Memo	**IMED 2351**
Subject	HTML/XHTML/CSS Review

In week 1 we do a review of the most commonly used XHTML tags for this course. Familiarization with the XHTML code will ensure that working with PHP & server side development will be easier. Please review the tags below and add any additional information you may need to master each tag.

File Technical

File naming conventions: Do not use spaces in the names of the .htm, .html or .php files. Although it is possible to use spaces in the file names, this will cause you a lot of grief as your project matures. It is fine to use "_" underscores but I do not recommend this.

Make the names of your pages humanly readable if possible. There is an advantage to this is in many ways. For example: if you're selling a product of books, name your book product page "books.htm".

Folder structure: Place all completed projects and assignments in the following folders:

http://studentXX.imed2309.net/midterm/
http://studentXX.imed2309.net/final/
http://studentXX.imed2309.net/assignment01/
http://studentXX.imed2309.net/assignment02/
(… and so forth …)
(where student_xx is the name of the folder assigned to you in the beginning of class)

Create an index (or relevant default document) page in the root folder of your site. It will need to link to your assignments and projects. If necessary you may also link to any exercises you may work on that you would like for me to review.

The recommended directory structure for project(s) should be as follows:
/ (*root*)
/midterm/images
/midterm/images/products/
/midterm/scripts
/midterm/html
/midterm/js

You may use a separate location for the testing of your files, but the finished version of your project(s) should be at the locations specified above.

Place the **images** used for the design of your site in the **/images** folder. Place the **images** for your products in the **/images/products/** folder. You may create additional sub-folders if needed.

Place any JavaScript or CSS files in the **/scripts** or **/css** or **/js** folders. Be sure to point to everything correctly.

If following the development standards of this course you may place your **html** files in the **/html** folder and leave your **.php** files in the **root** folder. This is not required but will allow you to separate any designed content files from the code.

- Feel free to use your own artistic design for all pages of your projects. The idea is to impress the client/customers with a good looking web site that is also functional. It may not have all the bells and whistles, but it will need to work.

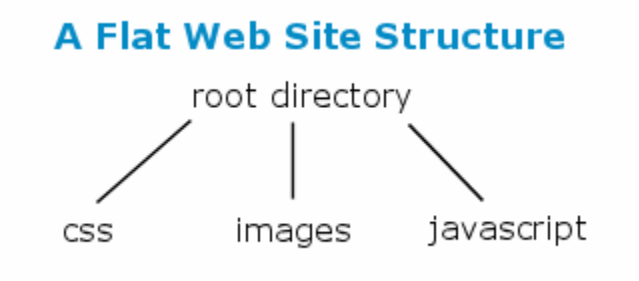
- Set a limit to how much you are going to tweak your pages. Lots of time can be wasted by trying to get something to be perfect.

NOTE: Each folder should/must have at most one **default** document (e.g. index.htm, index.html, index.php, default.htm, default.html). This will protect any content code and images you may have in your documents. It also presents your site in a professional manner and allows proper navigation for web surfers.

Image and File Paths

Keep the structure of your web site as flat as you can.

A flat web site structure uses only one level of subdirectories, which are each named according to the type of web objects they hold.



For small web sites (less than 100 pages), all HTML files should reside in the root directory.

The root directory of a site may vary, it can be named “publich_html” or “wwwroot” or “www”

For a flat structure, just use one level of subdirectories. Any directory below another directory is a subdirectory.

Keep similar web objects (htm, js, css, jpg, gif, png, etc.) in each subdirectory.

External CSS files should be placed in the “css” directory

Images in a directory named “images”

If you use JavaScript files, place them in the “javascript” or “js” directory

You can name subdirectories anything you want, but the name should describe the objects found inside.

On some web servers the capitalization of files names and directories is very specific. This means that the caps in the document must match that of the file or directory you are attempting to access.

Types of Paths

Paths indicate the location of a file/document you want to link to or access on a web server. Paths are commonly used for linking to images, other web pages on the same site, web pages on different web sites, videos (externally and embedded), dynamic objects like flash or Java and dynamic scripts like ASP.Net or PHP files.

Absolute Path

An absolute path includes the full URL to the images/object you want to display or access.

Example:
**

The full URL path to the image above includes:

- the http protocol (http://)
- the domain name (www.imed2309.net)
- the image subdirectory (/images/)
- the file name (missingimage-ie.gif)

An absolute path doesn't care where you are in a web site or which page you are currently viewing.

Relative Path

A relative path is always "relative" to a web page in your site structure hierarchy that is requesting the image. There are several variations that can be used for relative paths. In a simple structure, all HTML pages reside in the root directory. A simple relative path to the same images used in the example above would be:

Example:
**

"images" is a subdirectory of the location where the HTML file is calling the image, so you include the subdirectory name at the beginning of the path.

It can get complicated when an HTML page resides in a subdirectory, however. If we add a subdirectory called "articles" and place an HTML page in that directory, the simple relative path to the images directory will no longer work because the images are no longer in that subdirectory where the HTML file is located.

When creating a path we have to move up one level to the root directory in order to access the "images" directory.

To move up a directory level, just precede the path with a double-dot and a slash, (../). This tells the browser to request a path that starts one level **above** the location of the current HTML page.

The image tag path would look like this:

**

If the HTML page is deeper in the web site (two subdirectories) down from the root directory, as in /articles/october/books.html, then the path to the images subdirectory requires that you first move up two directory levels (back to the root directory) to find the path to the "images" subdirectory.

Example:
books.html lives here
/articles/october/books.html

books.jpg lives here
/images/books.jpg

correct link would be:
**

When working with web sites with multiple subdirectory levels, always think in terms of moving up to the root directory and then back down to the path to the object you want to use or display.

Root Relative

A root relative path always starts out at the root directory, so you do not need to move up one or more levels before you start the path to an image or object. When a path starts with a slash, it simply means "start out at the root directory".

Example:
**

What if the path is correct, but the image still doesn't display?

The issue may not be related to the path.

- File names for all images on Unix and Linux servers are case sensitive. That means that an image named MyDog.jpg is not the same as mydog.jpg or MyDog.JPG. Make sure that the file name is spelled correctly and all characters are of the proper upper or lower case.

- You may be using an invalid file name. The only characters that are technically valid for file name on Unix and Linux servers *are upper and lower case alphabetic characters, numbers, the underscore, a hyphen and a dot*. Spaces, pound signs, slashes, parenthesis, questions marks or any other characters are *not valid* in either file names or directory names. Some characters may be valid in URLs, but not in file or directory names. People pick up bad habits with file names primarily because Microsoft Windows and Microsoft servers allow you do things that are not valid with other systems. Microsoft has no influence on the way that the Internet is structured or how it runs. Make sure that your files and paths use valid characters.

- Paths are different on web servers than they are on your PC. If you are trying to test a web site that you are developing on your PC, it probably will not work correctly. Absolute paths are different on a PC and the root directory is not the same as the root directory on a web server. If you want to test a simple static HTML web site on a PC that is not configured as a server, use relative paths.

- Your HTML editor may be creating incorrect paths to the images subdirectory. Some editors automatically create paths that work on your PC, but will not work on a web server. You will need to look at the code to identify this problem. If you see a path that is not relative or absolute as demonstrated above, the editor may be using absolute paths on your PC, which will not work on a web server. An example would be any image path that starts with a hard drive designation on your PC, such as:

  c:\\MyWebProjects\images\missingimage-ie.gif

  If you see a path in your code that looks like this, you will have to determine how to configure your editor to use the paths that you choose. The clues are the backslashes and the drive designation (c:).

XHTML Prerequisites

The XHTML areas you need to understand are:

* **Basic Text**
  - Headers: <h1> etc
  - Blocks: <p> and <blockquote>
    + Attributes: align
  - Character formatting: <i>
* **Images**
  - Image Links <img>
  - Attributes: src, height, width, alt
* **Lists**
  - Ordered: <ol>
  - Unordered: <ul>
  - Attributes: type, start
* **Tables**
  - Structure: <table> <tr> <td> <colspan> <rowspan>
  - Attributes: bgcolor, align, valign, cellspacing, cellpadding
* **Links**
  - To pages, sites, email <a>
  - Attributes: href, target
* **Forms**
  - setup using basic form inputs:
    + text, textarea
    + checkboxes, radio buttons
    + select menus
  - form attributes: action and method

Document Type Definition

A valid XHTML document must begin with a proper Document Type Declaration or DTD. Simply choose the one that matches your document type and cut and paste! Unless you are using frames in your page, odds are you want the **transitional** type. If you are a stickler for detail, or anticipate a significant number of users browsing with alternative devices (handheld computers, PDAs, iPhones, PSPs, etc), then use the **strict** type.

You may have seen DTDs in use with HTML page, but they were not strictly necessary (though it is good to use them there as well!).

XHTML specifies three possible document types:

The **Strict** DTD is used for creating really clean markup and all formatting must be implemented with Cascading Style Sheets (CSS). Font, Bold, Italic and other presentation tags will cause your document to be invalid:

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

The **Transitional** DTD is the most commonly used to accommodate browsers that don't support (or don't support fully and properly) the CSS standard. This will allow many of the presentation tags as long as the syntax is correct. Use this for now, if you need to, but ultimately you want to be using strict:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

The **Frameset** DTD is used if you wish to use HTML frames to divide your document:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"

  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Character Sets

In order for a web page to be displayed correctly, there must be some indication to the browser of whether the characters used are ascii, western european, cyryllic, kanji, etc. A well-behaved server will send these codes automatically with your document. However, if your server is not doing so, or you are testing files by uploading them to a validator, then you can either ignore errors related to the character set or manually insert a character set meta tag as the *first* element in the HEAD of your document:

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
```

Declaring the Namespace

Because an XHTML document can contain elements of different kinds and different character sets, you need to declare which `namespace` the html section of your document comes from. You do this by adding the `namespace` attribute to the `html` tag. For our purposes we will be using the standard XHTML namespace:

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

Lower Case Tags and Attributes

Unlike HTML, all tags and attributes (not attribute values) must be in lower case. Many of us have grown fond of capitalizing HTML to make it stand out from the rest of the text:

```
<P>This rocks!</P>

<table WIDTH="500">
```

This is not valid XHTML. Instead, you **must** use all lower case:

```
<p>This rocks!</p>

<table width="500">
```

All Attribute Values Must Use Quotation Marks

Every attribute value in a tag must be quoted. So this table tag would be invalid:

```
<table width=500>
```

The value *500* must be enclosed in quotes:

```
<table width="500">
```

Single quotes or double quotes are fine… this will become an important consideration a little later in the course!

You Have to Use Closing Tags

It is common in HTML to leave off "optional" closing tags like `</p>` or `</li>`:

```
<p>One paragraph

<p>And another
```

This must be changed to utilize closing tags:

```
<p>One paragraph</p>

<p>And another</p>
```

You Must Close "Empty" Elements

This is perhaps the strangest change for most hand coders. Even "standalone" html tags like `<img>` must now be closed by putting a trailing / before the closing bracket:

```
<img src="flower.jpg" />
```

This applies to all tags, including things like the link tag used to link a CSS style sheet to your document:

```
<link href="mystyles.css" />
```

Elements Must be Properly Nested

In HTML, the following would be valid, even though the `<em>` tag should have been closed before the `<strong>` tag:

```
<strong><em>This is some strong, emphasized text.</strong></em>
```

In XHTML, you must nest properly:

```
<strong><em>This is some strong, emphasized text.</em></strong>
```

You Must Use "Entities" for Special Characters

You may have gotten used to "cheating" and sticking things like ampersands (&) in your text when using HTML:

```
<p>You really should go see "Jack & Jill"</p>
```

To create valid XHTML, you must use the "entities" that represent these characters:

```
<p>You really should go see "Jack &amp;amp; Jill"</p>
```

Attribute Pairs Can’t Be Minimized

Many users of XHTML don’t realize they are minimizing attribute pairs in the first place. However, all attributes must have a value. So, to create a horizontal rule without shading in HTML we would write:

```
<hr noshade>
```

In XHTML we have to expand the attribute (and add the closing / character):

```
<hr noshade="noshade" />
```

IDs, not Names*

In XHTML, the ID attribute replaces the NAME attribute. For example, instead of:

```
<img src="foo.jpg" name="uaf" />
```

You would use:

```
<img src="foo.jpg" id="uaf" />
```

Attribute cannot be minimized

Form element attributes should not be minimized. They should be correctly typed out.

```
<input checked=”checked” />
<input readonly=”readonly” />
<input disabled=”disabled” />
<option selected=”selected” />
```

Scripts

Javascript code must either be wrapped in a CDATA section (starting with *<![CDATA[* and ending with]]>):

```
<script language="JavaScript type="text/javascript">
<![CDATA[
document.write("<b>I am valid!</b>");
]]>
</script>
```

Or you can put your script in a second file and link to it (sometimes a better solution anyway):

```
<script language="JavaScript" src="myscript.js"></script>
```

The Minimal XHTML Document

Following is a minimal XHTML document:

```
<!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>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Minimal document</title>
</head>
<body>
```

```
</body>

</html>
```

Validation

Although current browsers will attempt to interpret even poorly written XHTML, later browsers will likely reject invalid markup out of hand. Even if you are *not* using XHTML, writing valid code will prevent a lot of possible problems in older browsers.

XHTML Tags

This section contains information about all of the valid tags belonging to the latest version of HTML. (Reference: http://www.htmldog.com/reference/htmltags/)

Structure		
<html>		
	The root element that specifies that the content of the document is **HTML**. The opening tag immediately follows the **DOCTYPE** declaration and the closing tag is the last thing in the document. The html element must contain the head and the body elements.	

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
        <head>
                [stuff]
        </head>
        <body>
                [stuff]
        </body>
</html>
```

<head>		
	The header of an HTML document where information about the document is placed. You must use the title element within the head element and meta, style, script, base and link can also be used. You must use this element and it should be used just once. It must start immediately after the opening html tag and end directly before the opening body tag.	

```
<html>
        <head>
                <title>Shiny Gongs</title>
                <link rel="Shortcut Icon" type="image/ico" href="/favicon.ico"
/>
                <script src="/script/somescript.js"
type="text/javascript"></script>
        </head>
        <body>
                [stuff]
        </body>
</html>
```

<body>		
	The main body of an HTML document where all of the content is placed. You must use this element and it should be used just once. It must start immediately after the closing head tag and end directly before the closing html tag.	

```
<html>
        <head>
        </head>
        <body>
                [a whole load of stuff]
        </body>
</html>
```

	Division. Defines a block of HTML. Commonly used to apply CSS to a chunk of a page.	

```
<body>
        <div id="navigation">
                [stuff]
        </div>
        <div id="content">
                [stuff]
        </div>
</body>
```

	Used to group in-line HTML. span applies no meaning and is commonly used solely to apply CSS.	

```
<p>You could apply styles to <span class="whatever">this text</span> or <span class="whatever">tis thext</span> using the span tag.</p>
```

Meta Information

<!DOCTYPE>		
	Document type declaration. This is used to let the browser know what version of HTML you are using. If you don't use it, or if you get it wrong, the browser will assume you don't know what you're doing and switch to 'quirks mode', which will not render things as they should be. Apparently it's more 'forgiving' but it actually seems to be quite random and confusing. The case must be like that in the example below (with upper case 'DOCTYPE'). It does not close like other tags. Note that the DOCTYPE tag does not need to be closed with a slash and is always in ALL CAPS.	

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

	strict <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">	All markup is XHTML-compliant
	transitional <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">	The markup is a mix of XHTML and depreciated HTML. So your old HTML code an co-exist happily with XHTML.
	frameset <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">	Same as transitional, but the use of frames is permitted. Frames are depreciated under XHTML.
<title>		
	Defines the title of a page. You must have a title element to produce a valid document and it must be placed within the head element.	

```
<html>
        <head>
                <title>Shiny Gongs</title>
        </head>
        <body>
                [stuff]
        </body>
</html>
```

<link>		
	Defines a link to an external resource. It is most commonly used to link a CSS file to an HTML document. link must appear within the head element.	

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

<meta>		
	Meta information. Used to provide information about the HTML page. It must be placed within the head element.	

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="keywords" content="darwin, evolution, natural selection, species, beagle, 1859" />
<meta scheme="ISBN"  name="identifier" content="0-14-043205-1" />
```

<style>		
	Used to define CSS at a page-level (as opposed to an external CSS file). A style element must appear inside the head element.	

```
<style type="text/css">
        body {
                color: red;
                background-color: yellow;
                font-size: 80%;
        }
        p {
                line-height: 1.5em;
        }
</style>
```

Text		
	Paragraph.	
** **Big long paragraph thing.** **		
<h1>…<h6>		
	Heading 1 to Heading 6 (1 being the highest level, 6 being the lowest). Headings should be used in order and h1 used just once.	
** **The main heading** ** ** **A subheading** ** ra ra ra ra ra ** **Another subheading** **		
	Strong emphasis.	
That's ****strong emphasis**** ladies and gentlemen.		
	Emphasis.	
That's ****emphasis**** ladies and gentlemen.		
	A line break.	
some text ra ra** ** and some more ra ra		

Links

<a>		
	Anchor. Primarily used as a hypertext link. The link can be to another page, a part of a page or any other location on the web. Note: An anchor (a point in a page where a link can jump to) does not need to be defined with the a tag. Applying the id attribute to any tag will achieve this.	

```
<p><a href="http://www.htmldog.com">Link to a URI</a></p>
<p><a href="#content">Link to a page anchor</a></p>
```

<base>		
	Defines the base location for links on a page. Relative links within a document (such as <a href="someplace.html"... or <img src="someimage.jpg"...) will become relative to the URI specified in the base tag. The base tag must go inside the head element.	

```
<head>
        <title>Peppers</title>
        <base href="http://www.somedomain.com/directory/" />
</head>
<body>
        <p><a href="someplace.html">This</a> will actually link to
http://www.somedomain.com/directory/someplace.html.</p>
        <div><img src="someimage.jpg" alt="Some image" /></div>
        <p> The location of the above image will be actually be
http://www.somedomain.com/directory/someimage.jpg.</p>
</body>
```

Images		
<img>		
	Image. Note: When an image is used as a link, many browsers will show a border around the image. To get rid of this you should use CSS (border: 0).	
****		

Lists

	Unordered list. Defines a list that has no logical sequence. Used in conjunction with li to define the list items.	

```
<ul>
        <li>This</li>
        <li>That</li>
        <li>The other</li>
</ul>
```

	Ordered list. Used to define a list that has a logical sequence. Used in conjunction with li to define the list items.	

```
<ol>
        <li>First item</li>
        <li>Second item</li>
        <li>Third item</li>
</ol>
```

	List item. Used in conjunction with ul or ol to make an unordered list or ordered list respectively.	

```
<ul>
        <li>This</li>
        <li>That</li>
        <li>The other</li>
</ul>
```


Tables

<table>		
	Defines a table used for tabular data.	

```
<table>
        <tr>
                <th>Question</th>
                <th>Answer</th>
                <th>Correct?</th>
        </tr>
        <tr>
                <td>What is the capital of Burundi?</td>
                <td>Bujumburra</td>
                <td>Yes</td>
        </tr>
        <tr>
                <td>What is the capital of France?</td>
                <td>F</td>
                <td>Erm... sort of</td>
        </tr>
</table>
```

<tr>		
	Table row. tr elements must appear within a table element.	

```
<table>
        <tr>
                <th>Question</th>
                <th>Answer</th>
                <th>Correct?</th>
        </tr>
        <tr>
                <td>What is the capital of Burundi?</td>
                <td>Bujumburra</td>
                <td>Yes</td>
        </tr>
        <tr>
                <td>What is the capital of France?</td>
                <td>F</td>
                <td>Erm... sort of</td>
        </tr>
</table>
```


<td>		
	Table data cell. If the cell contains a header rather than data, th should be used instead. td must be used inside a tr element.	

```
<table>
        <tr>
                <th>Question</th>
                <th>Answer</th>
                <th>Correct?</th>
        </tr>
        <tr>
                <td>What is the capital of Burundi?</td>
                <td>Bujumburra</td>
                <td>Yes</td>
        </tr>
        <tr>
                <td>What is the capital of France?</td>
                <td>F</td>
                <td>Erm... sort of</td>
        </tr>
</table>
```

Forms

<form>

Defines a form allowing user-inputted data to be sent somewhere. It is used with elements such as input, select and textarea.

```
<form action="/somedirectory/somformprocessingscript.php" method="post">
        <div>House number: <input type="text" name="housenumber" /></div>
        <div>Street: <input type="text" name="street" /></div>
        <div><input type="submit" /></div>
</form>
```

<input>

A form element that can be represented as a text box, password text box, check box, radio button, submit button, reset button, hidden input field, image (which acts as a submit button), file selection box or general button.

```
<form action="somescript.php" />
        <p>Do you like pie?</p>
        <div>yes <input type="radio" name="pie" value="yes" checked="checked"
/></div>
        <div>no <input type="radio" name="pie" value="no" /></div>
        <div>Your name: <input type="text" name="name" /></div>
        <div><input type="image" name="submitimage" src="someimage.gif"
/></div>
</form>
```

- `name` can be used so that the value of the element can be processed.
- `type` can be used to specify the type of input. Values can be `text` (default), `password`, `checkbox`, `radio`, `submit`, `reset`, `hidden`, `image`, `file` or `button`.
- `value` can be used to specify the initial value. It is required when type is set to `checkbox` or `radio`. It should not be used when type is set to `file`.
- `checked` can be used when `type` is set to `checkbox` or `radio` to set the initial state of a check box or radio button to be selected. It must be used in the format `checked="checked"`.
- `maxlength` can be used to specify the maximum number of characters allowed in a text box.
- `src` can be used when `type` is set to `image` to specify the location of an image file.
- `alt` can be used when `type` is set to `image` to specify the alternative text of the image, which should be a short description.
- `accept` can be used when `type` is set to `file` to specify which file-types should be accepted. This is a comma-separated list of MIME types.
- `disabled` can be used to disable an element. It must be used in the format `disabled="disabled"`.
- `readonly` can be used to specify that the value of the element can not be changed. It must be used in the format `readonly="readonly"`.
- `accesskey` can be used to associate a keyboard shortcut to the element.
- `tabindex` can be used to specify where the element appears in the tab order of the page.

<textarea>

A multi-row text area form element. The initial value of the text area can be placed in between the opening and closing tags.

```
<form action="somescript.php" />
        <p>Your address</p>
        <div><textarea name="address"cols="30" rows="4"></textarea></div>
        <div><input type="submit" /></div>
</form>
```

	Required Attributes • rows is used to define the number of viewable rows. • cols is used to specify the number of viewable columns. **Optional Attributes** • name can be used so that the value of the element can be processed. • disabled can be used to disable the element. It must be used in the format disabled="disabled". • readonly can be used to specify that the value of the element can not be changed. It must be used in the format readonly="readonly". • accesskey can be used to associate a keyboard shortcut to the element. • * tabindex can be used to specify where the element appears in the tab order of the page.
<select>	
	A drop-down list form element. option elements within the select element define each list item.

```
<select name="dogs">
        <option>Domestic Dog</option>
        <option>Arctic Fox</option>
        <option>Maned Wolf</option>
        <option>Grey Wolf</option>
        <option>Red Fox</option>
        <option>Fennec</option>
</select>
```

	Optional Attributes • name can be used so that the value of the selected option element can be processed. • size can be used to specify how many items of the list are displayed at any time. The default is 1. • multiple can be used to specify that more than one item from the list can be selected. This must be used in the format multiple="multiple". • disabled can be used to disable an element. It must be used in the format disabled="disabled". • * tabindex can be used to specify where the element appears in the tab order of the page.
<option>	
	Defines an option of a select form field.

```
<select name="dogs">
        <option>Domestic Dog</option>
        <option>Arctic Fox</option>
        <option>Maned Wolf</option>
        <option>Grey Wolf</option>
        <option>Red Fox</option>
        <option>Fennec</option>
</select>
```

	Optional Attributes • selected can be used to specify that the option is initially selected. It must be used in the

	format selected="selected". • * value can be used to specify a value for the option. If value is not used, the value of the option element is set to its contents by default.	

<table>
<tr><th colspan="3">Other</th></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td><script></td><td></td><td></td></tr>
<tr><td></td><td colspan="2">Used to define a scripting language, such as JavaScript.</td></tr>
<tr><td colspan="3"><script type="text/javascript" src="somescript.js"></script>
<script type="text/javascript">
function koala() {
alert('KOALA! KOALA!');
}
</script></td></tr>
<tr><td></td><td colspan="2">Required Attributes
• type is used to specify what type of scripting language is used. This takes the form of a MIME type such as text/javascript.

Optional Attributes
• src can be used to specify an external source of a script file.
• charset can be used to specify the character set of the element.
• defer can be used to specify that the script does not generate any document content so that the browser doesn't have to worry about it while the page loads. It must be used in the format defer="defer".</td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td><!-- --></td><td></td><td></td></tr>
<tr><td></td><td colspan="2">Comment tag</td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

XHTML Rules Overview:

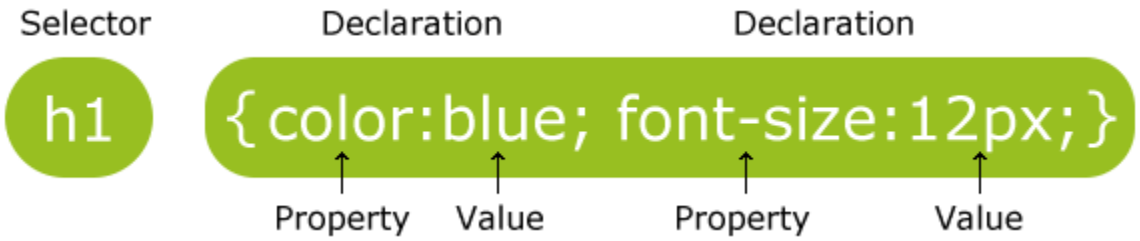
1. Declare a DOCTYPE
2. Declare an XML namespace.
   <html xmlns=http://www.w3.org/1999/xhtml lang=”en” xml:lang:”en”>
3. Declare your content type.
   <meta http-equiv=”content-type” content=”text/html”; charset=iso-8859-1” />
4. Close every tag, whether enclosing or non-enclosing
5. All tags must be nested correctly.
6. Inline tags cannot contain block level tags.
7. Write tags entirely in lowercase.
8. Attributes must have values and must be quoted.
9. Use encoded equivalents for angle brackets and ampersands within content.

CSS

- Stands for Cascading Style Sheets
- Define how to display HTML elements
- HTML was intended to define the content of a document

CSS is made up of items called “rules”.

A CSS rule has two main parts: a selector, and one or more declarations:



Selector = HTML element you want to style

HTML elements include, <body>, <h1>, <p>, <ul>, <a>, etc.

Declaration: consists of a property and a value

The property is the style attribute you want to change

Declarations always end with a semicolon

Declaration groups are surrounded by curly brackets.

How to use them

CSS rules can be created and attached to a document/html element(s) in three ways:

1. External style sheet
2. Internal style sheet
3. Inline style

External Style Sheet

Used when the style is applied to many pages. You can change the look of an entire web site by changing one file.

```
Example:
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

Internal Style Sheet

Used when a single document has a unique style. The styles used here will over-ride the styles from an external style sheet.

```
Example:
<head>
<style type="text/css">
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```

Inline Style

This type of style mixes content with presentation. It would be used to style an attribute in a relevant tag. The style attribute can contain any CSS property. The style used here will override any embedded or external style that is used.

```
Example:
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

ID and Class

In addition to styling an HTML elements, CSS allows you to specify your own selectors called "id" and "class".

id selector

The id selector is used to specify a style for a **single, unique element**.

The id selector uses the id attribute of the HTML elements, and is defined with a "#".

The style rule below will be applied to the element with **id="para1"**

Example:

```
#para1
{
text-align: center;
color: red;
}

<p id="para">My text</p>
```

The id selector is best used with CSS layouts as you want to apply a specific style to an element that exists on a page.

class Selector

The class selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on more than one element.

It allows you to set a particular style for any HTML elements with the same class.

The class selector uses the HTML class attribute, and is define with a "."

In the example below, all HTML elements with **class="center"** will be center-aligned:

Example:

.center {text-align: center;}

<h1 class="center">My heading</h1>

* My content *

You can also specify that only specific HTML elements be affected by a class.

In the example below, all elements with the **class="center"** will be center-aligned.

Example:

p.center {text-align: center;}

<h1 class="center">My heading will not be centered</h1>
My content will be centered

CSS Items you should know

Some additional basic CSS style information you should be familiar with include the following:

CSS Text

- text color
- text alignment
- text decoration
- text transformation

CSS Font

- Font Family
- Font Style
- Font Size

CSS Links

- Text decoration
- background color

CSS Lists

- list-style
- list-style-type

CSS Box Model

- Margin
- Border
- Padding
- Content

CSS Grouping and Nesting selectors

CSS Display and Visibility

- Hidding an element
- Block and Inline elements

CSS Positioning

- Static
- Fixed
- Relative
- Absolute
- Overlapping

CSS Float

- Floating elements next to each other
- Using Clear

Other items to take note of

CSS Pseudo-classes

CSS Pseudo-elements

CSS Navigation Bar

Additional Information:

http://reference.sitepoint.com/html

http://reference.sitepoint.com/css