CISC 3120
Design & Implementation of Software Applications I

Lecture #1 – XHTML & CSS

Professor: M. Meyer
Course Page:
http://www.sci.brooklyn.cuny.edu/~meyer/CISC3120
Contents

- Markup Languages
- XHTML
  - Overview
  - Rules, Validation
  - Major Tags
- CSS
  - Why
  - How
- Fundamentals of Web Design
(X)HTML Motivation (WWW)

- It is possible to send pictures over the Internet, but pictures are large (slow to transfer), fixed (difficult to rearrange) and difficult to resize (interpolation, aliasing).
- It's very cheap and easy to send plain text over the Internet (ASCI, Unicode, UTF-8, UTF-16 and UTF-32). But plain text is difficult to read.
- **Markup Languages**: Languages that allow us to add information to text, in a manner that is distinguishable from the text.
- Markup languages can be used to enhance the comprehension/understandability of the text.
Types of Markup

1. **Presentational markup**: Used by traditional word-processing systems, to create a WYSIWYG effect.
   - Examples: add line break, bold word, change font style or color.

2. **Procedural markup**: Provides instructions for programs that are to process the text.
   - Examples: add an image, applet or link to a document.

3. **Semantic markup**: Used to label parts of document and attach additional meaning to those sections.
   - Examples: define the title of a document or declaring that a section of text is an address.
Markup Languages - Key Terminology

- **Tag**: A markup construct that begins with "<" and ends with ">". Tags come in three flavors: start-tags, for example `<p>`, end-tags, for example `</p>`, and empty-element tags, for example `<br/>`.

- **Element**: A component that begins with a start-tag and ends with a matching end-tag, or consists only of an empty-element tag. The characters between the start- and end-tags, if any, are the element's content, and may contain include other elements, which are called child elements. An example of an element is `<p>Hello, world.</p>`. Another is `<br/>`

- **Attribute**: A construct consisting of a name/value pair that exists within a start-tag or empty-element tag. In the example (below) the element img has two attributes, src and alt: `<img src="button.jpg" alt='button'/>`. 

**HTML vs. XHTML**

- XHTML is almost identical to HTML 4.01
- XHTML is a stricter and cleaner version of HTML
  - Many pages on the internet contain "bad" HTML.
  - Browsers are still expected to interpret this "bad" HTML correctly (mobile devices).
- XHTML is HTML defined as an XML application
  - This allows you to create and define your own tags.
  - This allows webpages to be parsed (searched) with XML utilities like XPath and Xquery.
- There is already work on a “XHTML 5” and HTML5 webpages can follow the stricter ruleset of XHTML with absolutely no ill effect.
The Most Important Differences ARE:
(all examples are WRONG)

1. **XHTML elements must be properly nested**
   - EX: `<b><i>This text is bold and italic</i></b>`

2. **XHTML elements must always be closed**
   - EX: `<p>This is a paragraph</p>`

3. **XHTML elements must be in lowercase**
   - EX: `<P>This is a paragraph</P>`

4. **XHTML documents must have one root element**
   - EX:
     
     ```html
     <html>
     <head> ... </head>
     </html>
     
     <html> <html>
     <body> ... </body>
     </html>
     ```
XHTML Rules (2)

Some More XHTML Syntax Rules (all examples are CORRECT)

5. Attribute names must be in lower case
   • Ex: `<table width="100%">`

6. Attribute values must be quoted
   • Ex: `<table width="100%">`

7. Attribute minimization is forbidden
   • Ex: `<frame nosize="noresize" />`

8. The id attribute replaces the name attribute

9. XHTML has predefined mandatory elements
Minimum Components of a "Transitional" XML document

```xml
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
   <title>simple document</title>
</head>

<body>
   … some content …
</body>

</html>
```
XML Documents Can Be "Validated"

- You can check your .html documents to see if they are "valid" by going to the following link: http://validator.w3.org/
- If your xhtml file violates any rules or is missing any required elements it will generate errors. The "minimal transitional document" from the previous slide is free from errors (although it generates "warnings").
- XHTML pages (and websites) that are "valid" can then add the following picture:
Structural Tags

- **Paragraphs:**
  
  ```html
  <p>This is the first paragraph.</p>
  <p>This is the second paragraph.</p>
  ```

- **Line breaks:**
  
  ```html
  This is the first line.<br />
  This is the second line.
  ```

- **Division**
  
  ```html
  <div>A specific section of text</div>
  ```

- **Note:** The div and span elements are very often used with CSS to layout a web page. Browsers usually place a line break before and after the div element.
Character Tags

- **Emphasis**: This is `<em>interesting</em>`!
- **Italics**: This is `<i>interesting</i>`!
- **Bold**: This is a `<b>amazing</b>`!
- **Preformatted Text**:
  
  `<pre>
  Text in a pre element is displayed in a fixed-width font, and it preserves both spaces and line breaks
  </pre>`
Link Tags

- A link is a tag that directs your browser to another page if the user clicks on the link. The content of the link tag is:
  1. The URL where you want the browser to go if the user clicks on the link, and
  2. The text that you want the user to see (i.e., the text that the user will click on to activate the link)

- **Examples:**
  This a link to
  
List Tags

Un-ordered list:
<ul>
  <li>the first list item</li>
  <li>the second list item</li>
  <li>the third list item</li>
</ul>

Ordered list:
<ol>
  <li>the first list item</li>
  <li>the second list item</li>
  <li>the third list item</li>
</ol>

Note: You can nest lists!
Table Tags

- Begin a table with `<table>`
- End a table with `</table>`
- Begin each row with `<tr>` and end each row with `</tr>`
- Within each row, begin each column with `<td>` and end each column with `</td>`
- Options:
  - borders
  - `cellpadding` (padding within a cell)
  - `cellspacing` (spacing between cells)
  - width and height (in pixels)
- Tricks:
  - empty cells — use `&nbsp;`
  - spanning multiple rows or columns
- Coloring cells:
  - `<td bgcolor="red">ASDF</td>`
Escape Characters (HTML Entities)

- In HTML, there are certain special characters that must be escaped in HTML documents, otherwise they will be taken as special characters. Some common examples are:

  < - Unescaped this indicates the start of a tag
  & - The escape character itself
  Blank - The HTML rendering process ignores repeated white-space

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Escape Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
</tr>
<tr>
<td>blank</td>
<td> </td>
</tr>
</tbody>
</table>

- Escape sequences can also be used to represent all 110,182+ Unicode Characters.
CSS Motivation

- Imagine you've just designed a two hundred page web site for a client, who at the last minute decides that the background color of all the pages needs to be changed.
- Stylesheets act as partners to HTML/XHTML documents; taking care of all layout, fonts, colors and the overall look (the aesthetics) 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.
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 style sheet is has 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 style possibilities only available using CSS.
A Smaller Table

- Classic HTML

```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)

```html
<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>
```
Implementation

CSS files are termed “cascading” stylesheets for two reasons:

1. One stylesheet have influence over multiple pages.
2. Many CSS files/statements can define a single page.

There are 3 ways to implement CSS in your site:

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.

Note: With CSS you can use all three of these methods together, overriding values as you go.
Syntax

selector {  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

- The selector is usually the name of a tag, without its surrounding angle-brackets.
  - div, span, h1 etc.
- The braces are {curly}, not [square] or (round).
- After the property name there is a colon, and between each individual part there is a semicolon.
- Each of pair 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:

```html
<link rel="stylesheet" type="text/css" href="mystyles.css" />
```

- You can link multiple stylesheets to a page if you want (have one file for 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 the <head> section:

  ```html
  <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-size: 20px;">`

- 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.
  - Note: Curly braces aren’t 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 to put it in 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 very similar), 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. The dot implies ownership, if nothing is in front of the dot, anyone can use the class.
- Example: If you put this line of CSS into your style:

  ```css
  .caution { font-size: 24pt; color: red; }
  ```

- You can add the caution class to any element.
  ```html
  <p class="caution">
  ```

- **NOTE**: Try to name the classes based on their function rather than their presentation.
id

- An id is similar to a class, with one difference: Only one element can use each id (per page). The code is the same, but with hashes (#) in place of the dots.

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

- In the webpage:

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

NOTE: class and id names can contain characters a-z, A-Z, digits 0-9, underscores and hyphens, but 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;
}
<span> & <div>

- <span> and <div> are used to mark specific sections of code and are only different in that div tag acts as if a <br/> was declared before and after the start and end tag.

- These two tags are useful for identifying and selecting sections of text that you want to use a class or id on.

<span class="caution bold">Fire</span>

- 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}
CSS Documents Can Be "Validated"

- You can check your .css documents to see if they 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**: Nice site, and they have "try it" pages that let you test sections of (X)HTML and CSS code)

http://www.w3schools.com/

Other sites to look at:

- **HTML Reference**
- **CSS (Cascading Style Sheets) Information**
Website Design

- You don't have to play, but if you are going to play, play to win.
- Likewise, if you are going to make a webpage or website, you should strive to make it a great one.
- There are many, many well defined and understood rules regarding website design, website creation and website maintenance. You should strive to learn some of them.
Practical aspects of a good website.

1. A good website is compelling.
2. A good website has clear goals.
   - Make money
   - Provide a service
   - Organize a group or cause.
3. A good website is easy to navigate.
   - Principles of human computer interaction.
4. A good website is visually attractive.
   - Many, many solid rules to follow for those not naturally of an artistic bent.
Website Organization

1. **NEVER EVER** edit a live site: Edit a private copy and upload/copy it when you are finished and have tested it.

2. **Use multiple folders:** Have those folders in a consistent structure and naming scheme.
   - For very large sites it might even be worthwhile to keep individual pages in their own folders.

3. **Link internally:** As much as possible link to files and functions hosted on the same machine.
Note: Relative vs. Absolute references

1. Putting files in separate folders will require the use of relative references in your links and sources.

2. **Absolute references** work anywhere in the world (Fully Qualified Domain Name).
   - www.google.com

3. **Relative references** use shortcuts to specify a location based on where you are starting from:
   - `./images/image1.png` (start in current folder, then go to images folder)
   - `../images/image1.png` (go up one folder, then go to images folder)

   - Example: Image is in images folder on your site:
     `<img src="./images/image1.png" alt="Picture1"/>`
Effective Design Tips

1. Create a visual theme that is consistent.
   • Complimentary colors.

2. Align elements on the page.
   • Don't center everything!
   • Don't be afraid of whitespace.

3. Use contrasting color and text weight.

4. Group elements together so that the reader can more easily grasp the information you are presenting.

5. Make limited and intelligent use of images including images that display text.
You got this!