CSS REFRESHER

Content versus Presentation

When moving toward XHTML best practices, we need to separate content from presentation.

- In the early days of the web (1990's), the browser dictated presentation through local user preferences. Default was black text on gray background with blue links and purple visited links.
- Later iterations in browsers add the <font> tag with attributes face (type), color, and size. Example:

  `<h1><font color= "#3366ff" face= "Helvetica">My Web Site</font></h1>`

- Problem: Assume we have a website consisting of 50 pages each with 10 <h1> tags that use the tag <font color="#663300" size="4" face="Arial, Helvetica, sans-serif">...</font>.
  - That's almost 700 extra characters per page – time consuming to type, extra download bandwidth, and lots of places to make mistakes
  - To change the color, need to search and replace on 500 instances.
  - Harder for search engines to index properly.
- Cascading Style Sheets (CSS) provide designers with control over most presentation aspects such as positioning, spacing, colors, margins, padding, and borders. CSS-1, the first specification of CSS, was ratified in 1998.

Check out the power of CSS
For an example of what you can do with style sheets, check out the CSS Zen Garden (www.csszengarden.com). Only the style sheet is changed between the different pages.

Advantages of CSS

- Search engine spiders parse for keywords in <h1>, <h2>, <h3>, etc. tags making it important to use them instead of images. CSS allows us to effectively use header tags.
- Easier site maintenance – to change the look of an entire web site, changes need only be made in a single place.
- Removing formatting makes for smaller HTML files.
- CSS allows for a greater number of attributes (typography, spacing, and layout) to modify.
- Improved placement and spacing control replacing unreliable frames and cumbersome tables.

Disadvantages of CSS

- Although vast majority of browsers in use support CSS, some interpret the code differently. ***Be sure to test!*** (Note: Virtually all browsers are CSS-1 compliant (you'd have to work pretty hard not to be) and most clients now have CSS-2 compliant browsers.)

Is Your Browser CSS-2 Compliant?
There are a number of ways to check if a browser supports CSS-2. Check one of these web sites.
- http://www.webstandards.org/action/acid2/
- http://jigsaw.w3.org/css-validator/

What is a Style Sheet?

- Like your XHTML files, a style sheet is simply a text file editable with text editors such as Notepad. They can also be edited using web development environments such as Dreamweaver or a CSS-specific editor.
- Within the style sheet is a list of rules defining how the designer wanted browser to display certain components of the web page.
- The general structure of a rule is:

  `selector {property: value; property: value; ... property: value}`

- Like XHTML, whitespace is ignored. Therefore, you can use tabs, carriage returns, etc. for organization.
- Examples:
  - `h1 {color: blue; font-size: 37px; font-family: arial}`
  - `p {color: blue; font-family: impact; font-size: 12pt}`
  - `code {text-indent: 1cm; background: #ccffcc; font-family: courier; color: green}`

Units

Many properties such as dimensions require units. The following is a list of CSS recognized units.

px:	pixels
pt:	points (1/72 of an inch)
pc:	picas (1/6 of an inch)
in:	inches
mm:	millimeters
cm:	centimeters
em:	ems = the height of the element's font
ex:	x-height, the height of the letter "x"
%:	percentage

Examples:

- 24px represents 24 pixels
- 2.5in represents 2.5 inches

Colors

- Colors can be identified by name, e.g., aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow
- Colors on the web can also be represented with RGB values. There are a few ways to do this:
  - o **#rrggbb** (e.g., #125211 or #0c0 = #00cc00)
  - o **rgb(ri,gi,bi)** where ri, gi, and bi are integers between 0 and 255 inclusive (e.g., rgb(54,196,20))
  - o **rgb(rp%,gp%,bp%)** where rp, gp, and bp are percentages of intensity between 0 (off) and 100 (full on) inclusive (e.g., rgb(34%,20%,86%))

Additional CSS References

- http://www.devguru.com
- http://www.w3schools.com/css/css_reference.asp

URL's

In some cases, a style may need to reference a URL, e.g., the source for a background image

- URLs are represented with ***url(site_address)***, where ***site_address*** is the URL. For example☹

```
BODY {background: url(http://www.myweb.com/image.gif) }
```

- If a reserved character such as ':', ')', '(', or a comma is part of the URL, it must be "escaped" with a backslash, e.g., the URL my:web must appear as my\:web. (Note that the ':' in "http://" does not need to be escaped.)
- Absolute and relative URLs are permitted. Relative URLs can be unpredictable since NN 4.x interpreted URL's relative to the HTML source while all others interpreted URLs relative to the style sheet source.

How to Associate a Style Sheet to Your Web Page Elements

There are four ways to connect a web page to the styles it will be using.

- In-Line Styles
- Embedded Styles
- External Style Sheets – Linking
- Importing Style Sheets

In-Line Styles

- In-line styles allow a designer to change the style for a single instance of a tag. For example, if you have three paragraphs and you only want to change the style on the second one, use an in-line style.
- In-line style uses the attribute ***style*** within the tag to define the rules, i.e., style= "property: value"
- Example: `<p style="color: red; font-size: 24pt">...</p>`
- This will change the style only on the paragraph with the in-line style.

Embedded Styles

- Embedded style sheets are inserted into the document's head and affect ***all*** elements within a page.
- Style definitions/rules are placed between the tags `<style type="text/css">...</style>`
- Browsers that do not support style sheets should ignore this text.
- The attribute ***type*** defines the format of the cascading style sheets to be used. For our purposes, use "text/css".
- To force browsers without CSS support to ignore the rules, place rules inside of comment tags <!-- and -->.
- All elements within the web page will follow these rules unless overridden by an in-line style.

Embedded Styles (continued)

- Example:

```
<style type="text/css">
<!--
   h1 {color: blue; font-size: 37px; font-family: arial}
   p {color: blue; font-family: impact}
-->
</style>
```

External Style Sheets – Linking

- To maintain a common look and feel across all of the pages on the site you are designing, it is best to create a single set of style rules. This way all of the pages on the site can use the same file.
- To do this, you will create a separate text file that contains the rules.
- This file uses the same syntax as the embedded style sheet rules.
- Although the file extension isn't critical, the common file extension for style sheets is .css.
- For external style sheets, the ***<link>*** tag replaces the ***<style>...</style>*** tags in the head of the html file.
- Its format is **`<link rel="stylesheet" href="exer3.css" type="text/css">`**
- ***rel*** defines how the external file will be used in the HTML document.
- ***href*** gives the browser the URL (relative or absolute) of the style sheet.
- ***type*** is just like the type for the embedded style sheet.

Importing Style Sheets

- There is a second (less popular and not recommended) method to retrieve style rules from an external style sheet file. It is called "importing".
- The link tag is more universally supported by browsers.
- Importing works like a combination of embedded style sheets and linking.
- It replaces the rules of an embedded style sheet with the text **`@import url(mystyle.css)`**.
- Example:

```
<style type="text/css">
<!--
    @import url(mystyle.css);
-->
</style>
```

- At run time, the text in mystyle.css would be inserted at point of the @import syntax.
- The beauty of this method is that you may also add additional style rules after the import command.

Understanding the "Cascade" in CSS

It is possible to have more than one style rule applying to an XHTML element. There are two reasons for this:

- Designers may use several style sheets and in-line styles to define the look of a page.
- Sometimes nested tags need to take on the properties defined by the tags in which they are nested.
  - o For example, an <em> tag that defines the weight of the text should inherit nested the traits defined by the <p> tag within which it is nested. An <li> tag should in some cases inherit the traits of the <ul> tag.
  - o This suggests a hierarchy of tags, i.e., which tags inherit traits from which other tags.
  - o This causes a problem when it comes to tags that try to define the same trait

The following rules specify how a browser should resolve conflicts. (A more in-depth discussion can be found at http://www.w3.org/TR/CSS1#the-cascade)

- More specific rules override the more general ones. For example, <em> is morespecific than <p>.
- The most recently defined rule overrides past rules. This means that embedded rules coming after an imported style sheet override those of the imported style sheet just as in-line rules override all those declared in the head.
- A property using the ***important*** declaration overrides normal declarations. Example:

```
p { font-size: 14pt ! important }
```

A ***selector*** identifies to the browser what the rules/properties apply to. So far, we have only discussed tags as a selector. The designer, however, is granted greater control through different types of selectors:

- Type selectors (the tag is the selector)
- Contextual Selectors
- Class Attribute Selectors
- ID Attribute Selectors
- Pseudo-Selectors

Contextual Selectors

Contextual selectors apply for a specific nesting of tags, e.g., we can define a style to be used only when the <em> tag is nested inside a <p> tag. For example, the following rule applies to the <em> tag only when it occurs within a <p> tag:

```
p em {font-weight: bold; color: blue}
```

Several contextual selectors can be grouped together using commas as shown below:

```
p em, h1 em {font-weight: bold; color: blue}
```

Class Attribute Selectors

By adding a class attribute to specific HTML element tags, you can identify elements and modify them with a single style rule. For example, to identify certain paragraphs as important, create a class called “important”:

```
<p class=“important”>...</p>
```

A style used only for important can then be created:

```
p.important {color: red}
```

To make all elements of a class obey the same rules, eliminate the tag name in the selector leaving the period:

```
.important {color: red}
```

ID Attribute Selectors

For a specific element rather than a group of elements, the ID selector may be used. Identify the element in the XHTML page using the attributes id and name.

```
<p id=“bob" name=“bob">...</p>
```

The style rule becomes:

```
p#bob {margin: 24px}
```

Once again, leaving out the tag name (but not the #) applies rule to all elements with id.

Pseudo-Selectors

CSS also allows designers to define styles for specific components or states of elements. The following example would make the first character 3 times the size of the other characters and red.

```
P:first-letter {font-size: 300%; color: red}
```

Several types of selectors may be combined. For example, the rule below is perfectly legal.

```
p.custom:first-letter {font-size: 300%; color: red}
```

Some Pseudo-Selectors
For paragraphs:
- first-letter
- first-line

For anchors/links:
- hover
- visited
- active
- link

New Tags with CSS

The following tags were added with the advent of CSS to allow the designer better control of presentation:

- <div>…</div> – The begin tag <div> and the end tag </div> can be used to set the style for a block of HTML code. This block usually covers a larger area of code possibly with multiple tags nested within it.
- <span>…</span> – The begin tag <span> and the end tag </span> can be used to set the style of an HTML element embedded within other HTML tags. It is usually used for very short sections of code such as a phrase within a paragraph.

The next two tags are also new with style sheets and can be used to represent edited text within an HTML document. It sometimes is nice to use these tags if a number of people are working on a single document and wish to see the items that were changed from one revision to the next.

- <del>…</del> – Represents deleted text. As a style, the property **`text-decoration: line-through`** works very well.
- <ins>…</ins> – Represents inserted text.

Minimizing Problems with Style Sheets

In order to minimize the chances of a problem with a style sheet, the designer should use the following strategies:

- Follow strict XHTML formatting guidelines
- Use CSS properties supported by the majority of browsers
- Use a client-side script such as JavaScript to detect browser so as to load appropriate style sheet