Final Review:

**title tag**

XHTML documents must have a <title> element within the header section

The <title> element has been around since HTML 2.0, and although recommended for any web page, it is not required by the browser. If omitted, the viewer will see 'untitled' in the browser's title bar. In XHTML, the <title> element is required.

every HTML page must have a TITLE tag within the HEAD. This TITLE is displayed in the drag-region of the web browser window once the HTML page is loaded. The TITLE of an HTML page is probably the single most important piece of code in HTML as far as the outside world is concerned; this is because of the manner in which search engines (such as AltaVista or Lycos) use the words in the TITLE as keywords for categorization of your web page in their databases

every page MUST have a TITLE, and that TITLE MUST go in the HEAD!

The HEAD tag is the parent object, and the TITLE tag is the child object.

Each individual word in the TITLE of an HTML document receives a separate keyword entry in the database.

Internet search: these links are usually represented by the TITLE of the HTML page in question, the text of that TITLE having been hyper-referenced to the desired page

**meta tag**

<meta charset="utf-8"> HTML5

The **META** tag is used primarily for two purposes today:

1. It helps robotic search engines (such as Google and Yahoo) better categorize your HTML pages.
2. It automatically moves users from one page to another (as seen on splash pages or forwarding-address pages) using its auto-refresh capabilities.

The **META** tag requires two attributes: **NAME** and **CONTENT**.

**NAME** identifies the type or variety of **META** tag in question;

The **NAME** attribute, in some cases, is interchangeable with the **HTTP-EQUIV** attribute

**CONTENT** sets the necessary information for that version of the **META** tag.

either **NAME** or **HTTP-EQUIV** will be used as the identifier attribute for the **META** tag, depending on the purpose of that particular **META** tag; the second attribute will always be **CONTENT**.

The **META** tag should self-close in XHTML, but does not need to close in HTML5. The **META** tag always goes within the HEAD of an HTML page; it is NEVER placed in the BODY. You may place as many different **META** tags in the HEAD as you like.

Each individual word in the TITLE of an HTML document receives a separate keyword entry in the database. Each individual keyword in the **KEYWORDS META** tag receives a separate keyword entry in the database.

 whether the word is in the TITLE (top priority),

 whether the word is in the **KEYWORDS META** tag, and

 whether the word appears in the text in the BODY of the HTML page.

Many search engines also recognize the **DESCRIPTION META** tag. The **DESCRIPTION META** tag allows you to create a brief description of your page or website which will display below the TITLE and URL for your website on a robotic search engine's results page.

<meta name="keywords" content="Steve Rubin, web design, html, critical thinking">   
<meta name="description" content="Home Page of Steve Rubin, CNIT Instructor at CCSF.">

 If the **DESCRIPTION META** tag has been set for that page, then the user sees the CONTENT of the **DESCRIPTION META** tag. If the **DESCRIPTION META** tag has NOT been set, then the user sees the first 10-20 words from the BODY of that HTML page in place of the description.

Meta – auto-refresh feature

<meta http-equiv="refresh" content="10; url=http://www.yahoo.com/">

Two attributes of the **META** tag are required to utilize the auto-refresh feature: **HTTP-EQUIV** and **CONTENT**. The**HTTP-EQUIV** attribute must be set equal to **"refresh"**. The **CONTENT** attribute must be set equal to a value following this format: **"secondsOfDelay; url=someURL"**

The value of the **CONTENT** attribute is broken into two segments, separated by a semi-colon. The first segment is an integer representing the number of seconds of delay you want once the HTML page which contains the **META** tag has been loaded. The second segment is **url=** followed by the absolute or relative URL desired.

**Background and bgcolor**

Description: The BGCOLOR attribute in the BODY tag can be used to set the background color for the BODY of an HTML page.

Tag: body

Attribute: **BGCOLOR**

Example: **<body bgcolor="#FFFFFF">**

The BODY tag, which represents the visible "document" of a web page, may be modified to have a specific background color using the BGCOLOR attribute.

Attribute: **BACKGROUND**

Value: a URL (relative or absolute) for a GIF or JPEG file

NOTES:

 Background images should not be more than 10K in file size; some browsers will not display foreground content until the entire background image has been downloaded.

You may set both the BGCOLOR and the BACKGROUND attributes for the BODY tag; as the background image is loading, the foreground content will be displayed on a background the color of BGCOLOR.

**Text**

sophisticated language for combining text with pictures and other digital media, which they dubbed "HTML", or "Hyper-Text Markup Language".

 provide a language for displaying text and pictures SIMULTANEOUSLY

"hyper-text", clickable words which could be used to summon OTHER digital resources, such as HTML pages, pictures, and sound files.

HTML (Hyper-Text Markup Language) is a display language, a text-only code which tells a web browser (such as Internet Explorer or Netscape Communicator) how to assemble a web page of text, pictures, and other multimedia content.

HTML or text files are resources, pictures are resources, sound files are resources, Flash animation files are resources; any digital media file is a resource. Resource is a catch-all term for digital media or files of any kind.

**link**

To check whether you have broken links (especially non-clickable links), go to [Link Checker from W3C](http://validator.czweb.org/link-checker.php).

You know you have a link problem, if you click on your validation icon and you get:  
**Sorry! This document can not be checked.** Error

500 No Host option provided

The A (anchor) tag allows you to create hyper-text references on an HTML page. Hyper-text, hyper-reference, hot-link, these terms all mean the same thing: a clickable link on a page (usually a word or a picture) which invokes another HTML resource.

Tag: **A**  
Attribute: **HREF**  
Description: The anchor tag allows you to create an anchor point in the flow of HTML text. The HREF attribute assigns a URL to an anchor tag and makes an active hyper-reference out of the word(s) or picture marked by that tag.

<a href="http://www.yahoo.com/">Yahoo Link</a>

**intra-page link**

Tag: **A**  
Attribute: **NAME**  
Description: The NAME attribute turns an A (anchor) tag into a "jump-to" point, which can be referenced by other A (anchor) tags used as hyper-references/hot-links.  
Example: **<a name="fred">Word</a>**

**Note:** Each NAME on an HTML page MUST be unique (no repeats). A NAME anchor MUST surround some content; it may not be placed into a page without marking something; this content will NOT be affected in appearance (e.g., it will not become "hot" or clickable).

<a href="#fred">Link that jumps to "fred" NAME anchor</a>

<a href="#">This link will jump to the top of the page</a> //broken in IE

Change link color

Tag: **BODY**  
Attribute: **LINK**  
Value: any named color or hexadecimal color code  
Description: use LINK to set the color for unvisited hyper-links.  
Example: **<body link="#0000FF">**

Tag: **BODY**  
Attribute: **VLINK**  
Value: any named color or hexadecimal color code  
Description: use VLINK to set the color for visited hyper-links.  
Example: **<body vlink="#FF00FF">**

Tag: **BODY**

Attribute: **ALINK**  
Value: any named color or hexadecimal color code  
Description: use ALINK to set the color for active hyper-links (links currently being clicked on)  
Example: **<body alink="#FF0000">**

**img**

**mailto**

When creating an absolute URL, you must specify what type of absolute URL you want, called the "scheme". Is this a web address? Is this an ftp address? Is this a telnet address? These are the questions that are resolved by stating the "scheme".  You will be normally be using only two schemes, which are probably the most common types of absolute URLs on the Web:

**http** (Web address)  
**mailto** (email address)

There are two more schemes which you are likely to see, although we will not be using them in this class specifically:

**ftp** (ftp site address for transfer of files)  
**file** (file address on a hard disk or server)

www.yahoo.com

"com" is the domain name.  
"yahoo" is the sub-domain name (which is commonly called the domain name).  
"www" is the name of the server in the "yahoo.com" domain.

"scheme" for a Web address is "http", which stands for "hyper-text transfer protocol". This "scheme" is followed by a colon, then two slashes which form the beginning of the "scheme\_specific\_part". Thus, web addresses always begin with the following code:

http://

"http" is the "scheme".

":" separates the "scheme" from the "scheme\_specific\_part"

"//" is the required prefix for an absolute web address.

Absolute URLs for email addresses involve using the "mailto" scheme mentioned above. The format is simple: "mailto", followed by a colon, followed by the email address desired (NO SPACES!). This would be a mailto URL for my email address:

mailto:srubin45@comcast.net

mailto URLs are normally used to create automatic email messages from a hyper-text link on an HTML page.

**Special Characters**

In order to display reserved characters, you must use character entity references, also known as escape sequences or escape characters.

The syntax for escape characters in HTML is always the same:

The ampersand (**&**) character

 followed by the character code,

and ending with a semi-colon (**;**)

NOTE: The character codes may be represented either by name (named entity) or by number (numeric entity); named entities are special text abbreviations of the character name, while numeric entities are special number codes representing the character preceded by a pound (**#**) sign.

Character: **<** (less-than sign)  
Escape Character: **&lt;** or **&#060;**

Character: **>** (greater-than sign)  
Escape Character: **&gt;** or **&#062;**

Character: **©** (copyright symbol)  
Escape Character: **&copy;** or **&#169;**

Character: **&** (ampersand)  
Escape Character: **&amp;** or **&#038;**

Whitespace: &nbsp;

<http://www.utexas.edu/learn/html/spchar.html>

For full accent list, see  
<http://www.tlt.psu.edu/suggestions/international/web/codehtml.html#accent>

Unordered Lists

 UL stands for "Unordered List", while LI stands for "Line Item"

<ul>

<li>Point 1</li>

<li>Point 2</li>

<li>Point 3</li>

</ul>

Tag: **UL**  
Attribute: **TYPE**  
Values: **disc**, **circle**, **square**  
Description: The TYPE attribute allows one to set the bullet type for an unordered list.  
Example Opening UL Tag: **<ul type="square">**

**HTML5**

The new standard incorporates features like video playback and drag-and-drop that have been previously dependent on third-party browser plug-ins such as Adobe Flash, Microsoft Silverlight, and Google Gears. According to the W3C, HTML5 reached W3C Recommendation in 2012.

The Worldwide Web Consortium (W3C) says it's still on track to release the final HTML5 specification in 2014

Note that many XHTML tags are valid in HTML5.

<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta charset="utf-8">  
<title></title>  
</head>

<body></body>.. closing.

Note: HTML5 is not yet an official standard, and no browsers have full HTML5 support. Many HTML5 tags are not currently supported by all browsers.

**Two new interesting HTML5 elements are the canvas and svg tags.**

The canvas tag will draw a rectangle and can be modified with colored gradient fillings . It is used with javascript.

SVG stands for Scalable Vector Graphics. SVG is used to define vector-based graphics for the Web.

1. Some HTML5 pages won't validate unless you also use some CSS on the page.  
2. Unlike XHTML, statements that should self-close in XHTML are not required to self-close in HTML5. In HTML5, there is no error if the self-closing / is included.

XHTML: <img src="abc.gif" />  
HTML5: <img src="abc.gif"> or <img src="abc.gif" />  
  
3. Unlike XHTML, tagnames and attributes in HTML5 are allowed to be capitalized. For example:  
  
<IMG SRC="abc.gif">

HTML5 is a case-insensitive language, which means that you may use either upper-case or lower-case characters in tag names.

For instance, **<br>** is the same as **<BR>** as far as HTML5 is concerned.

XHTML is a case-sensitive language, with all lower-case tag syntax. In XHTML, all tags must have closing tags.

In XHTML, there is only one acceptable way to code the BR tag: **<br />**

CLOSING

HTML

<img src="http://fog.ccsf.org/~srubin/satthumb.gif" width="170" height="129" alt="saturn" title="saturn">

XHTML

<img src="http://fog.ccsf.org/~srubin/satthumb.gif" width="170" height="129" alt="saturn" title="saturn" />

As of HTML5, all appearance information for an HTML page has been removed from the HTML tags themselves and placed in Cascading Style Sheets (CSS). The sorts of background and text color attributes cited in the examples, therefore, would not be used in BODY tags on websites which have implemented HTML5.

UTF-8 stands for **U**nicode **T**ransformation **F**ormat-**8**. It is an octet (8-bit) lossless encoding of Unicode characters. The following meta statement is highly recommended in HTML5 files.

<meta charset="utf-8">

MULTIMEDIA

Audio Filetypes:

Midi Musical Instrument Digital Interface (MIDI)

similar to an electronic synthesizer.

Wav Wav files are proprietary to Microsoft and IBM and are most commonly used on Windows-based PCs.

Mp3 and Mp4

CD quality songs that are compressed down from huge files to much smaller files without any noticeable decrease in sound quality.

MP4 is short for Moving Picture Expert Group-4

Embedding an audio file on a web page means that the viewer's sound controller will be loaded on the page, at the time that the browser loads the page.

Note that the embed tag has been deprecated and it will **not validate** for XHTML.

<embed src="beethoven.mid" autostart="false" loop="false" width="20%" height="20" />

autostart="false" - means that the midi file will NOT start playing automatically right after the page is loaded. This is generally a good idea. The viewer must click the start button on the controller for the file to play.

loop="false" - means that the midi file will not keep playing continuously after the viewer clicks start.

width="20%" - means that the controller will occupy 20% of the viewer's screen width.

heigth="20" -means the controller will be 20 pixels high.

You can also embed an audio file using just the **object tag**. The object tag **will validate** for XHTML.

**Note:** classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B" is necessary for the audio file to play in an IE controller. IE needs a non-standard value to the valid classid attribute, an identifier to load an associated activeX.

<object type="audio/x-mpeg" data="jungle.mp3" width="200" height="16"> <param name="src"value="jungle.mp3" />

<param name="controller" value="true" />

<param name="autoplay" value="false" />

<param name="autostart" value="0" /> <param name="pluginurl"value="http://www.apple.com/quicktime/download/" /> </object>

For HTML5, you can use the **audio tag**, for example:

<audio controls="controls">  
<source src="jungle.mp3" type="audio/mpeg" />  
<source src="jungle.ogg" type="audio/ogg" />  
Your browser does not support this audio  
</audio>

 For HTML5, the embed tag will not validate. You should use the audio tag or the object tag.

Video Filetypes:

MPEG, which stands for Moving Picture Experts Group, is the name of family of standards used for coding audio-visual information (e.g., movies, video, music) in a digital compressed format. The major advantage of MPEG compared to other video and audio coding formats is that MPEG files are much smaller for the same quality. This is because MPEG uses very sophisticated compression techniques. MPEG file extensions can be .mpg, .mpeg, or .mpe.

MP4 is a multimedia container format standard specified as a part of MPEG-4. mp4

Avi Files:

AVI stands for Audio Video Interface. AVI is used in Windows operating systems to provide sound and video. It may drop frames to keep the sound playing, thereby allowing the format to work on almost any Windows machine. The file extension for AVI files is .AVI.

Quicktime Files:

QuickTime is Apple's award-winning, industry-standard, software architecture for creating, playing and streaming digital media for Mac OS and Windows. The file extension for quicktime files is either .qt or .mov.

WMV Files:

Windows Media Video (WMV) is a generic name for the set of video codec technologies developed by Microsoft.

Embedding a video file on a web page means that the viewer's associated video controller will be loaded onto the page, at the time that the browser loads the page.

<embed src="julia.qt" height="140" width="160" autostart="false" loop="false" controller="true" />

autostart="false" means that the quicktime file will NOT start playing automatically right after the page is loaded. This is generally a good idea. The viewer must click the start button on the video controller for the file to play.

height="140" - means the controller will be 140 pixels high.

width="160" - means that the controller will be 160 pixels wide.

loop="false" - means that the quicktime file will not keep playing continuously after the viewer clicks start.

controller="true" - This adds user controls to the movie.

For HTML5, you can use the video tag, for example:

<video width="200" height="150" controls="controls">  
<source src="chimp.mpeg" type="video/mpeg">  
<source src="chimp.ogg" type="video/ogg">  
Your browser does not support the video tag.  
</video>

 The HTML5 <video> element also has methods, properties, and events.

There are methods for playing, pausing, and loading, for example. There are properties (e.g. duration, volume, seeking) that you can read or set. There are also DOM events that can notify you, for example, when the <video> element begins to play, is paused, is ended, etc.

 For HTML5, the embed tag will not validate. You should use the video tag or the object tag. If you use the object tag, you must use the type attribute.  
  
For example:  
  
<object height="136" width="155" type="video/qt">  
<param name="src" value="julia.qt" />  
<param name="controller" value="true" />  
<param name="autostart" value="false" />  
  
<object type="video/quicktime" data="julia.qt" height="136" width="155" class="qt">  
<param name="controller" value="true" />  
<param name="autostart" value="false" />  
Error text.  
</object>

Applet tags

<applet code="flame.class" width="250" height="110">

<param name="text" value="Bungle+in+the+Jungle" />

<param name="link" value="http://javaboutique.com" />

</applet>

The first param statement sets the applet's "text" attribute (which customizes the text this applet displays) to be "Bungle+in+the+Jungle". A second param statement will allow the applet to also act as a link, if one clicks the text at the bottom of the applet. If the page is viewed by a browser that can't execute Java applets, then the browser will ignore the APPLET and PARAM tags, displaying only the HTML between the <param /> and </applet> tags (the alternate HTML).

<object classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93" height="200" width="200">   
<param name="code" value="flame" />  
<param name="text" value="Bungle+in+the+Jungle" />  
</object>   
</object>

The **classid** attribute identifies which version of Java Plug-in to use.

For Firefox - classid="java:classname.class"  
  
(where classname.class should be replaced by the actual name, e.g.,   
classid="java:flame.class"  
  
  
For IE - classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"  
(always use this classid for IE, but note that it may not display the applet in some versions/platforms for IE)

For HTML5, the applet tag and the classid attribute are not valid. Instead, you should use the object tag with a param statement that has name="code" and value equaling the class name of the applet.

Here's a valid HTML5 example:

<object type="application/x-java-applet" height="200" width="200">  
<param name="text" value="Bungle+in+the+Jungle">  
<param name="code" value="flame.class" >  
</object>

TABLE tag

<table border="1" width="400">

A TH or TD cell of a table may span more than one column or row. Here's an example:

Tag: **TH, TD**  
Attribute: **COLSPAN**  
Values: positive integers representing the number of columns to be spanned by a cell.  
Description: COLSPAN causes a TH or TD cell to span multiple columns of cells.  
Example: **<td colspan="2">This cell spans two columns</td>**

Tag: **TH, TD**  
Attribute: **ROWSPAN**  
Values: positive integers representing the number of rows to be spanned by a cell.  
Description: ROWSPAN causes a TH or TD cell to span multiple rows of cells.  
Example: **<td rowspan="2">This cell spans two rows</td>**

Color table

<th colspan="3" bgcolor="#9999FF">This cell spans three columns</th>

<td rowspan="2" bgcolor="#9999FF">This cell spans two rows</td>

table align and valign

Tag: **TH, TD**  
Attribute: **ALIGN**  
Values: **left, right, center**  
Example:**<th align="right">Something</th>**

The ALIGN attribute allows you to align content in a cell to the left, the right, or the center.

Tag: **TH, TD**  
Attribute: **VALIGN**  
Values: **top, middle** (or center), **bottom**  
Example: **<td valign="top">Something</td>**

The VALIGN attribute allows you to align content vertically in a cell at the top, middle, or bottom of the cell. Both TH and TD cells are ALWAYS **valign="middle"** by default. Note: The values "middle" and "center" for the VALIGN attribute have an identical effect; "middle" is the official value.

cellspacing

To increase the amount of space in-between the cells, we must also set the CELLSPACING attribute using pixel values; by default, **cellspacing="1"**, which is one pixel of space between the cells; **cellspacing="10"** would be 10 pixels of space.

Tag: **TABLE**  
Attribute: **CELLSPACING**  
Values: integers representing number of pixels between cells.  
Description: creates space between cells in a table.

Another attribute for table: CELLPADDING. Cellpadding is the space between the content of the cell and the edge of the cell. Default cellpadding values vary cross-browser and cross-platform, but are usually about 1 to 2 pixels.

Tag: **TABLE**  
Attribute: **CELLPADDING**  
Values: integers representing number of pixels between content and edge of cell.  
Description: creates space between content in a cell and the edge of a cell.

Another attribute for table: BGCOLOR. Bgcolor allows you to set the background color for a single cell, a row of cells, or an entire table.

Tag: **TABLE, TR, TH, TD**  
Attribute: **BGCOLOR**  
Values: hex codes (**#FFFFFF, #3399CC**) or named colors (red, blue, etc)  
Description: colors tables, table rows, and table cells.

**Graphics**

Tag: **IMG**  
Description: Use the IMG tag to indicate where you want to place a picture in an HTML page. The IMG tag must close in XHTML, as displayed below

Example:

<img src="capitalA.gif" width="54" height="54" />

If you forget to put WIDTH and HEIGHT into your IMG tag, the web browser may prevent the display of anything on the HTML page following that IMG tag until the entire picture is finished downloading.

A pixel is a square or rectangular dot of colored light. Computers use pixels to display visual content to a user; you put enough pixels together, and you have a picture. Pixels are always formed in a square grid, just like graph paper, in rows and columns.

 images for the Web are always 72 pixels per inch.

Image Types:

GIF compression works best with images that have lots of flat color space (no gradations or continuous tones), such as drawings and flat-color graphics.

JPEG compression works best with pictures that are photographic in nature, without sharp edges, and containing lots of graduated, continuous tone color.

PNG is a bitmapped image format that employs lossless data compression. PNG was created to improve and replace the GIF format, as an image-file format not requiring a patent license. PNG is pronounced "ping".

Web-safe colors may only use the following digits (in hex): **00, 33, 66, 99, CC, FF**.

There are only three types of shapes which may be defined as clickable areas in an imagemap: **RECT**, **CIRCLE**, and**POLY**.

Rect:

It is defined by two points on the digital image: the position of the upper-left-hand corner of the rectangle, and the position of the lower-right-hand corner of the rectangle.

Circ:

center point x,y coordinate position and the radius value

There are two parts to a client-side imagemap: the **IMG**tag which displays the image on the XHTML page, and the**MAP**and **AREA**tags (invisible to the user) which contain all of the imagemap information. The **A**(anchor) tag is NOT involved in creating a client-side imagemap.

<map>

<area />

<area />

<area />

</map>

Tag: **MAP**  
Attribute: **NAME**  
Value: any valid name (based on the naming rules discussed in earlier modules)  
Description: the **NAME**attribute of the **MAP**tag provides an identifier which the **USEMAP**attribute of the **IMG**tag will use to connect itself to the imagemap information in the **MAP**.  
Example (abbreviated): **<map name="fred"></map>**

Note: In order for your XHTML pages to validate with the W3C standards, you need to insert an id attribute into your map statement. For example:  
  
<map name="fred" id="fred">

The **AREA**tag has three required attributes: **SHAPE**, **COORDS**, and **HREF**. The **HREF**attributes are identical to their **A**(anchor) tag counterparts except for the fact that they work within an imagemap.

Tag: **AREA**  
Attribute: **SHAPE**  
Value: **rect**, **circle**, **poly**  
Description: sets the shape of the clickable area which the individual **AREA**tag will define. Each **AREA**tag in a **MAP**may be a different **SHAPE**, or they may all be the same **SHAPE**.

Tag: **AREA**  
Attribute: **COORDS**  
Value: integer **x,y**(and **r**) coordinates separated by commas; the number of integers required depends entirely on the value of the **SHAPE**attribute for that **AREA**tag.  
Description: the **COORDS**attribute of the **AREA**tag defines the **x,y**(and **r**) coordinates for the shape defined by the**SHAPE**attribute of that particular **AREA**tag.

Tag: **AREA**  
Attribute: **HREF**  
Value: any URL, relative or absolute  
Description: the **HREF**attribute of the **AREA**tag defines the hyper-reference for that clickable area in the imagemap. Note that HTML5 does not require the area tag to self-close.

<img src="demoMap.gif" width="200" height="100" alt="Demo Map" border="0" usemap="#ethel" />

<map name="ethel" id="ethel">

<area shape="rect" coords="7,7,71,71" href="destination1.html" />

<area shape="circle" coords="127,202,25" href="destination2.html" />

<area shape="poly" coords="80,247,41,187,39,131,45,111,84,124" href="destination3.html" />

</map>

CGI

perl – cgi-bin

<form action="formA.php" method="post"> php file must be in php directory

FORM

<form method="post" action="mailto:srubin45@comcast.net"

enctype="text/plain">

<!-- all form element statements -->

</form>

Note: enctype="text/plain" should be included in your form statement when sending unformatted email. It specifies the MIME type of the data to be sent by the post method.

If you use Internet Explorer, the email that you receive will have as its header, "Form posted from Microsoft Internet Explorer". If you use Firefox, the email that you receive will have as its header, "Form posted from Firefox". If you do not include enctype="text/plain" in your form statement, you will be able to open the email but the data will occur in a string that will be somewhat hard to read. Regardless, the file will be called, POSTDATA.ATT. If you do include enctype="text/plain", when you open the email, you will see something that is easy to read. The above description is also true for hills email opened through pine.

TEXTAREA requires three attributes: NAME, COLS, and ROWS. NAME names the TEXTAREA so that it can be accessed by the script, COLS determines the width of the TEXTAREA, and ROWS determines the height of the TEXTAREA. The TEXTAREA tag always closes.

<textarea name="theMessage" cols="40" rows="4"></textarea>

In addition to the button INPUT, there are two final variations of the INPUT tag, **FILE** and **HIDDEN**. The file INPUT allows the programmer to create a text field which may be used to find and submit files to a server. The hidden INPUT is invisible to the user entirely and has many uses

**Remember**: only ONE radio button may be pre-checked, while as many checkboxes as you like may be pre-checked.

CSS

A CSS document is nothing but a plain text (ASCII) file, saved with the ".css" file name extension; remember to follow all of the regular naming rules for any file name on the Web (NO SPACES IN THE NAME!).

Example CSS Document Names:

myStyles.css

linksty.css

steve.css

Style sheet properties are divided into two parts:

1. property name
2. value

property-name:value;

declaration:

tagname { property-name:value; property-name1:value1; property-name2:value2; }

selector { property-name:value; }

h1, h2, h3 { text-align:center; color:#000000; }

You must be VERY careful NOT to repeat properties for the same tag within separate declarations.

// this is a single line comment

Example (multiple line comments):

/\* this is multiple line comment syntax on a single line \*/

Having said this, single-line comments are broken in CSS; do NOT use single line comment syntax!

Contextual selectors allow you to define the appearance for HTML tags in parent-child relationships with other HTML tags.

p b { color:#660066; font-weight:bold; }

Using contextual selectors, I can define how an HTML tag will look when it is IN CONTEXT of another specific HTML tag (in other words, when it is a child of another tag).

Linked style sheets are the preferred form of style sheet. An external ASCII/plain-text document (named with the ".css" file name extension) contains the style sheet information. In general, use linked style sheets whenever possible; they are the most flexible, the most portable, the most convenient style sheet.

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

Tag: **LINK**  
Attribute: **REL**  
Value: **stylesheet**

Tag: **LINK**  
Attribute: **TYPE**  
Value: **text/css**

Tag: **LINK**  
Attribute: **HREF**  
Value: any URL, relative or absolute, to the external CSS document that you want to link to the HTML page.

Embedded style sheets use CSS declarations placed within a STYLE tag in the HEAD of a particular HTML page.

<head>

<style type="text/css">

p { font-size:14pt; color:#000000; }

b { font-weight:bold; }

</style>

</head>

Inline style sheets place style information in a **SPAN** tag inside the BODY of an HTML page, and are used in much the same way as the old **FONT** tag. Inline style sheets are the least flexible of all style sheets, and have all of the inconvenience of the old **FONT** tag (although with more control and power). Inline style sheets apply to only one single element inside the BODY of an HTML page (the **SPAN** tag), and must all be changed individually when a change to the look-and-feel of the site is proposed. Inline style sheets can only use a limited portion of basic CSS syntax; they do not affect HTML tags on a page as a whole, they merely affect whatever is marked by their particular**SPAN** tag.

The **STYLE** attribute may ALSO be added to ordinary HTML tags, as in the following example:

<p style="font-size:36pt;">Here is some more text.</p>

<p>Here is <span style="font-size:36pt; color:#660066; font-family:'Arial', 'Helvetica', sans-serif;">some more text.</span></p>

css class

.warning { color:#FF0000; }

<p class="warning">This paragraph has had the warning class applied

to it.</p>

pseudo-class was developed to define tags with more than one state.

The only tag in HTML that has more than one state is the **A** (anchor) tag, which has three official states: link, visited, and active.

a:link { color:#0000FF; }

a:visited { color:#FF00FF; }

a:active { color:#FF0000; }

a:hover { color:#000000; }

a.x:hover {color: #000000; background-color: yellow; font-weight: bold;}

<a href="#">Hover does nothing</a><br /><br />  
  
<a class="x" href="#">Hover with class="x"</a>

To turn OFF the underline, set the text-decoration property to the none value.

Example (abbreviated):

text-decoration:none;

on:

text-decoration:underline;

Individual values in CSS NEVER have spaces in them. The only exception to this rule is in regard to regular font names like "Times New Roman" and "Courier New", which must be enclosed in quote marks.

Pixels are considered to be relative units because monitors display pixels at different sizes. Despite this, pixels are an absolute quantity in CSS, because 1 pixel is 1/72nd of an inch as far as web pages are concerned.

Percentage Values: **50%**, **100%**, etc.

Absolute Values: **in**, **cm**, **mm**, **pt**, **pc**

Relative Values: **em**, **ex**, **px**

As with the deprecated **FONT** tag with its **FACE** property, you need to give users more than one font choice, in case the first choice font is not loaded on their system.

font-family:"Times New Roman", Times, serif;

font-weight:bold;

font-style:italic;

text-decoration:underline;

text-indent:.5in;

line-height:24px;

line-height:normal;

The line-height property sets the "leading" for text, creating space between lines in a paragraph. Set to percentage values, pixel values, or "normal".

text-align:left;

To support all browsers consistently, use only the following properties:

margin

margin-left

margin-right

margin-top

margin-bottom

padding

border

don't use these properties with the IMG tag (in fact, do NOT define the IMG tag in CSS at all, PERIOD, or you'll be sorry!).

#### Property: margin

Example:

margin:.5in;

Values: any valid unit of measurement, relative, absolute, or percentage.

The margin property sets all sides of the margin evenly (top, bottom, left, right). If you wish to set the individual sides separately, don't use the margin property; instead, use the specific margin-left, margin-right, margin-top, and margin-bottom properties.

#### Property: padding

Example:

padding:.5in;

Values: only positive values using any valid unit of measurement, relative, absolute, or percentage.

#### Property: border

Example:

border:1px solid #003300;

The border property requires three values: the width of the border, the style of the border, and the color of the border, each value separated by a space. You can use pretty much any value for width (although pixel values work best). There are only a couple of values for the border style which work with the version 4 browsers, the best being "solid". The color of the border can be anything.

In the most modern browsers, you may set the sides of the border individually; this is not supported in the version 4 browsers, however, so be warned.

Note: I must again emphasize the importance of using these properties VERY sparingly, as they can rapidly get you into trouble.

p { background:url(newsletter.gif); }

An ID is identical, in many ways, to the **NAME** attribute in HTML; in fact, in future browsers, ID will completely replace the **NAME** attribute! An ID is used to identify an element for manipulation in a scripting language, such as JavaScript.

#banana { color:yellow; }

<p id="banana">This is the banana paragraph.</p>

CSS-P (or Cascading Style Sheets Positioning, a part of the unevenly implemented CSS2 standards) allows the web programmer to position HTML-formatted resources anywhere they like on a web page.

Only one type of HTML tag should normally be positioned: the DIV tag.

#fred {

position:absolute;

left:100px;

top:30px;

}

elements are positioned from THEIR upper-left-hand corner in relation to the upper-left-hand corner of the web page.

Property: **left**  
Values: integer-based pixel values (i.e. 10px, 172px, 59px, etc)  
Description: sets the x-coordinate for the positioned element, in pixels.  
Example: **left:10px;**

Property: **top**  
Values: integer-based pixel values (i.e. 10px, 172px, 59px, etc)  
Description: sets the y-coordinate for positioned element, in pixels.  
Example: **top:25px;**

ALWAYS define left and top properties using integer pixel-based values.

Again, position, left, and top properties are all REQUIRED when using CSS-P.

The higher the z-index value is, the farther in front an element becomes; the lower the z-index value is, the farther in back an element becomes. You may not use numbers with decimal places.

Property: **width**  
Values: any positive integer-based pixel value (100px, 250px, etc).  
Description: defines the width of a positioned element.

Property: **height**  
Values: any positive integer-based pixel value (100px, 250px, etc).  
Description: defines the height of a positioned element.

Property: **background-color**  
Values: any hex code color value (#CC66CC, #FFCCCC, etc)  
Description: defines background color for a positioned element in IE or Firefox.

use with background color::

Last, but not least, you MUST set the border property for the positioned element. The border property ensures that the box will be solid and opaque on all browsers. Without the border property, the box will not be cross-browser compatible.

Example: **border:solid 1px #CC99CC;**

Child nested elements are positioned in relation to the PARENT element, NOT to the web page as a whole. Therefore, "childDiv" is 10 pixels from the left and 10 pixels from the top of the PARENT element's upper-left-hand corner.