'RELationship', and type shows that it's a text file acting as a CSS stylesheet.
- You can link multiple stylesheets to a page if you want, (having one file with all your fonts, another for margins and spacing etc.)

Individual Style blocks

If, a number of pages need some particular styling and you need to override the values you've defined in your main stylesheet, you can use the style blocks method.

To embed style, put this into your head:

```
<style type="text/css">
    p {font-weight: normal; color: gray; }
    h1 {color: black; }
</style>
```

The type attribute here allows browsers to treat this code as CSS. CSS code applied in this way is not technically a stylesheet , but is called an "inline style block."

Using the Style Attribute

If you need to modify one tag on its own you can embed style information into it using the style attribute:

<p style="color: blue; font-family: Arial; ">

The style formatting will stop as soon as you close the tag it's applied to, just like any other attribute, so it will be just this paragraph that will be affected. Also note that there aren't any curly braces used here, but the colon/semicolon rule still applies.

This method is useful for once-off formatting, and overriding previously defined properties, but you shouldn't use it very much. If you find yourself adding the same style to multiple tags, it might be worth your while promoting it to your main stylesheet, to save time and space.

Classes and IDs

If you have been using a stylesheet to reformat HTML tags you might wish you could just set up certain ways of formatting HTML elements and apply them to multiple tags.

You also might want to be able to define multiple types of a single tag, such as 2-3 standard paragraph types.

Using classes and ids (which are roughly the same thing), you can set up these custom options, which allow you to format single tags in many different ways. They're easy to set up, fast and flexible.

classes

Class selectors are created by typing a dot followed by the class name.

Example: You want to format lots of individual pieces of text as 12 point red Verdana (adding the font face, color and size XHTML for instance would suck). Put this line of CSS into your style:

.caution {font-family: Verdana; font-size: 12pt; color: red; }
.center {text-align:center}

Note the dot before the name you want to use for it. You can add classe to any element.

Examples:
<p class="caution">
<h1 class="center">.

NOTE: For classes that have multiple attributes, try to name the classes based on their function rather than their presentation.

id

ids are practically the same as classes, with one difference. Only one element can be given each id per page. The code is the same, but with hashes (#) in place of the dots.

#header {width: 90%; background: white; font-size: 20px; color: purple; }

<h1 id="header">stuff</h1>

NOTE: Both class and id names can contain characters a-z, A-Z, digits 0-9, underscores and hyphens, but they cannot start with a number or dash.

Limited Classes

It is possible to create "limited" classes that can only be applied to specific tags. This allows you to reuse tag names and control the application of classes without resorting to using ID's

```
/* ********************* A Custom Unordered List *********************/
ul.cust
{
    list-style-type:none;
    padding:0px;
    margin:0px;
}
li.cust
{
    background-image:url(arrow.gif);
    background-repeat:no-repeat;
    background-position:0px 5px;
    padding-left:14px;
}
```

&

 and are used to mark specific sections of code and are only different in that div tag acts as if a
 was declared before and after the start and end tag. These two tags are incredibly useful tools for identifying and selecting sections of text that you want to use a class or id on.

affected text

This would create your desired text, without a font tag in sight. The span tag does absolutely nothing on its own without the class attribute.

NOTE: As shown above, you can add multiple class descriptions at once.

General Model

```
/* ********* Styles specific to this site (may be separate sheet) ************/

body {background-color: teal}
h1 {color:black; font-size:20pt}

/* ***** Styles appropriate whenever (may be separate sheet) **************/

/* *** Color **/

/* Color class selectors */
/* Example: <h1 class="center black"*/
.black {color:black}
.aqua {color:aqua}
.blue {color:blue}
.white {color:white}
.yellow {color:yellow}

/* A cheat sheet is available at:
 * http://www.sci.brooklyn.cuny.edu/~meyer/css/CheatSheet.css
*/
```

CSS Documents Can Be "Validated"

You can check your .css documents to see if the are "valid" by going to the following link:
http://jigsaw.w3.org/css-validator/

If your css file violates any rules or is missing any required elements it will generate errors.

Website that are using "valid" css files can then add the following picture:

Getting More Help/Information

W3C Schools: (fantastic site, you can learn everything here, and they have "try it" pages that let you test sections of CSS code)
http://www.w3schools.com/css/

Other sites to look at:

- http://www.tizag.com/cssT/pclass.php
- http://www.echoecho.com/cssintroduction.htm
- http://www.davesite.com/webstation/css/chap01.shtml
- http://www.yourhtmlsource.com/stylesheets/introduction.html