# Introduction to HTML and HTML 5

This section will cover HTML and basic tags, HTML and XHTML versions, and HTML 5, the latest version and finally HTML editors. Web pages are created using markup code called HTML You can see that the strong tag is used to make the text bold**. <strong>Welcome</strong>** Home would read, **Welcome** Home.

Tasks

* Describe the structure of a web page
* Use HTML to create a web page
* Use validation tools to verify HTML code
* Compare HTML and XHTML versions
* Insert images, tables, frames and forms on a web page
* Identify tags and attributes in HTML 5
* List software applications used to create web pages
* Describe accessibility standards (briefly, we’ll also cover this later in the course)

## Creating Web Pages with HTML

*Original intent of HTML*: HTML was used because it would provide support for browsers to render web pages on various platforms, including Macintosh and PC's. The World Wide Web Consortium maintains the standards for HTML and other web related technologies. [www.w3.org](http://www.w3.org/)

Here comes the best part - creating that web page! You can use any text editor such as Notepad. In your homework activity you will create web pages. You will read about HTML in your readings, but here is an overview of the steps.

## Creating a web page

Simply open **Notepad**. For most users go to **Start**, then **Accessories**, then click **Notepad**. Then you type in the code in the page to create the basic web page structure. All web pages have a **heading** and **body** section. The **title** in the heading is used to display text in the blue title bar in the browser window. The page uses the contents in the body section to store the text and content that will be displayed in the browser. It is useful to type your text content first, and then insert the tags if you have a lot of text.

<!DOCTYPE html>

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

<!-- Created by: Katie Kalata -->

<!-- Last Modified: 9/9/2011 -->

<head>

<meta charset="UTF-8">

<title>My Company Web Site</title>

</head>

<body>

<h1>Welcome to My Web Page </h1>

<p>

<a href="http://www.google.edu/">Google</a>

</p>

<br />

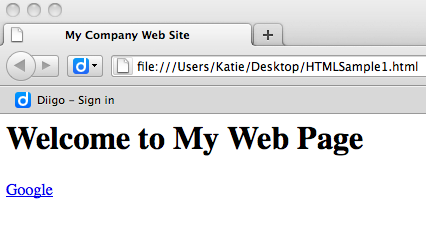
</body>

</html>

Here is a sample web page. Notice how the code is indented.

Figure Your First Web Page

Notice the **indentation** that helps identify the **nested** HTML elements. Some elements format content styles, like bold, italic and underline in Word, and some are used to enclose content, like tables, headings, and paragraphs. All elements are **lower case** and have a **beginning and closing tag**. It's useful to indent the nested tags for readability. However, this is not required by XHTML. Also notice that when the hyperlinks (also known as links) when they start with http, they are absolute URLs (web addresses). However, you can use relative addressing. This includes using the ../ shortcut that many of you may know from DOS. Notice that the **attributes** are all in quotation marks. You also can only put the comment after the opening html tag. There is a structure to these elements, for which elements can be nested within other tags.

HTML Tags are called **elements**. Elements often delimit content and specify something about how the content should be arranged in the document**. Attributes** provide additional information about the content of an element. **Controls** are elements and their attributes.

Save the page using the file extension .**html**. The file extension htm may also be used for the web page. For example "HTMLSample1.html" Make sure to use quotation marks when you save the file.

View the page in a browser. In **Internet Explorer**, you go to the **File** menu and select **Open**, and then click the **Browse** button to locate your page, and then click the **OK** button. Your web page is not on the Internet, so no one except you can view it.

Figure Preview Your Web Page Across Browsers

Your web page will look different depending on your browser and your settings.

# Browser Testing

Today it is important to test your web pages across the major browsers and if possible different platforms and resolutions, monitors and devices. Mac OS users may also test on **Safari**. Always test using the most recent version available. In real scenarios, the quality teams will also test the web site code on multiple versions of the browsers. Some tools make testing easier today. No matter how you test, TEST! You cannot expect your teacher ONLY to use Internet Explorer or FireFox because you use them!

1. So, test your web pages at minimum in **FireFox** (the most recent versions) and **Internet** **Explorer**.

The rest of this lecture is just learning about more HTML elements like p and br and attributes like href. In your readings, you will learn all about the first three lines of code, which identify that this is an XHTML web page, and all about the HTML attributes and elements. But, the process to create a web page is always the same.

Here is a tip. You may want to create a basic HTML page and save it as template.htm. Then any time you need to create a page, such as for your homework, you don’t have to retype the DocType or other basic HTML code! Make sure to include comments with your name, date, file name or other indicators that your instructor may require.

# Validate Your Web Page Code

So, now that you have a web page, you need to make sure that you entered the code correctly. Browsers will still display pages if the code is wrong. But sometimes it displays them incorrectly. It’s a toss up. So, to be sure, we validate the code in the web page. Sometimes editors help verify the code during development.

If you are looking for a review of the tags, check out the W3Schools HTML 4.01 / XHTML 1.0 Reference at <http://www.w3schools.com/tags/default.asp>. You can see that some tags (elements) are deprecated.

An alternate [HTML 5 Validator](http://hsivonen.iki.fi/validator/html5/)/conformance checker is available in beta.

One commonly used validation tool is the **W3C HTML Validation Service.**

* 1. Go to <http://validator.w3.org/>
  2. Click on **Validate File by Upload**.
  3. (or just go directly to the page at [http://validator.w3.org/#validate\_by\_upload](http://validator.w3.org/))
  4. Click on **Browse**
  5. Locate your **Sample1.html** file.
  6. In the **More Options**, select **HTML 5** for DocType. This will validate the code using the experimental validation for HTML 5. It’s only experimental because the final specifications have not been formally approved.
  7. Click the **Check** button.
  8. Fix any errors based on the messages returned and try it again until you have no errors.

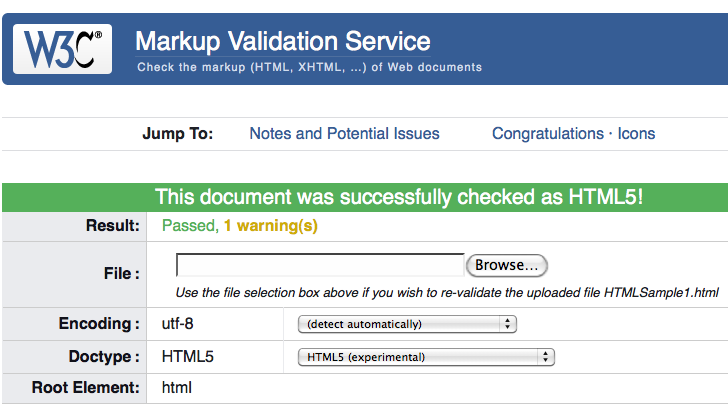


Figure Validate Your Web Page HTML Code

# HTTML, XML and XHTML

## Caution!

Some of this content may seem overwhelming. We do not expect you to memorize all of the details of HTML versus XHTML.XML. But you should have a good understanding of the issues, and reference this document when you run into issues programming or debugging your code.

Before going more into HTML you should know something about the HTML versions. Why? Because many companies must support many types of browsers and older versions, often called legacy browsers. The examples in this class will be using HTML 5. However, we will also include information on XHTML, which has been commonly used for many years. First lets discuss the reason why XHTML came into existence and then what the difference is between XHTML and HTML.

## Early Versions of HTML

Early on, Netscape Navigator and Microsoft Internet Explorer supported different versions of HTML. During the browser wars, Microsoft supported HTML 4.0 standards, and free and Netscape didn't fully support HTML 4.0.

1. **HTML 3.0** – extended basic HTML and followed with updates as **HTML 3.2**, which was completed in 1997.
2. **HTML 4.0** – 1997 introduced many new features but was completed in 1998.
3. **HTML 4.01 –** 1999 included some minor changes. This version continues to be used by many web sites.

## XHTML

The newest version of HTML is called **XHTML**. XHTML version 1.0 is a combination of HTML 4.0, updated to support the **Extensible Markup Language (XML).** XML is a meta-markup language and has a set of rules that provide more stability with HTML but no presentation details. For example, the tag to insert a paragraph (<p>) didn't require a closing tag (</p>). With XHTML, all tags have to have a beginning and closing tags. A simple and universal way of representing data of any textual kind, used to create a new markup language for a particular purpose or area. Because the tags are designed for a specific area, they can be meaningful.

XML has become integral to many technologies, including facilitating the ability to transfer data across web applications. In addition to your textbook readings, you should read about XML technologies. XML is used to store the data, and XSLT is used to manipulate and display the data. XML is a meta-markup language and self-describing. You will learn more about this next week. However, you need to know about the XML rules, because they apply to all web pages that are created using XHTML. The headings on will cover this for you.

## XHTML Rules

Review – what are some differences between HTML and XHTML? XML has a set of basic rules.

1. A simple and universal way of representing data of any textual kind - No presentation details are required.
   1. Used to create a new markup language for a particular purpose or area. So you will see additional markup languages like MathML and SVG (Structured Vector Graphics) which allows you to ‘draw’ on the page using data and other graphic components that you could not do with HTML
   2. Because the tags are designed for a specific area, they can be meaningful
2. Must be **Well Formed**   
   1. XML has a set of rules that apply to all web pages.
   2. First line usually contains a **processing instruction**, which begins with the characters **<?** to declare the character encoding set (such as Unicode) and XML statement.
   3. When you are working with XHTML, tags, or elements, are also called **nodes**.
   4. **Element nesting** - The nodes must be nested correctly in the correct Nesting Order.
      1. You can't say <b><i>Welcome</b></i>. This worked in HTML browsers, but is not valid XHTML.
   5. **Case sensitivity** - XHTML is case sensitive! HTML was not, which is why you see so many old examples in all uppercase.
      1. All tags must have a beginning and closing tags that match.
   6. **Closing tags are required**.
      1. All tags must have a beginning and closing tags. All tags must have a **beginning** and **closing** tags that **match.**
      2. Must have matching tags such as <b>Hello World</b>.
      3. XHTML is case sensitive! (HTML was not). Must use lower case.
      4. Elements without closing nodes can be closed using a forward slash in the opening tag.
         1. In XHTML, you can minimize the attribute if there is no contents by adding / like <p /> or <br /> or <hr />
         2. For example use <img src="window.gif" /> to close the image tag.
         3. Insert a line break with <br />
3. **Syntax rules** must be complied with.
   1. **Quoted** attribute values – attributes must be in quotes
   2. **Explicit** attribute values – You have to have a value for each attribute.
      1. So for a checkbox to configure it as selected, or checked use this code:
      2. <checkbox selected=”selected” text=”Pick Me”/>
   3. **ID and name** attributes – ID is what is often used with scripting and other program interactivity.

Again, although these seem ‘picky’ you will want to review these rules when you are debugging your code. Often student’s mistakes are because your syntax is incorrect or you did not follow the rules above. While editors can help identify these errors at design time, long term knowing about them will help you be a better web developer. XML has become integral to many technologies, including facilitating the ability to transfer data across web applications. In addition to your textbook readings, you should read about XML technologies. XML is used to store the data, and XSLT is used to manipulate and display the data. You will learn more about this next week. However, you need to know about the XML rules, because they apply to all web pages that are created using XHTML

Review – what are some differences between HTML and XHTML?

1. Case sensitivity
2. Closing tags
3. Quoted attribute values
4. Explicit attribute values
5. id and name attributes
6. Element nesting

## Organization of Elements in XHTML Documents

So, if XHTML must be well formed, that means that the tags must be nested in the correct order. XHTML rules define the nesting order and you will learn more about this as you learn the tags. There is a structure to these elements, for which elements can be nested within other tags.

For now, it’s easy to group tags into two categories, container and leaf nodes. Remember that in XHTML, tags are referred to as ‘nodes’.

1. **Container Nodes**
   1. Container nodes can contain other elements. You can only nest other nodes in a container node.
   2. <p><strong>Welcome</strong> to our house</p>
2. **Leaf nodes**

Figure Sample Nesting Elements in XHTML

* 1. Leaf nodes do not contain other elements.
  2. A horizontal rule element creates a line on the page with the <hr /> tag. This tag is not a container node. You can't nest another element inside of a <hr /> tag. So the <hr /> tag is a leaf node.

In the picture you can see the difference between container and leaf nodes.

You can look at the W3 standards for XHTML fo find the nesting order for each tag. But, if you use an editor like Visual Studio or Dreamweaver or Expressions Web, the editor will check that for you.

**Block and Inline Elements**

This is just another way to think about elements.

1. **Block** elements
   1. These block elements are used to enclose content, like tables, headings, and paragraphs.
   2. Some elements will often (but not always) insert a line break at the end of the content when it’s displayed. <span> is an example of a block element, but, it does not insert a carriage return.
   3. **Block** elements such as the P tag, insert a line break at the end of the content when it’s displayed
2. **Inline** elements
   1. Inline elements format content styles, like bold, italic and underline in Word
   2. **Inline** elements format the content and do not move the cursor to the next line.
   3. Block elements CANNOT be nested inside inline elements

## XHTML Versions

[**XHTML 1.0**](http://www.w3.org/TR/xhtml11/) **– 2000,** is a combination of HTML 4.0, updated to support the Extensible Markup Language (XML).

1. **Transitional** – This version still supports deprecated elements and is more commonly used.
2. **Strict** – This version does not support deprecated elements.
3. If you are looking for a review of the tags, check out the [W3Schools HTML 4.01 / XHTML 1.0 Reference](http://www.w3schools.com/tags/default.asp). You can see that some tags (elements) are deprecated.
4. **Frames** are used to show two or more web pages at one time, within one browser window. If you use frames, which we will talk about later, this version will support them.

[**XHTML 1.1**](http://www.w3.org/TR/xhtml11/) **–** 2001 <http://www.w3.org/TR/xhtml11/>

1. Modularized 1.0 into groups. So each module has a collection of nodes (elements/tags) and attributes. But this is only a logical collection. Drops support for frames. Most browsers support it
2. **Transitional** – This version still supports deprecated elements. This is most commonly used in this version.
3. Strict – This version does not support deprecated elements. Errors in syntax are not well tolerated.

[**XHTML 2.0**](http://www.w3.org/TR/xhtml2/)Latest Version <http://www.w3.org/TR/xhtml2/>

1. **Status**
   1. XHTML 2.0 never made it to the final stages, but it’s interesting looking at the organization of the modules. Why? HTML 5 actually includes a similar organizational structure.
   2. Never was fully accepted and work on the [Working Draft](http://www.w3.org/TR/2006/WD-xhtml2-20060726/) 26 July 2006
   3. XHTML 2.0 contains even more changes than XHTML 1.0.
   4. 2004 Apple, Mozilla, and Opera wanted to work on a different approach and formed [WHATWG](http://wiki.whatwg.org/wiki/Main_Page)
2. **XHTML 2 logically organizes elements into modules.** 
   1. Many elements and attributes are deprecated, no longer supported in the standards.
   2. Nesting elements becomes even more formalized.
   3. In XHTML modularization each module is organized and pertains to a group of elements.
   4. You have already seen the tags in the **Document** Module. The main modules are Document which has to do with the base structure of the document, **Structural** which are elements that contain other content, **Text** used for formatting text, **Hypertext** for links and List for managing data, ordered and unordered lists, **Forms** contains form elements.

**Resources:**

Figure XHTML 2 Modules and Elements

1. [Latest Version](http://www.w3.org/TR/xhtml2/)
2. [Major Differences with XHTML 1](file://localhost/%C3%90%09http/::www.w3.org:TR:2006:WD-xhtml2-20060726:introduction.html#s_intro_differences)
3. [List of all elements](file://localhost/%C3%90%09http/::www.w3.org:TR:xhtml2:elements.html)

**Reference**: The picture is from the W3 XHTML 2 Working Draft Document.

[**HTML 5**](http://www.w3.org/TR/2010/WD-html5-20100304/) – 2011

You can see the [HTML 5 Overview of the Standards](http://dev.w3.org/html5/spec/Overview.html) from the W3.

1. [Web Developers Guide](http://dev.w3.org/html5/html-author/) – This is a very useful document for web developers.
2. [HTML5: Edition for Web Authors](http://www.w3.org/TR/2011/WD-html5-author-20110809/) – Aug 9, 2011
3. [XHTML versus HTML](http://wiki.whatwg.org/wiki/HTML_vs._XHTML) from WHATWG.org and [www.html-5.com](http://www.html-5.com)
4. [HTML 5 Standards](http://www.w3.org/TR/html5/)
5. A document describing the [difference between version HTML 4 and 5](http://www.w3.org/TR/html5-diff/) You should review [HTML, XHTML versus HTML 5](http://dev.w3.org/html5/spec/Overview.html#html-vs-xhtml) and <http://www.w3.org/standards/webdesign/htmlcss>.

**Other references**

**HTML Version 5** standards is described by the W3 organization at <http://www.w3.org/TR/2010/WD-html5-20100304/>. A document describing the difference between version 4 and 5 is at <http://www.w3.org/TR/2010/WD-html5-diff-20100304/>. Notice that tags such as Applet are no longer used. However, browsers will support them and for this course, you may use either the applet or object tag. However, many of the attributes and tags that are for presentation that are listed in this document, should be avoided in your web pages. They are better managed in the style sheet. A copy of this document is located in Doc Sharing.

You can see the HTML 5 standards from the W3 at <http://dev.w3.org/html5/spec/Overview.html>.

Tabular Data - <http://dev.w3.org/html5/spec/Overview.html#tabular-data>

Forms - <http://dev.w3.org/html5/spec/Overview.html#forms>

You’ll see the forms have new tags and attributes too. As you can see below, the input now has other types such as tel and email and time. These can help with mobile compatible web site applications. So you can see that HTML is always changing! You should review HTML 5 at <http://dev.w3.org/html5/spec/Overview.html#html-vs-xhtml> and http://www.w3.org/standards/webdesign/htmlcss . A validator/conformance checker in beta is available at <http://hsivonen.iki.fi/validator/html5/> for version 5.

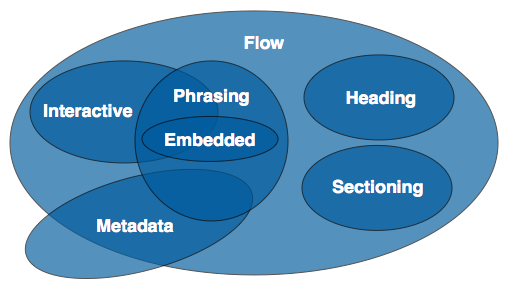
**Details on Specific Elements**

You will learn more about the individual elements for HTML 5, but you can also reference them easily in the W3 documents. Many web classes teach the basics, but do not explain where these rules come from. Some administrators have made decisions based on incorrect interpretation of the standards. So it is always useful to know where these standards are located. They are open standards, freely available on the web.

1. [Tabular Data](http://dev.w3.org/html5/spec/Overview.html#tabular-data)
2. [Forms](http://dev.w3.org/html5/spec/Overview.html#forms)

Notice that tags such as Applet are no longer used as they are **deprecated**. However, browsers will support them and for this course, you may use either the applet or object tag. However, many of the attributes and tags that is for presentation that is listed in this document, should be avoided in your web pages. They are better managed in the style sheet, which we will cover later in the course.

You’ll see the forms have many new tags and attributes too. The **input** tag, which we have used for text boxes on forms, has other types such as **tel** and **email** and **time**. These can help with mobile compatible web site applications. So you can see that HTML is always changing!

**Organizing** **Elements – Reference Only**

HTML 5 organizes elements into logical groupings instead of modules. W3 even included in the draft that they plan to list the elements within each category. Notice that some elements/tags will overlap content areas. Use this page simply as a reference.  
  
Reference: <http://www.w3.org/TR/html5-author/content-models.html#content-models>

**Metadata content**

1. Base, command, link, meta, style, script, noscript, title

Figure Organization of Elements in HTML 5

1. But noscript is now out – so the list may not be up to date! Remember this is in development.

**Embedded content**

1. Audio, canvas, embed, iframe, img, obj, math, svg, object, video

**Sectioning content**

1. Article, aside, nav, section

**Interactive content**

1. A, audio, button, details, embed, iframe, img, input, keygen, label, menu, object, select, textarea, video

**Heading content**

1. H1, h2, h3, h4, h5, hgroup

**Flow content**

a

abbr

address

area

article

aside

audio

b

bdi

bdo

blockquote

br

button

canvas

cite

code

command

datalist

del

details

dfn

div

dl

em

embed

fieldset

figure

footer

form

h1

h2

h3

h4

h5

h6

header

hgroup

hr

i

iframe

img

input

ins

kbd

keygen

label

map

mark

math

menu

meter

nav

noscript

object

ol

output

p

pre

progress

q

ruby

s

samp

script

section

select

small

span

strong

style

sub

sup

svg

table

textarea

time

u

ul

var

video

wbr

Text

**Phrasing content**

a

abbr

area

audio

b

bdi

bdo

br

button

canvas

cite

code

command

datalist

del

dfn

em

embed

i

iframe

img

input

ins

kbd

keygen

label

map

mark

math

meter

noscript

object

output

progress

q

ruby

s

samp

script

select

small

span

strong

sub

sup

svg

textarea

time

u

var

video

wbr

T

# Polyglot Markup – Have it Both Ways

So what about XHTML? Can you make an HTML 5 document XML compliant? Why? In the past I would have worried that browsers will likely move to HTML 5 only versions. But, browsers today must be able to deliver many kinds of documents beyond simple web pages. However, you might have an application that generates the web code using XML tools. The W3C recommends:

“*The choice of HTML or XHTML syntax is largely dependent upon a number of factors the,   
including needs of a given project, the skill set of the developers involved,   
level of support in browsers used by the site’s target audience,   
or it may simply be a matter of personal preference.”*

**Reference**: <http://dev.w3.org/html5/html-author/#choosing-html-or-xhtml>

The overlap is called **polyglot markup**. The goal is to have both a well-formed XML document and a valid HTML document. What can you do to make this happen?

**Polyglot markup** allows you to create a document that conforms to both standards with some minor adjustments. Notice below the markup looks just like the one you created earlier! In this course we will work with HTML 5. However, because web developers also need to support older versions and XHTML, you will also be provided with reference material to the other methods.

<!DOCTYPE html>

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

<head>

<title></title>

</head>

<body>

</body>

</html>

# Basic Markup Sample – TRY IT!

Figure Basic HTML 5 Markup Code

So, what is different from XHTML and HTML 4.01? There are some minor changes that you should be aware of. Although HTML 5 is similar to HTML 4 in that it is loose and not rigid about certain syntax rules but HTML 5 has many new tags, XML is rigid, requiring a well-formed document.

The solution would be then, to maintain the XML requirements except where HTML 5 disallows them. Next let’s look at where these basic conflicts can occur.

## VOID Elements

As you learned earlier, these are self-closing tags.

1. You can minimize the tag such as changing <br></br> to <br />
2. The tags that are classified as VOID elements in HTML 5 and XHTML are:
   1. [area](http://www.w3.org/TR/html5/the-map-element.html#the-area-element), [base](http://www.w3.org/TR/html5/semantics.html#the-base-element), [br](http://www.w3.org/TR/html5/text-level-semantics.html#the-br-element), [col](http://www.w3.org/TR/html5/tabular-data.html#the-col-element), [command](http://www.w3.org/TR/html5/interactive-elements.html#the-command-element), [embed](http://www.w3.org/TR/html5/the-iframe-element.html#the-embed-element), [hr](http://www.w3.org/TR/html5/grouping-content.html#the-hr-element), [img](http://www.w3.org/TR/html5/embedded-content-1.html#the-img-element), [input](http://www.w3.org/TR/html5/the-input-element.html#the-input-element), [keygen](http://www.w3.org/TR/html5/the-button-element.html#the-keygen-element), [link](http://www.w3.org/TR/html5/semantics.html#the-link-element), [meta](http://www.w3.org/TR/html5/semantics.html#the-meta-element), [param](http://www.w3.org/TR/html5/the-iframe-element.html#the-param-element), [source](http://www.w3.org/TR/html5/the-iframe-element.html#the-source-element),
   2. There are two elements [track](http://www.w3.org/TR/html5/the-iframe-element.html#the-track-element), [wbr](http://www.w3.org/TR/html5/text-level-semantics.html#the-wbr-element) which are void elements in HTML 5 but are not used in Polyglot markup because they are not supported in XHTML. So avoid these tags.

Elements, which are not VOID elements that are empty, should not be minimized in HTML 5.

* 1. You can minimize <p /> attributes with XML and <p></p>
  2. Don’t minimize non-void attributes in polyglot markup.
  3. Do not minimize empty content elements so use <p></p> and not <p />

## General Attributes

1. Since so many attributes are used across elements, in HTML 5, there are [General Attributes](http://www.w3schools.com/html5/html5_ref_globalattributes.asp), a collection of attributes that applies across elements such as **ID** and **lang**.
2. You can view the approved [ISO 639-1 language codes](http://www.w3schools.com/tags/ref_language_codes.asp).

## Basic Differences in Elements

Have you ever been to a web site that the web page didn’t load correctly or you needed to download a plug-in to view the page in the browser? Web developers need to know what browsers support and what is different across versions of the hypertext markup languages (HTML/XHTML). Later you will also learn about other markup languages that allow us to expand the browser and web server to deliver math and graphics capabilities through a browser.

HTML does not have to be well formed but XML documents are required to be well formed. So, do this:

* 1. Lowercase for all html element names and values
  2. Use lowercase for ‘standard’ values
  3. Surround all attribute values with single or double quotation marks
  4. ID attribute is lowercase, id, and values cannot have spaces

Don’t use elements that are not supported in both HTML 5 and XHTML.

* 1. For example, <noscript> is not supported in XML documents.

Don’t use attributes that are not supported in both HTML 5 and XHTML. If you want your web site to be supported across the majority of browsers, then try to stay within the standards as much as possible.

* 1. New lines are not allowed in any tag with URL/URI or the <title>
  2. Some elements and attributes are disallowed, but they are not commonly used

## Escaped characters

Web pages are not like Word documents. When you create a web page you are using a markup language. So you have to be careful when you create content that contains special characters that have special meanings within the markup language.

Some characters in HTML are used as delimiters, to identify when the markup code starts and ends. These include the < > greater and lesser then signs. So, we use special character references or codes, to represent the < > signs. Another problem is that when you create the markup code, any whitespace or carriage returns do not impact the content on the page. So, to add blank spaces in your page in Word you would use the space bar. In HTML, you will have to use a character reference, a special code to represent the blank space.

* 1. These are predefined entities need to be escaped so browsers do not interpret them as tags.
     1. So, if you are writing your HTML and you need to display < in the browser, use the &lt;.
     2. <p>If 5>10 then you are correct!</p>
     3. Only **Named Character Entities** allowed.
  2. Below are the five main character references you should memorize. Always uses character references for this group listed on the left.
     1. **&lt;** less than < brackets – The numerical version is &#60;
     2. **&gt;** greater than > brackets
     3. **&amp;** ampersand – The numerical version is &#38; (160)
     4. **&apos;** apostrophe
     5. **&quot;** quotation mark

There are other characters that can also be converted to character codes or numeric codes. Use **numeric** (decimal or hexadecimal) references instead of **literal** characters. Many web editors will insert the codes for you.

* 1. Use decimal values for escape characters such as &#160; for polyglot markup for other characters that you need to escape. But W3 says **hexadecimal** form is **preferred**.
  2. This might be more of an issue with JavaScripts that dynamically change content in a web page.
  3. Reference: [Unicode Charts](http://www.unicode.org/charts/) from Unicode.org and includes other languages, playing cards, emoticons, symbols, dingbats
  4. Another reference for [ISO-8859-1](http://www.w3schools.com/tags/ref_entities.asp), character sets and [symbols](http://www.w3schools.com/tags/ref_symbols.asp)

**Reference Material**

**Hexadecimal # Definition and Numerical character code**

* 1. &#x9; U+0009 for a tab rather than the literal character '\t'
  2. &#x0A; U+000A for a newline rather than the literal character '\n'
  3. &#x0D; U+000D for a carriage return rather than the literal character '\r'
  4. &#xA0; U+0020 **SPACE** for a blank space instead of **&nbsp;**
  5. &#36; The dollar sign $
  6. &#174; The Registered trademark symbol
  7. &#64; The @ symbol
  8. &#xA9; The copyright symbol or &#169; It also used to be &copy; but use the number!

So, now let’s go learn about the individual   
HTML elements and web page programming.

While you are reading, you can open Notepad.

You will be shown code in the textboxes   
that you can type and try out.

# MIME TYPES

**MIME extensions** are not written in code, but you should know what they are. They are used to identify the type of file, so that software applications know how to use it. For example, Word documents are “.doc” for older versions and “docx” for versions 2007 and 2010. You will not write this in the code, but be aware that this information is sent in the header information with your page. Great reference for [MIME types](http://www.iana.org/assignments/media-types/index.html) is IANA, who maintain the standards.

* 1. You must use **text/html** for HTML 5
  2. You must use **application/xml** or **application/xhtml+xml** for XML, such as XHTML.

# XML Version and Character Encoding

This first line, also known as the **preamble or prologue**, identifies the **XML version** and optionally encoding used for the characters in the document. You have likely all heard of Unicode characters. In early versions of computers, the number of bits in a character was 7, and later changed to 8. But, early computers could not handle languages with more than 26 character alphabet. Each character had an assigned byte representation. Remember there are 8 bits in a byte. There simply was not enough characters available. The solution was Unicode, which allows the characters to be stored different and can accommodate international languages. Commonly used character sets include ISO-8859-1 with UTF-8 punctuation or ISO or ASCII charsets

## XHTML

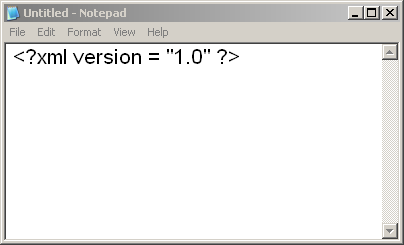
In XHTML you could identify the character set you wanted to use for the page in the XML prologue, on the first line of the web page. The <? Indicated that the statement was a **processing instruction** and would be ignored if not supported. Comments beginning with "<?" are deprecated. But you may see some editors include it if they use older versions of XHTML.

Figure XML Version and Encoding

1. **<?xml version=”1.0” encoding=”UTF-8” />**
2. Later you will learn about meta tags and the Content Type. You can put the code in Content Type meta tags.
   1. **<meta http-equiv="content-type" content="text/html; charset=UTF-8">**

## HTML 5 Coding

This is not used with HTML 5. **XML** declaration is **forbidden** in HTML 5. You can modify the **HTTP Content-Type** in the **TCP/IP header** where the MIME type is defined, **not in the web page.**

* + 1. **Content-type: text/html; charset=utf-8**
    2. **Content-type: application/xhtml+xml; charset=utf-8**

## Polyglot Markup

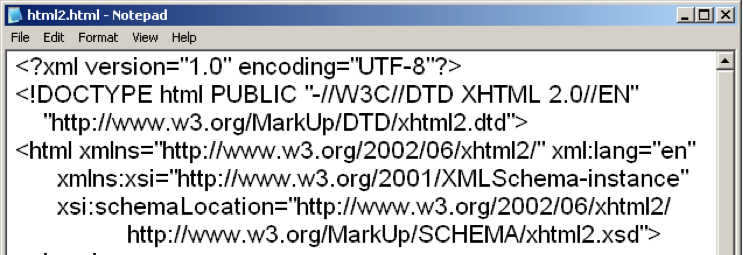
Polyglot markup uses the **UTF-8 character encoding** because it’s the **only** one supported by HTML and XML. Text is said to be in a **Unicode** encoding form if it is encoded in UTF-8, UTF-16 or UTF-32 so as long as you use UTF-8, then it is Unicode compliant. To identify the character encoding put this **into the heading section** of the page. We’ll cover meta tags in a few pages. Only use this code and only in the heading section!

**<meta charset="UTF-8"/>**

# DOCTYPE

XHTML needed a DOCTYPE to be specific for the different versions of XHTML (strict, transitional).

## [DOCYPES for HTML and XHTML](http://www.w3.org/QA/2002/04/valid-dtd-list.html) from W3C

1. **XHTML 1.0** **Transitional** example
   1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/**xhtml1**/DTD/**xhtml1-transitional.dtd**">
2. **HTML 4.01 Transitional** example
   1. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/**html4/loose.dtd**">
3. **XHTML 2**
   1. **xml:lang** attribute used for internationalization

in DocType

Schemas were used to identify the rules to interpret the document structure.

Figure DOCTYPE for HTML 2

## [HTML 5 DOCTYPE](http://www.w3.org/TR/html5/syntax.html#the-doctype)

1. Do not use SYSTEM or PUBLIC
2. There is no ending or closing tag.
3. Use this code:

**<!DOCTYPE HTML>**

1. An optional legacy support example below

**<!DOCTYPE html SYSTEM "about:legacy-compat">**

## Polyglot

1. Use this code:

**<!DOCTYPE HTML>**

TIP – A useful tip is to start looking at the source code of web pages. When you visit a web page, go to the View menu and locate the Source or View Source selection. You should review the HTML from other web sites to see what kinds of options are available and how other programmers created their web sites.

# HTML

**Root Element**

1. The entire document has **1 root node**, which is **<html>**. There can be only 1 root node in a document and everything has to be nested inside the pair of tags.
2. HTML and XHTML documents include HTML, HEAD, TITLE and BODY nodes.

## HTML 4 and XHTML

HTML 4 and XHTML support properties such as **xml:lang**="en" as well as the **lang** attribute. This is permitted in HTML 5 if needed.

<**html xml:lang="en" >**

Figure XHTML and XMLNS Attribute

**Namespace** attributes

1. XHTML must have the **xmlns** attribute to identify the **xml** **namespace** used for the document
2. Don’t use other custom tags or namespaces. If you do, use CSS to change the display: none so they are not included. You can identify the language optionally in the html tag instead of in the meta tag.

**<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" >**

**Default Namespaces –** For information only. You’ll see these used in different kinds of web sites.

* 1. The **HTML** namespace is: <http://www.w3.org/1999/xhtml>
  2. The **MathML** namespace is: <http://www.w3.org/1998/Math/MathML>
  3. The **SVG** namespace is: <http://www.w3.org/2000/svg>
  4. The **XLink** namespace is: <http://www.w3.org/1999/xlink>
  5. The **XML** namespace is: <http://www.w3.org/XML/1998/namespace>
  6. The **XMLNS** namespace is: <http://www.w3.org/2000/xmlns/>

## HTML 5

**Legacy Code** - To help transition, the attribute if included has no effect on the page.

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

**Cache manifest** - You can include the URL of the document's cache manifest; but most beginners won’t need this

1. manifest = “url”

**Default Language** - You can optionally identify the default language with the global attribute, lang.

* 1. <html>
  2. <html lang="en" >

## Polyglot – TRY IT

So here are the current choices of what your web page code starts out with the code below.

Figure Starting Your Web Page

Type this in Notepad:

<!DOCTYPE html>

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

# Comments

Comments can be inserted anywhere in the page, but do this after the HTML tag and not after the closing HTML tag and can be used on one line or across multiple lines. Firefox is more rigid with requiring the correct syntax.

## XHTML

1. "<!--" and must be ended by "-->"
2. For XML content of comments must not contain two consecutive hyphens, or U+002D HYPHEN-MINUS (-)
3. Technically just include the space for browser support.

**<! - - Comments go here. - - >**

**<! - - ------------------------- HEADER ------------------------- - - >**

**<!-- Last Modified: 9/9/2011 -->**

1. **White space** does not make a difference in code view.
   1. So you have to use **&nbsp;** puts in a blank space if you want more than one blank space.

## HTML 5

1. Use comments of the "<!-- Comments to here -->" to be compliant with both standards. Make sure to add spaces as described above. Browser support may be an issue!

## PolyGlot

1. Use comments of the "<!-- Comments go here -->" to be compliant with both standards

# TRY IT!

Type this in Notepad:

Figure Sample Using Comments

<!DOCTYPE html>

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

<!-- AboutMe.html -->

HEAD and BODY Tags

Content in the **heading** such as title is not displayed in the web page in the browser.

1. The **<head>** tag is used as a container for other tags. They are root tags for the heading section. Title, meta, base, link, meta, script, style are examples of tags nested in the heading.
2. It does not have any attributes other than the general attributes.

Content in the **body** is displayed in the browser.

1. The **<body>** tag does have attributes, which will be described after the heading section child elements.

# META

The **meta** element stored information such as the page author, keywords used by search engines to classify your page, and the name of the editor used to create the page. Meta tags are different in HTML/XHTML versions

* 1. [How To Use HTML Meta Tags](http://searchenginewatch.com/showPage.html?page=2167931) By Danny Sullivan, Editor-In-Chief
  2. [HTTP Headers](http://vancouver-webpages.com/META/bycount.shtml)

TIP – Checkout a web page like <http://www.metatags.org/meta_name_no_email_collection> and view the HTML source code for the page. Look at the meta tags they used.

There are four main attributes to meta tags and they are charset, http-equiv, name and content.

## CharSet Attribute

This is used to identify the character encoding for the page. You can include many types of character sets such as <meta charset="ISO-8859-1"> but we use UTF-8 to be compatible with XHTML and HTML 5. You should include this in all of your web pages.

**<meta charset="UTF-8" />**

## Name and Content Attributes

User defined metadata are very important to understand because search engines will use this to classify your site. These are optional, but they have very important uses! You use both the name and content attributes together. The content might be thought of as the value. Many times Content Managed Systems (CMS) like WordPress allow the user to enter data, in the management tools, that is written to the meta tags. It helps keep track of page metadata as well as allows the web pages to be searched on these values.

Search engines to classify your web site use this metadata. Search engines like Google have rules that try to prevent developers from circumventing their search engines. Some web sites have long pages of keywords, just to try to attract people to the site for advertising revenue! Keep the number of characters short, like 200 characters, and make the content relevant! Search engine optimization (SEO) is all about making it easy for your customers to locate your web site. So, choose wisely! You should include this in all of your web pages.

**Description and Keywords Used By Search Engines**

**<meta name="description" content="HTML Tutorial" />**

**<meta name="keywords" content="HTML, XHTML, HTML 5" />**

**Author Information**

Sometimes you want to use author to represent an individual, position or department or even another company that create the web page or web site.

**<meta name="author" content="Web Development Team" />**

**<meta name="web\_author" content="Web Page Consulting Company">**

**Web Editor or Program**

Many times customers show me a web site. I can detect what web page editor created the web site by looking at the meta tag and locating the generator code. Notice that you can also see the id attribute used here, because it’s a general attribute. The **Content Management System** put in the id and the name was in upper case. It’s a user-defined value so it can be upper case or lower, but do not use spaces in the name! It’s like the name of a variable!

1. <meta name="**generator**" content="Dreamweaver" />
2. <meta name="generator" content="WordPress.com" />
3. <meta name="**GENERATOR**" content="**DotNetNuke**" id="**MetaGenerator**" /

**Publishing Data**

Web content sometimes needs to track the publishing information, including copyright and version data. Notice that you can use multiple values separated by commas. It is user defined so you can add your own values.

1. <meta name="**publisher**" content="My Company">
2. <meta name="**copyright**" content="2011">
3. <meta name="**revised**" content="Katie Kalata, 09/01/2011" />
4. <meta name="**revised**" content="09/01/2011" />

**Renew and Expire Content**

In some time-sensitive web sites, you do not want to have the web pages in the search engines for extended periods of time. You can help search engines and spiders by identifying when pages expire so they know to revisit the page. Expires will tell the search engine that the page can be removed from the database. Make sure to include a redirection on the page to another page, like your home page so users do not see outdated material or error messages like Error 404, File Not Found. Setting the content to 0 will delete the page once the search engine or spider sees it. Dates need to be in the right format, day, dd mm yyyy so the year in 4 digits which follows the **RFC1123** specifications. Adding the time is optional. Use the samples below as a template for how to format dates.

1. <meta **name**="**revisit-after**" content="3 days">
2. <meta name="**expires**" content="tue, 26 Sept 2011 14:35:00 **GMT**">
3. <meta name="**expires**" content="0">

Note that you can also expire content with **http-equiv** attributes too.

**<meta http-equiv="expires" content ="Sun, 26 Sep 2011 14:30:00 GMT">**

**No Collecting Email Addresses**

Some tags may be used to help inform others about your web site but the information does not need to be included in the web page. Programs may or may not use this information. For example, you may want to rate your web site or prevent users from accessing email in the web site. An initiative of **unspam.com** is to try to help forbid compliant robots from harvesting email addresses. It does not prevent harvesting yet but it does make the statement that they have been notified that it is not allowed. Michigan & Utah already climbed on board.

*“And if you have this statement on your website you are able to complain*

*about spam harvesting and spam runs. In the USA you can even start a lawsuit*”

Reference: <http://www.metatags.org/meta_name_no_email_collection>.

**<meta name="no-email-collection"  value="[link or terms]" />**

**<meta name="no-email-collection" content="http://www.metatags.info/nospamharvesting">**

**Web Site Content – Safety for Kids – FYI Only**

In the 1990s, there were several initiatives to help parents and schools identify safe content. In the 1990s the Communications Decency Act was struck down. However, there are many programs available today that help us lock down content for kids at home and school.

One method called Platform for Internet Content Selection (PICS) was a standard for labeling online content developed by the W3C. (See [www.w3.org/PICS/](http://www.w3.org/PICS/) and <http://www.weburbia.com/safe/index.shtml>) PICS is replaced by [Protocol for Web Description Resources](http://www.w3.org/2009/08/pics_superseded.html), (**POWDER**) which is used to describe resources using **Description Resource (DR)** documents which are XML compliant documents. The [POWDER Recommendations](http://www.w3.org/TR/powder-dr/) were updated last in 2009. [Powder Resources](http://www.w3.org/2007/powder/) are available, including a validation tool from W3. You can view a [sample DR document](http://www.w3.org/TR/2009/NOTE-powder-primer-20090901/#ICRA1), which contains the information about what is included and not included in the web site. Below is how the web page knows where to find the DR document. It is a much more complicated way to identify content safety but it is more flexible. An alternative is using a meta tag for “rating”.

* 1. <**link** rel="describedby" href="/**powder.xml**" type="application/powder+xml" title="ICRA labels" />

**Rating**

If you want to use a simpler method, you can use the meta tag to set a rating in the page. Possible values include: general, mature, restricted, 14 years, safe for kids

<meta name="**rating**" content="safe for kids">

**Refresh the Page**

This is used to refresh the current page. There are some side effects. You don’t know if the browser treats this as a refresh or page-reload. The user can click escape and prevent the redirection on a slower connection. This feature will **cause a failure** in the most recent [W3C's Web Content Accessibility Guidelines](http://www.w3.org/WAI/intro/wcag.php" \t "_blank). We will cover accessibility guidelines later in the lesson. However, this is often used in combination with a **static** hyperlink to a new page.

1. **<meta http-equiv="refresh" content="5"/>**
2. **<meta http-equiv=”refresh” content=”5;url= http://www. google.com” />**

**Location**

This is used to redirect the user to another page or to another web site. The user can click escape and prevent the redirection on a slower connection. Notice you can specify the number of seconds to wait before redirection.

**<meta http-equiv="location" content="URL=http://www. google.com" />**

**<meta http-equiv=”location” content=”5; http://www. google.com” />**

**Robots**

Robots and spiders crawl though each page on your site to gather data, which can be used in the searches. A good reference is <http://www.metatags.info/meta_name_robots> . **Robots and Spiders** visit your site to elicit content and determine if it should be included in their search engines. Robots may traverse this page but not index it. You can stop the indexing on individual pages. You can combine the values here.

* 1. The basic syntax is:

< meta name ="**robots**"   content ="**all** | none | index | **noindex** | follow | **nofollow**">

* + - 1. ALL – default, allow all of the files to be indexed and NONE not to index any files, & not to follow the hyperlinks
      2. INDEX page may index and NOINDEX prevents anything indexing
      3. FOLLOW free to follow the links and NOFOLLOW not to follow the links
      4. **NOIMAGEINDEX** prevents the images from being indexed
      5. **NOIMAGECLICK** prevents the use of **links directly to the images**, instead there will **only be a link to the page**
  1. Sample – Index the page and follow through to the other pages on the site.

**<meta name="robots" content="index, follow">**

* 1. You can stop them from indexing this page or following the rest of the pages on the web site.

**<meta name="robots" content="noindex, nofollow, noimageindex">**

## HTTP-EQUIV

HTTP-EQUIV attributes are the equivalent of setting HTTP headers. You cannot simply choose your own value for the http-equiv attribute, because this information needs to be compatible with the HTTP headers. [W3Schools.com](http://www.w3schools.com/html5/att_meta_http-equiv.asp) has a summary of the attribute values that are supported in HTML 5.

**Content Language –** Can be used by search engines to classify your site.

**<meta http-equiv="content-language" content="en-US" />**

**Content-Type**

Identifies the type of content in the page, the MIME type. Usually you add the character set with the Content Type.

<meta http-equiv="**content-type**" content="text/html; charset=windows-1252">

<meta http-equiv="content-type" content="text/html; charset=ISO-2022-JP">

<meta http-equiv="content-type" content="text/html; charset=UTF-8">

**Creating a Cookie**

Cookies are small text files stored on the client’s system. You can retrieve the cookie using JavaScript and using server side programs like ASP. As a web developer you should know how to locate your cookie settings in your browser. There are so many browsers today, and the versions change. So, you should take time to explore your browser settings! Usually there is a Tools menu and Internet Options or Settings. Sometimes the cookie settings are in the privacy or security settings.

Cookies are text files and viewable on the client. You can see the userid is 2. Though encoded, this is text and not secure!

You can specify the path as a URL or use / to represent the current path.

Figure Viewing a Cookie

**<meta http-equiv=" set-cookie " content="MyCookie=MyValue; expires= Sun, 26 Sep 2011 12:00:00 GMT; path=/">**

<meta http-equiv="**set-cookie**" content="**MyCookie**=MyValue;  
**expires**=Sun, 26 Sep 2011 12:00:00 GMT; path=http://www.google.com">

**Other MetaTags – FYI Only**

There are many other meta tags that are useful. Another new trend is the he [Dublin Core](http://dublincore.org/" \t "_blank) **Metadata Initiative.**

Some of them are based on a specific browser. In Internet Explorer 4, Microsoft added a transition effect when you entered and exited the site (**Page-Enter**, Page-Exit, Site-Enter, Site-Exit), similar to PowerPoint presentations. But this only works in Internet Explorer browsers. Another reference on this is [Transitions Between Pages](http://www.ruleweb.com/dhtml/html/trans.html) (Ruleweb) Caution should be used when using browser dependent features that are not supported by the standards.

<meta http-equiv="**Page-Enter**“ content="revealTrans(Duration=3.0,Transition=2)">

# TY IT!

Type this in Notepad:

Figure Add in the Meta Tag Data

<!DOCTYPE html>

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

<!-- AboutMe.html -->

<head>

<meta charset="UTF-8" />

<meta name="**description**" content="HTML Tutorial" />

<meta name="**keywords**" content="HTML, XHTML, HTML 5" />

<meta name="**author**" content="Katie Kalata" />

<meta name="**revised**" content="09/01/2011" />

</head>

## Title

You should use the title in EVERY web page you create. The title is used in toolbar in browser, in favorites, search engine results, so you need it for marketing your web site. Some developers advocate having a different title on each page describing the page. But you should still include the company name on the page!

**<title>My Company</title>**

## Link

The **link** element links to external resources such as style sheets and JavaScripts. We will cover this in later lessons. Let’s look at linking a **style sheet** a **javascript** file. You just identify the MIME **type** with type, and **href** is the location where the file is located. The **rel** is the relationship of the file. Other values for rel include **author** (links to the author of the document, **license** (links to copyright information), **sidebar** (link to a document to display in the sidbar), **help** (link to a help document) and archives, author, bookmark, external, first, index, last, nofollow, noreferrer, pingback, search, tag, up. So many kinds of data can be linked to a web page!!! Here are examples.

1. <link **type**="**text/css**" **href**="MyStyles.css" rel="stylesheet">
2. <link type="**text/javascript**" href="MyScript.js">
3. <link type="image/gif" rel="**icon**" **sizes** = "heightxwidth" *>*
4. <link rel="alternate" type="**application/rss+xml**" **title**="Company RSS" href="http://news. company.com/news.xml">

## Image Icons

For now, you can use the link tag to link graphic elements, which can be used when the user saves the web page.

1. <link **rel**="**shortcut icon**" type="**image/png** " **href**="**favicon.ico**">
2. <link rel="**icon**" type="**image/png**" **href**="http://www.google.com/favicon.png">
3. <link **rel**="**shortcut icon**" type="**image/x-icon**" **href**="/**favicon.ico**">
4. <link rel="icon" type="image/x-icon" href="http://www.google.com/favicon.gif">
5. <link rel="shortcut icon" href="favicon.ico"/>

<!DOCTYPE html>

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

<!-- AboutMe.html -->

<head>

<meta charset="UTF-8" />

<meta name="description" content="HTML Tutorial" />

<meta name="keywords" content="HTML, XHTML, HTML 5" />

<meta name="author" content="Katie Kalata" />

<meta name="revised" content="09/01/2011" />

<link rel="icon" type="image/png" href=" favicon.png">

</head>

<body>

</body>

</html>

# TY IT!

Type this in Notepad, save it as **HTMLSample2.html**.

Make sure you have a **16x16** graphic icon and change the name and image type if needed.

Figure Using Meta Tags and Fav Icons

## Create Your Own Fav Icon

1. Go to <http://ie.microsoft.com/testdrive/browser/iconeditor/default.html>.
2. Click on **Open X-icon Editor** button, which takes you to <http://ie.microsoft.com/testdrive/browser/iconeditor/default.html> where you can use a browser HTML5 application called **X-Icon Editor** based on **Canvas**.
3. Click the **Import** button.
4. Click the **Upload** button. Locate an image on your hard drive. Reposition the image as desired.
5. Preview the image. Click **Ok**.
6. Click **Preview** to review your icon in the browser.
7. Select **16 x 16** image and click **Export**.
8. Click **Export** your icon button.
9. You will be downloading a **binary** file named **favicon.ico**.
10. Place that file after it is downloaded into the same folder as your web page or on the desktop, where you can save your web page.

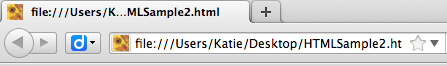


Figure Viewing the Favicon in the Browser

# Other Heading Section Tags

There are other tags, but we will be discussing them later in the course.

**Base**

1. Sets the base location for all links in the page. Blank opens in a new window. / means off the root of the web site. Otherwise, the links are relative to the page.
   1. <**base** href=”pages/” target=”\_blank”>
   2. <**base** href=”/pages” target=”\_blank”>

**Script**

1. Specify MIME **type** and an absolute or relative address in the **src**, source attribute.
2. **Async** is to execute asynchronously as opposed to **simultaneously**, **synchronous**. Used with AJAX.

<**script** type=”**text**/**javascript**“ **src**="/scripts/**MyFunctions.js**" **async** = “async” >

**Style – FYI Only**

1. Style sheet rules, that help design the layout and the presentation, can be assigned by the type of media including **braille, projection, screen, handheld, all, tv, tty, aural** by just using the media attribute.

**media**="**screen** and (device-width:500px)"

media="**print** and (resolution:300dpi)"

1. **Scoped** applies style to **parent and child** elements, but **not entire document**.

<**style** type="text/css" **media**="**print**" **scoped**="scoped" >

# THE BODY ELEMENTS

Finally, here we are at the body elements and attributes! These are the elements, the text and content that show up in the browser, in the web page itself!

# BODY

## HTML 4.01 and XHTML Transitional 1.0

In the past version of Transitional XHTML 1.0 and HTML 4.0 you could use many deprecated attributes like bgcolor and background to set the background image or color for the entire page. You could set the link default colors with link, vlink and alink. There were even events that could be configured such as **onload** for when the page loads. (Refer to [W3Schools](http://www.w3schools.com/tags/tag_body.asp) web site for more information.)

## HTML 5

Simple, there are no attributes other than the general ones mentioned earlier, such as ID, title, and style. But there are events that can be called. [W3Schools](http://www.w3schools.com/html5/html5_ref_eventattributes.asp) has a great list of global event attributes.

**<body>**

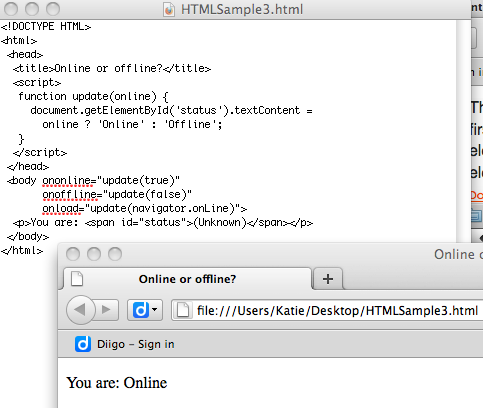
**Events**

Figure Using Event Attributes with the Body Tag

Here is an example of how you can use the <body> events to add functionality to your web site.

You will learn later how the code is written. But for now, simply know that these events are attributes and can call JavaScript functions when the event occurs.

So, don’t worry about these now if you see them in web pages, but later in the course you’ll learn about them.

## Events – Reference Material

These are the events associated with the <body> tag as attributes. You might want to keep this handy when we talk about JavaScript later in the course.

1. onabort
2. onblur\* onfocus\*
3. oncanplay, oncanplaythrough
4. onchange, oncuechange, ondurationchange
5. onclick, ondblclick
6. oncontextmenu
7. ondrag, ondragstart, ondragend, ondragenter, ondragleave, ondragover
8. ondrop
9. onemptied
10. onended
11. onerror\*
12. oninput
13. oninvalid
14. onkeydown, onkeypress, onkeyup
15. onload\* , onloadstart
16. onloadeddata, onloadedmetadata
17. onmousedown, onmousemove, onmouseou, onmouseover, onmouseup, onmousewheel
18. onpause, onplay, onplaying
19. onprogress
20. onratechange, onreadystatechange
21. onreset, onsubmit
22. onscroll\*
23. onseeked
24. onseeking
25. onselect
26. onshow
27. onstalled
28. onsuspend
29. ontimeupdate
30. onvolumechange
31. onwaiting
32. Onafterprint, onbeforeprint
33. onbeforeunload
34. onblur, onfocus
35. onerror
36. onhashchange
37. onload, onunload
38. onmessage
39. ononline, onoffline
40. onpagehide, onpageshow
41. onpopstate
42. onredo
43. onresize
44. onscroll
45. onstorage
46. onundo

Reference: <http://www.w3.org/TR/html5-author/global-attributes.html#global-attributes>

You don’t need to memorize these, but again, it’s a nice list to have as you develop your web sites. The asterisk is only to show attributes which when used with the body, refer to the document as a whole, the window, and not the body per se.

# Header and Footer Content Elements

Some elements are **deprecated**, which means they **aren’t supported in new versions**. Browsers still support many of the elements. So, you have to ‘test’ for them or look for polyglot methods.

## HTML 5 Only

Header and footer are **new** HTML 5 elements are simply **logical** ways to **organize content** in the code.

**<header>Put the banner image here and logo</header>**

**<footer>Address and text links are often in the footer</footer>**

# Heading Elements - H1 to H6

You have all used styles in Word. There are many tags that are used just like styles. The heading tags format all the content and insert a line break after. There are 6 headings with 1 the largest size font. Some elements are **deprecated**, which means they aren’t supported in new versions. Browsers still support many of the elements.

## XHTML 2.0

There is a new element, h, which can be used to nest headings. But this was not included in HTML5.

## HTML 5

**HTML 5** there are no specific attributes. The align attribute in HTML 4.01 is deprecated. Style and formatting is done using style sheets. The presentation and structure are now separated.

**<h1>This is heading 1 the largest</h1>** <h2>This is heading 2</h2>  
<h3>This is heading 3</h3> <h4>This is heading 4</h4>  
<h5>This is heading 5</h5> **<h6>This is heading 6 the smallest</h6>**

# Heading Groups - HGROUP

HGroup is a new tag in HTML 5, that allows you to group headings. You can think of them as similar to section or container, but it’s more than that. The highest element is considered the heading for that hgroup section. In the first example, Fish is the heading for the section, and Rabbits is the heading in the second, even though it appears after Cats. This isn’t about the order presented in the browser, but the order which other programs we might write, sees the listing of the headings.

**<hgroup> <h1>Fish</h1> <h2>Dogs</h2> </hgroup>**

**<hgroup> <h3>Cats</h3> <h2>Rabbits</h2> </hgroup>**

# Containers

The purpose of the Paragraph, span and div tags is to **contain** other content and tags.

# Paragraph - P

An error would be no starting P tag and you just start typing content. That means the content is nested directly in the body tag, and this is not proper web programming. However, you can see that no nesting the content in the P tag can alter the layout of the page. Line breaks moves the cursor to the next line, but does not include the line feed.

It’s important to understand the XHTML rules and proper web programming techniques. As you can see, the code is the same no matter how it’s spaced.

**<p>Once upon a time</p> <p>there was a girl named Dorothy. </p>**

Can also use **<p />** for blank line or **<p>&nbsp;</p>**

# Line Break - BR

The line break tag, br, is a VOID tag so it does not use a closing tag. It is not a container tag but is included here because it’s often used within a paragraph to insert a line break. If you closed the P tag, the space between lines would be bigger than if you inserted a line break tag. There is less white space with line breaks than p tags.

**<p>Put the text<br />on the next line.</p>**

# DIV

Div is a **block element** and is used as a container element. The DIV element showed up with HTML 4.0 and has been critical for integrating JavaScript with HTML to create animated web sites. DIV will continue to be supported in XHTML 2.0. You will learn more about DIV when we talk about animated web pages.

**<div>Put some content here</div>**

# SPAN

Span is an **inline** element and is used as a container element. Span is like the DIV tag, but **does** **not append a return** character to the content. Why? Well you can use the SPAN later with style information to format just the word content. But, we don’t put the format information here, because it’s more efficient to store it in a style sheet. Style and formatting is done using style sheets. Notice you can use general attributes such as **lang** for identifying the language.

**<p>Put <span>content</span> here</p>**

**<p><span lang="fr">Bonjour</span> Thomas!</p>**

At this point, it’s hard to specify exactly what to code, because there are so many ways to organize content in the page. So, this is merely an example. As you learn new tags, you’ll come up with content and how you want to use the tags to layout and format your content. For now, until we add styles, the content will flow, linearly, going line by line through the page.

# TY IT!

<!DOCTYPE html>

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

<!-- AboutMe.html -->

<head>

<meta charset="UTF-8" />

<meta name="description" content="HTML Tutorial" />

<meta name="keywords" content="HTML, XHTML, HTML 5" />

<meta name="author" content="Katie Kalata" />

<meta name="revised" content="09/01/2011" />

<link rel="icon" type="image/png" href=" favicon.png">

</head>

<body>

**<header>Put the banner image here and logo</header>**

**<hgroup>**

**<h1>Fish</h1>**

**<h2>Dogs</h2>**

**</hgroup>**

**<hgroup>**

**<h3>Cats</h3>**

**<h2>Rabbits</h2>**

**</hgroup>**

**<p>Once upon a time</p>**

**<p>there was a girl named Dorothy. </p?**

**<p><span lang="fr">Bonjour</span> Class!</p>**

**<div>Put some content here</div>**

**<footer>Address and text links are often in the footer</footer>**

</body>

</html>

Type this in Notepad, save it as **HTMLSample3.html**.

Make sure you have a**16x16** graphic icon and change the name and image type if needed.

# Sectioning content – New to HTML 5

**Problem**: These don’t work well with all or many browsers. Check out samples on the **HTML Gallery** in your browser. Note that one way is to use the tags, and use styles to design them. <http://html5gallery.com/>

# Article

This tag is **new** in html 5. The <article> tag specifies independent, self-contained content. An article should make sense on it’s own and it should be possible to distribute it independently from the rest of the site. Useful for **RSS feeds** on other web sites! You can nest section within a section, or section within an article. But don’t nest an article within an article.

**Examples of possible articles:**

forum post newspaper article

blog entry user comment

**Sample**

**<article>**

<a href="http://blog.netscape.com/2007/12/28/  
end-of-support-for-netscape-web-browsers">

Netscape is dead</a><br />  
“AOL has a long history on the Internet, being one of  
the first companies to really get people online . . .” (**more**)

**</article>**

# SECTION

This tag is **new** in html 5. Section is a container element. So, you can have multiple sections, with different nesting, without having to use a numeric value. This provides more flexibility in the long run. Nesting elements in your code will make it easier for you to differentiate the content. You can see that the logical groupings have no affect on the presentation. Normally the section has headings, title, and footer. Again the section is somewhat self-contained separate from the web page. You CAN nest article inside sections, but again browsers may not support these.

## HTML 5 Only

<**section**>  
  
  <h1>Animals</h1>  
  <p>The local zoo has a new baby gorilla!</p>  
  
</**section**>

# ASIDE

This tag is **new** in html 5. The <aside> tag defines some content aside from the content it is placed in.

The aside content should be *related to the surrounding content*. The <aside> content could be placed as a **sidebar** in an article. Examples are parenthetical remarks, comments, annotated references, anything you want to add to the discussion without actually putting it in the article.

**Sample**

<p>My family and I visited Denver this summer.</p>

**<aside>**

<h4>Denver</h4>  
<p>Denver has a fabulous fabric store!</p>

**</aside>**

# NAV

This tag is **new** in html 5. The <nav> tag defines a section of navigation links. If you have "previous" and "next" buttons in your document, they should be placed inside the <nav> element. You’ll want to use target or they leave the web site! This is useful to insert into footers too. Note the | vertical pipe used to separate links.

**Sample**

<**nav**>

<a href="Page1.html">Previous</a> |

<a href="default.html">Home</a> |

<a href="Page2.html">Next</a>

</**nav**>

# Details and Summary

The <details> tag specifies additional details or controls, which can be **hidden** or shown **on demand.**

The **content** of the <details> tag should **not be visible** unless the **open** attribute is set. The open attribute specifies that the details should be **shown**.

If this attribute is present the details will be shown. Use it together with the [<summary>](http://www.w3schools.com/html5/tag_summary.asp) tag to make your own header for the details. The header is visible, and could show the details when the user clicks on the header. The <summary> tag contains a header for the [details](http://www.w3schools.com/html5/tag_details.asp) element, which is used to describe details about a document, or parts of a document. **Tip:** Use it together with the <details> tag to make your own header for the details. The header is visible, and could show the details when the user clicks on the header.

**Note:** The "summary" element should be the first child element of the "details" element.  
**New supported only in Chrome!!**

**<details open="open" >**

**<summary>**Copyright 1999-2011.**</summary>**<p>Contact us for more information.</p>

**</details>**

# TRY IT!

Figure Using Meta Tags and Fav Icons

<!DOCTYPE html>

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

<!-- Micahelangelo.html -->

<head>

<meta charset="UTF-8" />

<meta name="description" content="HTML Tutorial" />

<meta name="keywords" content="HTML, XHTML, HTML 5" />

<meta name="author" content="Katie Kalata" />

<link rel="icon" type="image/png" href=" favicon.png">

</head>

<body>

<header>Michelangelo</header>

<section>

<h1>Title: The Life of Michelangelo Buonarroti</h1>

</section>

<div><strong>Author:</strong> John Addington Symonds</div>

<h2>CHAPTER I</h2>

<article>

<a href="http://www.gutenberg.org/cache/epub/11242/pg11242.html">  
Project Gutenberg</a><br />“An article on Micahelangelo” (more)

</article>

<aside>

<h4>Planetary Predictions</h4>

<p>Vasari tells us that ….</p>

</aside>

<section>

The Buonarroti Simoni, to whom Michelangelo belonged, were a Florentine family of ancient burgher nobility...

</section>  
  
<section>

On the 6th of March 1475….

</section>

<section>

<details open="closed">

<summary>Thank you for visiting!</summary>

</details>

</section>

<footer>

<nav>

<a href="Page1.html">Previous</a> |

<a href="default.html">Home</a> | <a href="Page2.html">Next</a>

</nav>

</footer>

</body>

</html>

Type this in Notepad, save it as **HTMLSample4.html**.

# 

# Blockquotes

Mostly used for indenting content and quotations (because the text is formatted as italic on some browsers) You can nest multiple blockquotes to push the indent further into the center of the page. The indentation and paragraphs are relative to the rest of the page. Mostly used for indenting content and quotations, because the text is formatted as italic on some browsers. Don’t nest other container tags like article or section within a blockquote.

You can nest multiple blockquotes to push the indent further into the center of the page.

* + - 1. < **blockquote**>You can insert other tags nested within <em>blockquote</em> tags**</ blockquote>**

# Text Level Elements

This is a review of most of the HTML 5 elements supported that involve text. You can [view examples](http://www.w3.org/TR/html5/text-level-semantics.html#usage-summary) from the W3 standards.

# Bold, Strong, Italic and Emphasis – B, I

The bold <**b**> and italic <**i**> tags are used to format the content as bold and italic. However, these are fixed as bold and italic. Though **deprecated**, they continue to be supported. Initially these tags were changed to <strong> and <em> tags. The <em> tag is for **emphasis** and will display the text as *italic* usually. That way the browser was left to determine what to do to present the content. Now, <i> is supported with a slightly different meaning.

## HTML 5

1. **<strong>** may make the text bold, but it’s meaning is to identify the text as **important**, not just bold.
2. **<i>** is used for **alternative voice**, not just italicizing text and <b> for indicating it was a **keyword**.
3. **<em>** is for stress **emphasis**.

## Examples

<**strong**>Strong text</**strong**>

<**em**>Emphasized text</**em**>

<p>My favorite place to visit is: <b>Ireland</b></p>

<p>My favorite place to visit is: <strong>Ireland</strong></p>

<p>My favorite place to visit is: <i>Michigan</i></p>

<p>My favorite place to visit is: <em>Michigan</em></p>

# Address

The Address element has been around for a while. It formats the content generally as *italic*. It’s used often to display contact information.

**<address href="mailto:me@me.com">Me!</address>**

# Preformatted Content - PRE

In the early versions, simple graphics were created using the pre element. The white space is considered part of the content and is preserved so you don’t need the &nbsp;

**<pre> \* \* \* \* \* </pre>**

Can you make a pretty picture with characters?

# Annotations/Underline - U

Used to underline text but in HTML 5 it is used for annotations. It can be confused with the hyperlink so it’s not recommended. Although some sites like W3Schools list this as deprecated, the W3 standards of Aug 2011 still contain reference to the element

Figure Previewing the Pre Tag in a Browser

**<u>Underline</u>**

# Citation - CITE

Defines a citation.

**<cite>Citation</cite>**

# Quotation – Q & CITE

The q tag is used to create a short quotation, which the browsers will place quotation marks around.

<p>The doctor said **<q cite="http://www.google.com"> he’s a hopeless nerd!</q>** </p>

# Small

The <big> and <small> are used to set the font size, relative to the document default font size. Although <big> is not supported in HTML 5, <small> is supported and will make the text size smaller. Small is often used to include a **side comment.**

<p>This text contains **<small>very small</ small >** text.</p>

# Subscript and Superscript – SUB and SUP

The <sub> and <sup> tags provide subscript and superscripting.

<p>This text contains **<sub>subscript</sub>** text.</p>

<p>This text contains **<sup>superscript</sup>** text.</p>

# Blockcode, Computer Code, Computer Output and Keyboard Tags

More and more applications need to display code to the end user, who may be a web programmer or system administrator. There are many options. Style sheets are ideal, but you can still use some of the tags available.

**<blockcode>Type this code</blockcode>**

Code will insert a monospace font – so that all of the individual characters are the same size. Code element is similar to blockcode, but is not a block element. It simply formats the contents using a code font, such as Courier. It was useful before style sheets.

Figure Using the Blockcode Tag

1. <**code**>Computer code text</code>
2. <**samp**>Sample computer code text</samp>
3. <**kbd**>Keyboard text</kbd>
4. <**var**>Variable</var>

# Highlighting - MARK

Mark is useful to highlight text. It puts by default in most browsers, yellow highlighter over the text area.

<p>The homework is due **<mark>Sunday</mark>** at noon **<mark>CST</mark>** .</p>

# Delete and Insert – DEL, INS

Some sites have content that changes often and you might want to show the difference. Ins will insert text and del will delete it. You need to identify the datetime to show when the text was changed (inserted/deleted) and use the cite attribute to identify a URL where the user can read why the text was modified.

<p>The current topic is **<del>fish</del> <ins>dogs</ins>** !</p>

<p>The current topic is **<del>turtles</del> <ins>whales</ins>** !</p>

# Inaccurate Text - S

S is a good example of how the standards redefine tags. In the earlier versions, S was used to define strikethrough text and was later deprecated. In HTML 5, the S tag is used to define text that is no longer correct, current or relevant. Again this can be useful with publishing and versioning of content.

<p>**<s>You failed the course.</s>** </p>

<p>Sorry, you really passed! Congrats! </p>

# Line Breaking Opportunity - WBR

This is a new tag in HTML 5 so it’s **not** supported in Firefox, Chrome and Safari but **not** Opera or Internet Explorer yet. The **<wbr>** tag identifies where a Microsoft Word would be ok to add a line-break. This is useful if you are using a very long word and don’t want the word to break in certain places.

I would recommend using this to break URLs or words that could be offensive if not broken in the right place or major words like a company name.

<p>http://www.w3.org/<wbr>TR/html5**/<wbr>**text-level-semantics.html#usage-summary</p>

<p>J.P. Morgan**<wbr>**Chase Bank</p>

# Abbreviation - ABBR

Abbr tag allows you to define an abbreviation or acronym and classify it for browsers, spell checkers, translation systems and search-engines. The title is used for showing the value when the user places their mouse over the term.

The **<abbr title="Association for Computing Machinery">ACM</abbr>** is a major computer professional organization that publishes research and provides help for computer clubs at universities.

# Defining Instance - DFN

You can define a term using dfn.

**<dfn>Definition term</dfn>**

# Horizontal Lines - HR

Was **not** in **XHTML** **strict**.

The horizontal rule element **was** deprecated. It is useful for dividing areas within a page. It is not a container tag, so you must close the opening tag. <hr />. In the past the hr tag simply inserted a line, for visual changes in the page. Today in **HTML 5**, the tag now only inserts the line, but also **indicates that there is a change in the content.** This can be used to indicate a different content.

In the past, we used attributes to style the line such as align, noshade, size, and width, and some of these were not supported in Netscape. Today, we configure the style using style sheets, not attributes. Like other tags, attributes other than the general attributes are not allowed.

<p>This is about fish.</p>

**<hr />**

<p>This is about turtles.</p>

# Bidirectional Order – BDO and BDI – Not common with US Sites!

Normally in western languages, text is read and written from **left to right**. But that is not the case in many other languages.

## BDO – Bidirectional Order

BDO allows you to change the bidirectional order to **right to left**. There is one attribute, **dir** indicates a right to left (**rtl**) or left to right (**ltr**) direction. Content in the tags is treated like a paragraph. This capability can also be configured in stylesheets using the **unicode-bidi** property.

**<bdo dir="rtl">Read this from right-to-left.</bdo>**

## BDI – Bidirectional

This tag is **new** in HTML 5. BDI is used for identifying parts of text. This is useful if you combine text content with numerical lists where the list appears on the left. **Lang** is the general attribute for language and would be likely used here if you were mixing languages in a document. For example, **en** is **English**, and fr, French.

**<p>Welcome <bdi lang="fr" dir="rtl">George</bdi>**, you have 2 messages. </p>

You want this to render as follows:

DNAW CIGAM — left! A priceless gift!

In the past you could try to use **dir** to change direction using **span** tags. But notice what happens!

**<span dir="rtl">MAGIC WAND</span>** — 4 left! A priceless gift!

4 — MAGIC WAND left! A priceless gift!

But because **Unicode** bidirectional algorithm (bdi) rules, the browser would override this and display the wrong content. (<http://unicode.org/reports/tr9/> for more information on BDI)

4 — DNAW CIGAM left! A priceless gift!

So, the bdi tag will isolate the content and only change the reverse order of that content. However, since this is new, not all browsers support this yet.

<span> **<bdi dir="rtl">MAGIC WAND</bdi>** </span> — 4 left! A priceless gift!

Sorry, but this doesn’t work too well either in the browsers. You get the same effect as earlier!

Again, deprecated elements, new elements all make it a challenge to program!

# Hyperlink - A

The A tag, is known as the **Anchor** element or **hyperlink** tag. It’s used to make links to other pages, resources, and even to internal targets within the same page. It’s the main reason why we were able to use the WWW – to locate resources quickly. The URL can be relative, within the document (called anchors) or to another web page or other resource.

Visit my **<a href="drinks.html">drinks</a>** page.

* + 1. **<a href=“URL”>Content Displayed</a>**

The content that can be displayed can be text, images, or other valid HTML.

The href can be used to identify **relative** and **absolute** resources, web pages and graphics. For relative pages, the / represents the root of the web site. You can also specify directory paths.

**Relative URL**

**<a href = "trains.html"> Information about Antique Trains </a>**

**<a href = "/trains.html"> Information about Antique Trains </a>**

**<a href = "pages/trains.html"> Information about Antique Trains </a>**

**Absolute URL**

**<a href = “http://www.company.com/trains.html"> Information about Antique Trains </a>**

**Link to Images**

The href can be used to identify relative and absolute resources, web pages and graphics. Link to an image, display text. **Target = “\_new”** can open it up in a new window.

**<a href = "train100.jpg" target = “\_new”> Pictures of Antique Trains </a>**

**Thumbnails**

Link to an image, display an image &/or text. The second one the text is also included as a link. But the third, it is not a link.

**<a href = "train100.jpg"> <img src="train100.jpg"> </a>**

**<a href = "train100.jpg"> <img src="train100.jpg">Click the thumbnail </a>**

**<a href = "train100.jpg"> <img src="train100.jpg"> </a> Click the thumbnail**

* 1. **Link to Word, PDF, other documents**
  2. You may want to create links to other resources.

<a href = "train100.**pdf**"> Open a word document about trains. </a>

<a href = "train100.**doc**"> Open a word document about trains. </a>

* 1. **E-mail Links**
     1. A link to your email address uses the **mailto** protocol instead of http.
     2. This link will open the users email program and insert the email address for them.

**<a href=“mailto:kkalata@lssu.edu”> Katie Kalata</a>**

## Named Hyperlink – Internal Links/Anchors

The hypertext tag/module will include the capability to be a link or anchor. The A tag can link to **internal targets** within the same page or internal targets in other pages. In other words, instead of linking to the top of the page, you can ‘go’ to a direct place on the web page. Each object can have an ID and you link to the ID attribute directly.There are 2 methods. Same output.

**The old method - Deprecated**

If you use the name attribute, it needs to point to another elements ID property, not name. It is preferred to use ID over name attributes.

<a name=”top”>Top

1. Create a bookmark where you wanted the target. (Notice this is wrong in the code shown!)

**<a name = “M”>**

**<a name = “top”>**

2. Then, in the hyperlink you would append the target name to the page name with a #

**<a href = “#M”>**

**<a href = “mypage.htm#top”>**

**<a href = “http://www.comp.com/trains.html#train52”>**

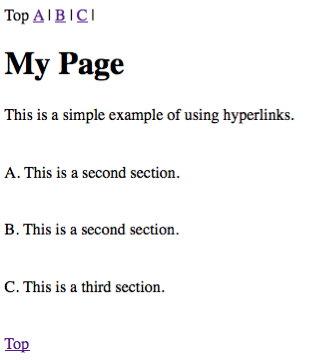
**New way:** Use **id** instead of **name** attributes

**<a id = “M”>**

**<a id = “top”>**

**Sample – the key is you need a lot of content.**

<a name="top" id="top">Top</a>

**<a href="#a">A</a> | <a href="#b">B</a> | <a href="#c">C</a> |**

<h1>My Page</h1>

<p>This is a simple example of using hyperlinks. </p>

<br /><br /><br /><br /><br /><br />

<a name="a" id="a">**A.** </a>This is a second section.

<br /><br /><br /><br /><br /><br />

<a name="b" id="b">**B**. </a>This is a second section.

<br /><br /><br /><br /><br /><br />

<a name="c" id="c">**C.** </a>This is a third section.

<br /><br /><br />

<br /><br /><br />

<a href="#top">**Top**</a>

<br /><br /><br /><br /><br /><br />

<br /><br /><br /><br /><br /><br />

# Image Format

1. GIF (Graphic Interchange Format)
2. JPEG (Joint Photographic Experts Group)
3. Portable Network Graphics (PNG)

**GIF (Graphic Interchange Format)**

1. Indexed color
2. A custom palette is devised  for each image using 256 colors
3. **8-bit** color (**256** different colors)
4. Often used in clip art and logos
5. Uses LZW **compression** - compresses  large blocks of the same color  reducing size of file considerably
6. The image is on a **transparent** layer – so you can see through to the background.

**JPEG (Joint Photographic Experts Group)**

1. **24-bit** color (16 million different colors) (photos)
2. Both GIF and JPG use **compression**, but JPEG compression is better
3. **Lossy** compression (Lose some information/quality each time you apply compression)
4. Compresses by combining several subtle shades into 1 shade
5. **Fake Transparency**
6. Make **1 color** in the background the **same as your background**

**Portable Network Graphics (PNG)**

1. Better than GIF; has been around since 1995; but not widely used until recently
2. Because of compression, transparency and multiple color support, it is often used today

# Image –IMG

The <img> tag defines an image in an HTML page. Notice that the text will and image will **scroll** to the right when you enlarge the browser window. Your book covers some information on image types.

Notice that images are not technically inserted into an HTML page, images are linked to HTML pages. The <img> tag creates a **holding space** for the referenced image.

The <img> tag has two required attributes: src and alt.

**<img src="pink.gif" alt="Pink Color Marker" />**

**Source attribute (src)**

<img /> tag has 30 different possible attributes. Key – you must have this image!

Relative to the current page and in the same directory

**<img src = "train.jpg" />**

If stored in an **images** folder off the root of the web site. The / indicates the **root** folder

**<img src = “/images/train.jpg" />**

**Relative** using DOS/File path commands ( . . )

* + 1. **<img src = “../images/train.jpg" />**

**Absolute** – provide a fully qualified URL. Use if images are stored on a separate web site or location.

* + 1. **<img src="http://company.com/images/train.gif" />**

**Other Attributes**

* 1. **src**=“” Relative URL or absolute URL
  2. **alt**=“” Required by XHTML 1.1 & for non-graphical browsers & **accessibility requirements**
  3. **width** and **height**=n

Scale image by changing width and height, but it will still take the same time to load – it **doesn’t change the original graphic**; in **pixels** by default; file size impacts download time

* 1. **border** = n
  2. **hspace**=n **vspace**=n **Whitespace padding frame**; in pixels by default
  3. **align**=“” Position of graphic on page – floats left or right

**Style = “attribute:value; attribute2:value2;”**

**So, you can change to use this:**

**<img src = “/images/train.jpg" style=”height:100px; width:50px; border:0px;float:right;/>**

## XHTML

The image element <img> is used to insert images into your web page. This was going to be replaced with the Image Module in XHTML 2.0, but this isn’t the case anymore.

# MAP

A clickable map is an image-map is an image with clickable areas. The shapes of the **clickable areas** can vary with basic circle, rectangle, and polygons. Image Maps allow you to click on part of an image and go to a corresponding link. The map element contains a number of [area](http://www.w3schools.com/html5/tag_area.asp) elements and defines the clickable areas in the image map.

In HTML5, the **id** attribute of the <map> tag is also identified and must have the same value as the name attribute.

**Identify the Map in the Image Code - USEMAP**

First, create an image such as a map of counties, states, countries or planets. You could use plates with food on a table for a restaurant; parts of a computer, artwork on a museum wall, or anything that has parts of an image you want people to click on. Identify the map with a hash character ("#") which is the pound sign, plus the name or id of the map element.

**<img src="table.gif" width="145" height="126" alt="Restaurant" usemap="#table" />**

**Properties**

Deprecated properties include align, border, hspace, and vspace. Active properties include:

**alt** text Specifies an alternate text for an image

**src** URL Specifies the URL of an image Possible values:

An absolute URL - points to another web site (like src="http://www.company.com/image.gif")

A relative URL - points to a file within a web site (like src="image.gif")

**height** pixels **%**  Specifies the height of an image

**width** pixels % Specifies the width of an image

**usemap** #mapname Specifies an image as a client-side image-map

he usemap attribute specifies an image as a client-side image-map.

The usemap attribute is associated with a [map](http://www.w3schools.com/html5/tag_map.asp) element's name or id attribute, and creates a relationship between the image and the map.

**ismap** ismap Specifies an image as a **server-side** image-map The ismap attribute specifies an image as a server-side image-map. An image-map is an image with clickable areas. When clicking a sever-side image map, the click coordinates are sent to the server as a URL query string. Not commonly used today.

<a href="mytable.html">

<img src="foodtable.gif" **ismap**="ismap" />  
</a>

# Area

The <area> tag defines an area inside an image-map (an image-map is an image with clickable areas). The area element is always nested inside a <map> tag. Note: The usemap attribute in the <img> tag is associated with the map element's **name** attribute, and creates a relationship between the image and the map.

<img src ="planets.gif" width="150" height="100" alt="Planets" **usemap ="#planetmap"** />  
  
<map name="planetmap">  
  <area **shape**="rect" **coords**="0,0,100,150" href="mytable.html#winelist" alt="Wine List" />  
  <area **shape**="circle" **coords**="100,50,3" href="mytable.html#hours" alt="Hours" />  
  <area **shape**="circle" **coords**="100,50, 8" href="menu.html" alt="Menu" />  
</map>

**Area Attributes**

**Coordinates** Specifies the coordinates of an area**.** The coords attribute specifies the coordinates of an area in an image map.The coords attribute is used together with the shape attribute to specify the size, shape, and placement of an area.The coordinates of the top-left corner of an area is 0,0.

**name** or **id** mapname Specifies the name of an image-map

**x1, y1, x2, y2** If the shape attribute is set to "**rect**", it specifies the coordinates of the **top-left**

corner and the **bottom-right** corner of the rectangle

**x, y, radius** If the shape attribute is set to "**circle**", it specifies the coordinates of the circle

**center** and the **radius**

**x1, y1, x2, y2,..,xn,yn** If the shape attribute is set to "**poly**", it specifies the coordinates of the edges of the

polygon. SO, each point. If the first and last coordinate pairs are not the same, the browser will add the **last coordinate pair** to close the polygon

**shape** rect Other: rectangle, circ, circle, poly, polygon. Specifies the shape of the area

**href** URL Specifies the target URL of the area

hreflang New. Language\_code. Specifies the language of the target URL.

A two-letter language code that specifies the language of the linked document.

To view all available language codes, go to our Language code

<http://www.w3schools.com/tags/ref_language_codes.asp> - en, Irish ga

**target** \_blank Which window to open the page in. \_parent \_self \_top

framename Specifies where to open the target URL

media query New. Specifies media/device the target URL is optimized for. Default value: all

nohref Not supported in HTML5

tag Specifies the relationship between the current document and the target URL

type New mime\_type Specifies the MIME type of the target URL

rel New alternate

author

**Other attributes: (not commonly used)**

1. *bookmark*
2. *external*
3. *help*
4. *license*
5. *next*
6. *nofollow*
7. *noreferrer*
8. *prefetch*
9. *prev*
10. *search*
11. *sidebar*

# Figure and Figure Caption – FIGURE and FIGCAPTION

This is new in HTML 5. The <figure> tag is used for self-contained flow content (like images, diagrams, photos, code, etc). The content of the figure element should be relevant to the image. The "figcaption" element should be placed as the first or the last child of the "figure" element so that it is used as the heading or footer of the caption. Notice that you use the image tag nested within figure to identify the figure to use in the web page.

**<figure>**

**<figcaption>The Blarney Castle </figcaption>**

**<p> A view of the Blarney Castle </p>**

**<img src="blarney.jpg" width="500" height="300" />**

**</figure>**

# Lists

* 1. In XHTML 2.0 a List Module was proposed. All of the lists have things in common such as having a beginning and end, and list items. The list item can be ordered or unordered. List items may change the character of the bullet or numeric values and can be indented. There are several types of lists & elements.
  2. Let’s look at the basic list tags that have been around since HTML. Labels are also used with forms.

1. Definition lists - dl, dt, dd,
2. Numbered Lists – nl
3. Ordered Lists – ol, li
4. Unordered Lists - ul, li
5. Label element - label

# Definition Lists

This type of list was used to create pages with terminology and definitions, and to create lists without bullets or numbers that could be used with the image tag to create an image list; and was used to indent lists (with the DD tag) without using blockquotes (to avoid italics)

1. **DL** is the **container** tag for the list
2. **DT** is for the **term** (like a word in a dictionary)
3. **DD** is the **definition** (like the paragraph in the dictionary)

Notice the  **indentation** of  the DD content

Sample – You could use an image for a  bullet & indent a list  without blockquote using the definitions list.

**<dl>**

**<dt>Coffee</dt>**

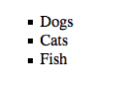
**<dd>Black hot drink</dd>**

**<dt>Milk</dt>**

**<dd>White cold drink</dd>**

**</dl>**

# Unordered Lists - UL

UL is a container element; it contains list item tags only. Bullet lists are not sequential. You can change the bullet type. Each item is identified with a list item tag. You can modify the attributes of the entire list, by modifying the UL tag, or, individual list items by modifying the LI tag. Modify the unordered list, which you can do with any UL or LI tag!

* + 1. **Type** = circle | disk | square

**<ul type="square">**

Now, **type** and **compact** are deprecated. You should be using styles to configure these attributes. But as you can see browsers still support them.

1. List-style-**type** (circle, square, alpha, roman, latin, greek)
   * 1. 1 – numeric
     2. a – lower case a, b, c
     3. A – upper case A, B, C
     4. i or I – Roman number| A | i | I
     5. circle | disc | square
2. List-style-**image** (notice the inclusion of url(path/imagename to identify the relative or absolute url)
3. List-style-**position** (to have inside or outside the flow of content)

Below you can see this example using inline styles and embedded style rules.

**<ul style="list-style-type:square;">**

**<li>Dogs</li>**

**<li>Cats</li>**

**<li>Fish</li>**

**</ul>**

<style>

ul.**circle** {list-style-type:**circle**}  
ul.square {list-style-type:**square**}  
ol.upper-roman {list-style-type:**upper-roman**}  
ol.lower-alpha {list-style-type:**lower-alpha**}

ul.duck {list-style-image:**url('images/duck.gif'**)};

**</style>**

**<ul class="circle">**

**<li>Dogs</li>**

**<li>Cats</li>**

**</ul>**

# Ordered Lists - OL

The elements in the  ordered list will  renumber if a new  item is inserted. In newer versions of HTML you have to change the order sequence in style sheets. **Start** – You can only modify (now) the **starting number** for the entire list

Tables

Tables were commonly used to provide page layout structure in early HTML and continue today. The lesson covers the basic table. Most browsers support the table tags now. So there are limited issues across browsers. Most issues are really with the style information applied to the browser.

Microsoft has always had more capabilities programmed into the layout and design of the tables, including the ability to quickl**y collapse the borders to a single thin border.** If you use tables, you should validate your page on multiple browsers. However, you will see that with the support for the later editions of CSS, these problems occur less. Today, the table, rows, and cells, all have attributes including the style attribute that can be accessed programmatically in the HTML and JavaScript code and formatted using style sheets.

**Table** – The parent tag to the table.

**<table> ….. </table>**

**Tables** wraps table **rows (tr)** inside a **tbody**, **thead** or **tfoot** element and **colgroup** **around col elements**.

**<table> <tr>…………………... </tr ></table>**

**<table> <tbody> <tr>…………………... </tr></tbody></table>**

**<table> <colgroup> <col>…………… </col></colgroup></table>**

TIP - If you need to use a form make sure the form tags are not nested inside the table.

**Border** – will work but can be set in CSS style sheets. Set to “1” to set the number of pixels. Set to 0 or “” for no border.

**<table border=”1”>**

**Table Caption – caption**

**<table border=”1”>**

**<caption>Number of students enrolled over two years</caption>**

**</table>**

**Table Heading and Footers – thead and tfoot**

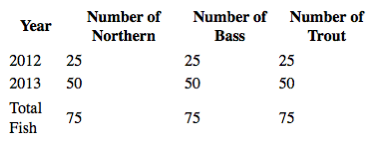
1. **Thead** is the heading for the table
2. Theadincludes the row of the table heading row cells. So the Thead is a group of TH tags for the first row. The Thead group will always be in the **first** row
3. **Tfoot** will always be the last row in the table.

**Table Heading Cells - TH**

You should know the **Table Heading** tag <th> is useful for later when you create your tables on the fly with code and data from a database

**Table Row (TR) and Table Data (TD)**

The table rows and data are identified with these tags. Table data cells are always nested within table row tags.



Basic example – Notice the head, foot, and body sections!

**<table>**

**<thead>**

**<tr>**

**<th>Year</th>**

**<th>Number of Northern</th>**

**<th>Number of Bass</th>**

**<th>Number of Trout</th>**

**</tr>**

**</thead>**

**<tfoot>**

**<tr>**

**<td>Total Fish</td>**

**<td>75</td>**

**<td>75</td>**

**<td>75</td>**

**</tr>**

**</tfoot>**

**<tbody>**

**<tr>**

**<td>2012</td>**

**<td>25</td>**

**<td>25</td>**

**<td>25</td>**

**</tr>**

**<tr>**

**<td>2013</td>**

**<td>50</td>**

**<td>50</td>**

**<td>50</td>**

**</tr>**

**<tbody>**

**</table>**

**Nesting tables –** Did you know you can nest tables? **But it’s tricky! Use an editor if you are a beginner!** Again, if you are going to do this, a web editor tool is very handy! The key is that you need to make sure the inner table is nested within the set of TD tags

**Cellpadding** and **Cellspacing** - **Deprecated in HTML 5**.

This is not supported in HTML 5. So use style sheets. Usually **cellpadding** is set to 5-10 pixels to give some space between the text and border or contents in the other cell. However, cell spacing is not often used when the borders are turned on. The attribute makes the borders thick between the cells and isn’t as attractive. Often, students turn this on, and they can improve their design by setting the **cell spacing to 0.**

**<table style=”cellpadding:5; cellspacing:0;”>**

**Aligning Content in Cells – Deprecated in HTML 5**

We used **align** and **valign** a lot in basic HTML. It's useful to use valign for header cells that simply puts the text **vertical alignment** at the top, center or bottom of the cell. You can also center the **horizontal text** in the header cell with **align**. However, as you will see, with the new XHTML, some of these presentation attributes will be deprecated. So, we are shifting focus to the cascading style sheets. However, for beginning students, you can use these attributes to learn how to setup the layout of your table cells.

Here is the old way:

**<tr><td valign=”top” align=”center”>Products</td></tr>**

And here is the newer method with inline styles:

**<tr><td style=”vertical-align:top; text-align:center;”>Products</td></tr>**

As you can see, the **names of the attributes of the tag**, may be **different** than the **style names** used in the style sheet rules. That is why we had you get the reference book on HTML and CSS. If you use an editor such as Dreamweaver and Visual Studio they come with references on the syntax for these styles.

**Other attributes – FYI Only**

**Headers** is the table **headers related to a cell**. Just pass the **header cell ID** as the **value** of the attribute.

**Scope** - Specifies the scope (which cells the table header applies to) of the table header

**Summary – Deprecated in HTML 5 -** Do **not** use the Summary attribute.

Although border is supported if set to “” or “1” it is recommended to use stylesheets.

**Rowspan** and **colspan** are commonly used as well to help **merge cells**. Although if you are going to do extensive table design, you should try to use a graphical user web editing program which will speed your development and provide validation of your XHTML code.

**colgroup** – group columns  
**col**  
**colspan** is the number of columns a cell should span (merge)

**row**  
**rowgroup**

**rowspan** is the number of rows a cell should span (merge)

Forms (Optional)

Your book does not cover forms until you learn JavaScript. But forms are HTML elements known as controls. Forms are important to web sites. We use them to collect information from the user. **Forms**, select, label, label, fieldset, input, button, form, label are all elements nested within the form tags.

Form

The form tag is important. All form elements are also known as controls or form fields. They are always inside the pair of form tags. With XHTML you can have more than a pair of form tags. However, later when you learn server side programming you will find that this will not work, **so stick to a single form in a web page for now.**

There are many attributes with the form tag. When you create a form, you are expected to provide an ID/name, action, and method for the form. Make sure to review the basics of forms from your book. Another good site for examples is the W3Schools at <http://www.w3schools.com/tags/tag_form.asp>.

**<form enctype="text/plain" action="mailto:myname@company.com" method="post">**

**</form>**

**<form action="mypage.aspx" method="get" >**

**</form>**

**Action Attribute**

The action attribute identifies the program to process the form. In the old days we used CGI, which means common gateway interface, programs created with languages such as Perl and C++ to process the form. You will see that other technologies are used more often today to process forms. Sometimes when you are just testing a basic form you can send it to your email using action=”mailto:myname@company.com”. But this sends the data encoded and difficult to read, as you will see below.

You may also see **action=”javascript:void(0);”** which is used to process the form using JavaScript and you need to **stop** the form from submitting the form.

**Encytype Attribute**

IF you are going to send form data attached, like a file, you need to set the encoding type with the enctype attribute of the form tag. The encoding can be enctype="**multipart/form-data**" or enctype="**text/plain**" or enctype="**application/x-www-form-urlencoded**".

**Method Attribute**

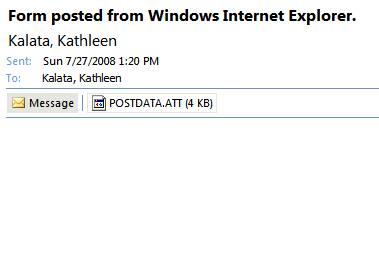
The method attribute identifies which method is used to send the form data. The two methods are post and get. We will discuss these later. Get is less secure as the information in the form is passed in the URL in a set of name and value pairs called a **querystring**. The form data is passed in the URL, so it is not secure even if you use **HTTPS** (SSL with http). The data is clearly shown in the URL in the browser and can be captured in the history of the browser!

About Security: So, if your form contains passwords you have to use **method="post"** but post is still sent as clear text and can be viewed on the network using a basic network packet sniffer program or virus. If you want the form to be secured you have to use SSL to encrypt the information. You know a web page uses **SSL** when you see **https** instead of http in the URL.

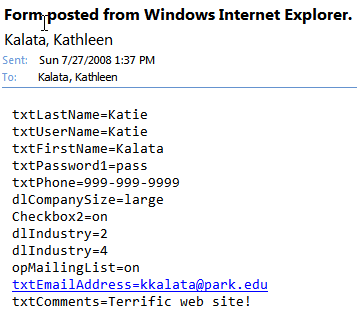
It’s important to note that routers and switches today do help limit local capturing of data, if they are secure. But, once on the Internet, there’s no guarantee of the route packets will take. So, there’s no guarantee that the packets aren’t intercepted and read along the way. So always use secure encryption with anything sensitive.

Notice with the Online Classroom, you “log in” with a secure method, https. But the rest of the communication is not encrypted. Some sites may choose not to encrypt every page sent, if the data contained is not sensitive. Only the login information is encrypted. You’ll find some types of sites, like financial; banks, hospitals, health care and military will be encrypting every page request. This does take additional processing on the client and server.

**Post**

****You can use post and email you the results to your email address. You will need to set the action to <mailto:youraddress> and you need to set the enctype attribute that is used to identify the mime type to send data.

There is another catch. You must also use the name attribute in your form, along with the id attribute. Without the name attribute, the values will not be passed.



If you attach it, you may need to save the attachment and then open it in Notepad.

If you use **enctype=”text/plain”** then the form is **sent in the email message** as shown. The name and values are displayed for you. As you can see this is not a friendly version, but is adequate for testing purposes at this point in the course.

Server-Side Programs

You will use ASP, PHP, JSP, ColdFusion, C++, to process the form and email the results in a more user friendly format.

<form method="post" **enctype= "application/x-www-form-urlencoded"** action="https://www.comp.com/login.cgi">

*QueryString Example*

Often an instructor or student wants to know if a course is being offered next term. One-way to list the products in a category is by using a shortcut, which has a querystring attached. The querystring is separated from the URL **by a ?.** “?category=mensclothing” is the querystring. The names and values are separated with an equal sign. You can see the **name=value pairs are separated with an ampersand (&).** You cannot simply pass special characters or blank spaces in the URL. Blank spaces are **encoded into %20**. They have to be **URLencoded** before you can pass the values. SO YOU NEED THE FIELD NAMES WHEN THE FORM IS SUBMITTED!!

**http://www.mycompany.com/getproductlist.aspx?category=clothing&style=mensware**

Why would you use this? Well, imagine how much time would be wasted if every day, you had to ‘lookup’ information on this page all day long. Going through the form is monotonous. So, this way, the course developers, faculty, operations and instructional designers can quickly locate the course information for any ‘current’ course and term.

If you use get, you are **limited to the number of characters allowed in the URL** or address and this is not encrypted data. As you can see, the data will be appended to the URL with a QueryString as explained in the readings.

Test

Here is the complete URL shown above. You can see the **exclamation point** was a special character and was **urlencoded as %21.** Blank spaces if there were any would be replaced with %20.

http://www.mycompany.com/FeedbackForm.html? **txtUserName**=Katie&txtPassword1=MyPasswordSecret &txtPhone=999-999-9999&**dlCompanySize**=large&**Checkbox2**=on&dlIndustry=2&dlIndustry=4 &**opMailingList**=on&txtEmailAddress=kkalata@mycompany.com&**txtComments**=Terrific+web+site%21

Notice that the names are sent based on the form field names. If no name or id is used, you may see generic ones used. Notice that even passwords are send unsecured this method. So, next time you are online, look in the address bar and the querystring.

# MENU

The problem – this was **deprecated**, then **redefined** in HTML 5. So support is limited across browsers.

The <menu> tag defines a list or menu of commands. It is relatively new, but it is not always used with the traditional forms. The <menu> tag is used for context menus, toolbars and for listing form controls and commands.

A toolbar with two menu buttons on it ("File" and "Edit"), each of which has a dropdown menu with a series of options. But remember you can’t create a windows program! These menus appear in a browser.

**Label** attribute indicates a visible label for the menu.

**Type** identifies which menu to display. Default is list, which specifies a list menu. **Context toolbar list** are the options. A list of commands (li elements), which the user can perform or activate context Specifies a context menu. The menu must be **activated** before the user can interact with the commands toolbar. Specifies a toolbar menu. Active commands, allows the user to interact with the commands immediately. The type can be toolbar, context or list. You can nest menus within menu tags, and within **</li>** elements. Here is a simple menu list.

**<menu type="context" label="File">**

**<li> <button type="button" onclick="file\_new()">New...</button></li>**

**<li> <button type="button" onclick="file\_open()">Open...</button></li>**

**<li> <button type="button" onclick="file\_save()">Save</button> </li>**

**</menu>**

# Datalist

Datalist is **New** and currently only works in **Opera** browser. Opera is the browser on **Nintendo Wii** devices and is available on mobile devices like the Apple **iPhone** and **iPad**. You just get a textbox in FireFox!

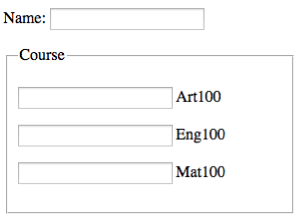
The purpose is to limit what the user can enter in the textbox!

<input list="menu" />  
  
<datalist id="drinks">  
  <option value="wine">  
  <option value="beer">  
  <option value="tab">  
</datalist>

# Form Fields

You can use several new techniques to **group** and **label** controls.

# Label

Label control will combine a label control with another element. You can separate the element and the label with the **for=** attribute and set the value to the id of the form element. The Label works with many form elements not just a textbox. For simplicity of the examples, label is not always shown.

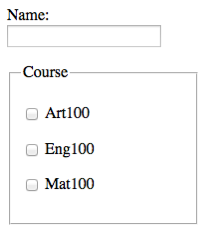
<p><**label**>Name: <input type=”textbox” id=”txtName”></label></p>

<p><**label for=“txtName”**>Name: </label>

<input type=”textbox” id=”txtName”></p>

# Fieldset and Legend

Group controls within a **fieldset** element. A legend element can be used to identify the information in the group as a title. **Legend** will put a title or caption, on a group of controls. Notice that grouped controls also have a box around them in some browsers.

<form>

<p><label>Name: <input></label></p>

**<fieldset>**

**<legend> Course </legend>**

<p><label> <input type="checkbox" name="course"> Art100</label></p>

<p><label> <input type="checkbox" name="course"> Eng100</label></p>

<p><label> <input type="checkbox" name="course"> Mat100</label></p>

**</fieldset>**

</form>

# Input Form Fields - Common

Creates a field where the user puts in some content. You can set the field type to identify the content.

1. **Text** - a single line of text. **Size** shows the number of characters **displayed** and **MaxLength** are used to control the length and how much information **can be inserted** into the field. So you can stop them from entering too many characters. Default size is 20 characters. **Disabled** will disable the control.   
    **Required**, **readonly, Autocomplete (new), Autofocus (new)** and **pattern** (new) are useful attributes. Pattern allows you to provide regular expressions, rules for what the user can enter. [0-9] Means any numeric value. This pattern is the 2 digit state code. You will learn more about this in other courses.

**<input type=”textbox” id=”txtName”>**

**<input type=”textbox” id=” txtName” size=”8” maxlength=”20”>**

**<input type=”textbox” id=” txtName” pattern="[A-z]{2}">**

1. **Password** – a textbox with \* \* \* in place of characters

**Password: <input type=”password” id=”txtPass”>**

1. **Hidden** - is used to pass data to the web server that does not appear on the web page, but does appear in the source code on the client. There is no need for a label since it does not appear on the page.

**<input type="hidden" id="course" value=”cust1234”/>**

1. **Checkbox** - a checkbox that sends a Boolean value (**Yes** or **No** valueis returned**!**)

**<input type="checkbox" id="course" value=”AR100”>Art 100**

**<input type="checkbox" id="course" value=”CH100”>Chem100**

1. **Radio** - when the names are identical, the option group of radio buttons functions together and you can only select one of the options at a time. (**Yes** or **No** valueis returned**!**) If no default is returned then No is returned. Ad you can see, you have to have the same name to represent the group of items. You can only select 1 from the radio button list. The value is what is passed to the web server.

**<input type="radio" id="MT" name="course" value=”MT100”> Math 100**

**<input type="radio" id="CH" name="course" value=”CH100”> Chem 100**

1. **Submit and Reset Button** - can be submit, reset, and command buttons. You can use an image for the button and modify the label of the button. Submit will send the name and values to the server or program to process the form. Reset will return the form to its normal state, usually clearing all fields.

**<input type="submit" id="btnsubmit" value=”Click Me” />**

1. **Button** – Usually this is to intercept the submit event to do some local processing like data validation or animation. **Value** – is what the button text will be displayed. In this case a javascript is called to validate the data when the button is clicked.

<input type="**button**" **value="Click me"** **onclick**="validateThisForm()" />

1. **Image** is and **image button** and is used to display an image that can be clicked and is treated like a submit button. It will also send the **x, y coordinates** of where you clicked with the upper left is 0, 0 on the image itself, not on the web page. So, you can make up a program capturing these coordinates. SRC and ALT are used just as they are with the image tag, and you can also use styles and the width and height properties.

<input type="**image**" **src**="imgSubmitButton.gif" **alt**="Submit" />

1. **File** is used to **browse** for the location and file name for a file you will upload. It will put a textbox and a Browse button for you. Selecting a file simply puts the path and name into the textbox. It does not do the uploading. You need a server program to do that.

<input type="**file**" name="myupload" />

**New** Input tags for **Telephone, email, time (**no time zone information). Notice the **min**, **max**, and **step** attributes.

<p><label>**Telephone**: <input type=”**tel**”></label></p>

<p><label>**E-mail**: <input type=”**email**”></label></p>

\* I think the idea is you can combine these now with Regular Expressions as part of HTML without JavaScripting but the browser support is lacking.

**More New input fields**

1. **Time** – as you can see, **Safari** and Google **Chrome** supports it, but not FireFox 6.

<p><label>**Time**:

<input type=”**time**” **min**="10:00" **max**="23:00" **step**="300">

</label></p>

1. **Date** (no time zone)
2. **Datetime** (UTC time zone)
3. **Datetime-local** (works in **Safari**, can scroll to the date)
4. **Month** and **week** with calendar control

**<input type="month" />** will in Safari show 2011-09

1. **Color** – for a color picker – is just a textbox not a **Color Picker** yet
2. **Number**– number defined with spinner control (1 line list box in Safari)
3. **Range** – number defined with slider control (very cool)
4. **Search** – used for searching
5. **URL** – for urls

For all practical purposes, many of these are not functional and simply put a textbox in place in Safari and FireFox.

However, look to newer browsers to implement them. If anything, they may make it easier to program the javascripts!

# Textarea – Multiline Textbox

Multi-line textbox. Used to create a text input field with multiple lines using the rows and cols attributes. Notice that if there is a basic text box, html will use the input control. Text within the **Textarea** or **pre** element does not begin with a new line. So formatting the content will be an issue! Uses a **fixed font** like courier to display the contents by default. But this can be changed. You can put **default values** between the tags.

1. **Rows** and **Cols** can be configured as **height** and **width** in style sheets. But rows and cols are not deprecated.
2. **Disabled**, **name** and **readonly** are still supported.
3. **Unlimited** number of characters, can be limited with **maxlength** (new)
4. Also supports the new attributes, **autofocus** and **required**.
5. **Wrap** is new and can be set to **hard** or **soft** to identify if the value should be wrapped.

**<textarea rows="5" cols="40">Put your comments here</textarea>**

<http://www.w3schools.com/html5/tag_textarea.asp>

# Select – Dropdown and List Boxes

Provides a selection list box and a dropdown list. The difference is that a dropdown list you set the size to 1 to only display 1 item. A list box generally displays a small number of items. The option tag is used with the select tag to identify each item in the list. You can set a default value to appear using the selected attribute for the option tag. And you can use the keyword multiple to allow the user to select multiple items.

**Size** lets you decide how many items to display at one time in the list. **Multiple** lets you pick one or more with holding the shift or option keys. **Selected** is the default-selected item.

**<select multiple=”multiple” size=”1”>**

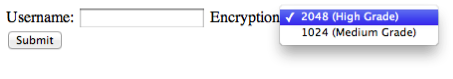
**<option value ="ruby">red</option>**

**<option value ="emerald" selected=”selected”>green</option>**

**<option value ="saphhire">blue</option>**

**</select>**

# KeyGen - New

New to HTML 5 – The picture on the right is from **Google Chrome**.

The <keygen> tag specifies a **key pair generator field** used for forms.

When the user clicks on the submit button, a **private key** is stored **locally**, and a **public** key is **sent to the server**.

Autofocus, disabled, name are all available to use for this element

You have to identify which **form** by **name** that the key should be used for.

You need to specify the security **algorithm** (rules) to use. Set **keytype** to – **RSA** or other

**Challenge** will identify the value of the keygen should be challenged when submitted.

<form action="getkey.aspx" method="get">

Username: <input type="text" name="firstname" />

**Encryption: <keygen name="security" autofocus="autofocus"**

**form="mysecureform" keytype="rsa" />**

<input type="submit" />

</form>

# Output - New

Only in **Opera** browsers AND **Chrome** – used to store the **value of a calculation**. But it does this **onforminput**!

<input name="price" type="**number**"> +  
<input name="number" type="**number**"> =  
  
<**output** name="result" **onforminput**="value= **price**.**valueAsNumber** + **number**.valueAsNumber"></output>

# Meter and Progress Bars - New

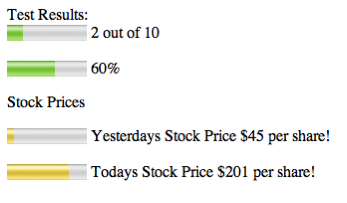
Support is only available right now in **Opera** and **Safari.**



**Progress bar** – Most often used to show an object's downloading or uploading progress. Max and value are numbers that you configure. Max is the ending number. Clearly you can write code to change the values!

TIP – some companies just use an animated graphic and swap it out when the progress is finished!

**<progress value="60" max="100"></progress>**



**Meter** - Also known as a **gauge**, the meter defines a **scalar** measurement within a **known range**, or a **fractional** value, like **disk usage**. The <meter> tag should not be used to show progress. It’s more static.

**Min** and **Max** – mean the typical max and min, start and ending point on the guage.

**Value** is the current value and can be set programmatically.

High and low and optimum are markers for values. The gauge turns YELLOW at these points.

1. **high** high value
2. **low** low value
3. **optimum** Best values

**Test Results: <br />**

**<meter value="2" min="0" max="10">2 out of 10</meter> 2 out of 10<br /><br/>**

**<meter value="0.6">60%</meter> 60% <br/>**

**<p>Stock Prices</p>**

**<meter value="45" min="25" max="250" high = "200" low ="50" optimum="125">**

**Stock Price</meter> Yesterdays Stock Price $45 per share!<br/><br/>**

**<meter value="201" min="25" max="250" high = "200" low ="50" optimum="125">**

**Stock Price</meter> Todays Stock Price $201 per share!**

# Noscript

Not as common - <http://www.w3.org/TR/html5-author/the-noscript-element.html#the-noscript-element>

# COMMAND BUTTON

**New in HTML 5 -** This is new and so it is only supported in **Internet Explorer** and **Safari.** (But I couldn’t get this to work in Safari!)

Creates a **command** button, like a radiobutton, a checkbox, or a button. But this is only visible if it is inside a **menu** element. Command can be used to specify a keyboard shortcut.

**<menu>**

**<command onclick="alert('Hello World')">Click Me!</command>  
</menu>**

# Date and Time – TIME

This can be used to show the date and/or time. Remember these **are not automating the process**, but **indicating** that it is a date or time, which might be useful for **content management systems** and **search engines**.

**<p>Last edited on <time>2011-09-26</time>.</p>**

# RUBY – FYI only as demonstration

**ruby, rt, rp** You could have a whole class on Ruby annotations.

They are often used with other languages, formulas, or annotations.

**<ruby> OJ <rp>(<rt>Orange Juice<rp>)</ruby>**

# MATH

Other namespaces have their own properties, attributes and elements. These have the same problems we have in HTML, that they are not fully supported in all browsers.

This sample, of the quadratic formula, was in **Safari**, but did not work in Firefox 6 or Chrome.

From [MathML](http://www.w3.org/TR/html5-author/namespaces.html#mathml-namespace) namespace - <http://www.w3.org/TR/html5-author/namespaces.html#mathml-namespace>



<!DOCTYPE html>

<html>

<head>

<title>The quadratic formula</title>

</head>

<body>

<h1>The quadratic formula</h1>

<p>

<math>

<mi>x</mi>

<mo>=</mo>

<mfrac>

<mrow>

<mo form="prefix">−</mo> <mi>b</mi>

<mo>±</mo>

<msqrt>

<msup> <mi>b</mi> <mn>2</mn> </msup>

<mo>−</mo>

<mn>4</mn> <mo>\*</mo> <mi>a</mi> <mo>\*</mo> <mi>c</mi>

</msqrt>

</mrow>

<mrow>

<mn>2</mn> <mo>\*</mo> <mi>a</mi>

</mrow>

</mfrac>

</math>

</p>

</body>

</html>

# Embedded Content and Media

Media Attribute Module – was what XHTML used to support images and multimedia. Although most browsers support images, not all cellular/portable/other devices have or will. The **media** attributed identifies the type of viewer that the image meant to be displayed on. (**Screen**, **print**, handheld, tv)

You can locate information on media types at <http://www.w3.org/TR/1998/REC-CSS2-19980512/media.html>

In previous versions, we used the DIV element and inserted background image. Then, some elements such as tables **allowed background images**. **Netscape** wasn’t supporting all of these capabilities.

Samples – You can have 2 versions of the graphic for each output. The device chooses the tag to display. You can also set this in the style sheets.

**<span src=“screen.jpg" media="screen">Me at work</span>**

**<span src=“print.jpg" media="print">Me at work</span>**

# Audio

We used to use **bgsound** to play when the page loaded. Now, we have finally a new tag in html 5, audio and it also works with streaming media.

Then there was plug-ins required like Flash. These were often proprietary and you had to pay for them. Some were free but yes others you had to pay for just to play music.

**Music Formats** – still can be problematic. So the audio tag lets you include multiple versions!

1. **Ogg** file, and will work in **Firefox 3.5**,/6 **Opera 10.5/11** and **Chrome 3/11**. **NOT Safari 3/5.**
2. **Wav** – **Not IE 9** but it works with all the others. Just they are very big files!
3. **MP3** files work well in **Internet** **Explorer** and **Safari.** And **Chrome 11.** **NOT FireFox (3 or 6!) or Opera**.

Attributes include **autoplay**, **loop**, **preload** (auto, metadata or none) and **src** (specify the url). Preload is useful because it loads the audio file when the page loads, making playback easier.

**Controls** will display controls. Default controls include, toggle play and pause, seek, volume, full screen toggle, caption/subtitles if available, and audio track if available.

**<audio controls="controls">**

**<source src="mysong.ogg" type="audio/ogg" />**

**<source src="mysong.mp3" type="audio/mpeg" />**

**Your browser does not support the audio element. Get the latest version!**

**</audio>**

**<audio src="mysong.ogg" controls="controls">**

**Your browser does not support the audio element. Get the latest version!**

**</audio>**

# Video

You should always check to see if video is supported. Again this used to be more of a problem developing across browsers, but there still are some issues.

**Video Formats** – still can be problematic. So the audio tag lets you include multiple versions! Your video editor will compress and save the files in the appropriate format with the appropriate **codec**.

\*These were tested in Firefox 6, Safari and Firefox.

1. **Ogg** - Work in **Firefox 3.5**,/6 **Opera 10.5/11** and **Chrome 3/11**. **NOT Safari 3/5 or IE**
   1. Videos work with **Theora video** codec and Vorbis audio codec
2. **WebM** – **NOT Safari 3/5 or IE**
   1. Videos work with **VP8 video** codec and Vorbis audio codec
3. **MPEG4** - **Internet** **Explorer** and **Safari.** And **Chrome 11.** **NOT FireFox (3 or 6!) or Opera**.
   1. **Videos work** with **H.264 video** codec and **AAC audio codec**

**Attributes** are similar to audio, but the additional **audio** attribute allows you to set to **muted** so you don’t hear the sound. You can also choose **poster** attribute that shows an **image** to represent the video.

Although the first code works, since the tag supports multiple formats, the second option is better.

**<video src="mymovie.ogg" width="320" height="240" controls="controls">  
Your browser does not support the video tag.  
</video>**

**<video width="320" height="240" controls="controls" poster=”/images/fish.gif”>**

**source src="mymovie.ogg" type="video/ogg" />**

**<source src="mymovie.mp4" type="video/mp4" />**

**<source src="mymovie.webm" type="video/webm" />**

**Your browser does not support the video tag.**

**</video>**

# Canvas

The HTML5 canvas element **uses JavaScript** to draw graphics on a web page.

A canvas is a **rectangular area**, and you control every pixel of it.

The canvas element has several built-in methods for drawing paths, boxes, circles, characters, and even adding images!

**Step 1 Create the Canvas object** – note some styles may not work right in the browser (background-color)

**<canvas id="myCanvas" width="200" height="200" style="border:1px solid #c3c3c3;">**

**Your browser does not support the canvas element.**

**</canvas>**

<script type="text/javascript">

**var c=document.getElementById("myCanvas"); // gets the canvas object, represented as c**

**var cxt=c.getContext("2d");** **// gets 2Dimensional graphics library object**

**cxt.fillStyle="#336699";** // use the object change to a **blue** **fill** color

**cxt.fillRect(25,25,150,75);** // use the object to **fill** the rectangle.

// **Top left** is **0,0** or 25,25 in this example

**var img=new Image(); // add an image object to insert an image**

**img.src="mydog.png";** / / identify the image source

**cxt.drawImage(img,0,0);** // actually insert the image at x, y location

</script>

**Draw Lines**

cxt.moveTo(10,10); // go to this point

cxt.lineTo(50,50); // draw a line from current point 10,10 to this point

cxt.lineTo(25,25); // draw a line from the new point, 50,50 to 25,25

cxt.stroke(); // stroke means to draw it.

**Draw Circle**

cxt.fillStyle="#FF0000";

cxt.beginPath();

cxt.arc(70,18,15,0,Math.PI\*2,true);

cxt.closePath();

cxt.fill();

**Use Gradient instead off Fill color to fill rectangle**

var grd=cxt.**createLinearGradient**(**25,25,150,75**); // gradient object completely covers rectangle

grd.addColorStop(0,"#FF0000"); // starting color

grd.addColorStop(1,"#00FF00"); // ending color

cxt.fillStyle=grd; // set the drawing tools to use the gradient

cxt.fillRect(**25,25,150,75**); // fill the rectangle

# Local and Session Storage – in lieu of cookies!

In the past we used Cookies to store local information in small text files, which were in control of the user and limited in number and size. Cookie data is sent in the heading packet and written by the browser, but web programmers can use JavaScript to program and write cookies, but the new data is only accessed when requested. If you close the browser, some web pages are then reloaded and you can see the numbers are still there.

**LocalStorage** – stored with no time limit

**<script type="text/javascript">**

**localStorage.lastname="Smith";** // sets the variable

**localStorage.firstname="John";**

// writes out the variable to the page. This only works when you write out the page code again.

**if (localStorage.pagecount){** // if they have a page counter, then continue.

**localStorage.pagecount=Number(localStorage.pagecount) +1;** // add 1 to the page counter

**}**

**else{**

**localStorage.pagecount=1;** // set the value to 1 since they have not been there before

**}**

**document.write("Hello " + localStorage.firstname + " " + localStorage.lastname + "<br />");**

**document.write("Visits: " + localStorage.pagecount + " time(s).");** // write out the counter #

**</script>**

**SessionStorage** – dies when the session dies or browser is closed. Just replace localStorage with sessionStorage.

**<script type="text/javascript">**

**sessionStorage.lastname="Smith";**

**sessionStorage.firstname="John";**

**if (sessionStorage.pagecount){**

**sessionStorage.pagecount=Number(sessionStorage.pagecount) +1;**

**}**

**else{**

**sessionStorage.pagecount=1;**

**}**

**document.write("Hello " + sessionStorage.firstname + " "**

**+ sessionStorage.lastname + "<br />");**

**document.write("Visits: " + sessionStorage.pagecount + " time(s).");**

**</script>**

# Embed

There was an embed tag, then it was gone, now it’s new to HTML 5 again. Basic attributes include **height**, **width**, **src** (with the URL of the item being inserted in the page) and **type**, to identify what type of file or mime-time. All browsers’ support the embed tag.

List of **MIME Media** types: <http://www.iana.org/assignments/media-types/index.html>

**<embed src="balloons.swf" width="500" height="500" type="application/x-shockwave-flash" />**

**<embed src="balloons.swf" width="500" height="500" />**

# Object

The object tag is similar to embed. All browsers’ support the object tag. Objects include: images, audio, videos, Java applets, ActiveX, PDF, and Flash. Commonly used for applets and proprietary objects. **Height** and **width** and name are supported and **usemap** for client-side imagemap support.

1. **Data** is the **URL** of the **objects** data (like parameters needed to be passed)
2. **Type** is the **MIME** type of the data in the **data attribute**

**<object width="500" height="500" data="balloons.swf"></object>**

# Iframe

Similar to frames, without all the window issues. iFrames contain pages within pages. Attributes include **name**, **height**, **width**, **scr** (URL) of the other page, **srcdoc** (so you can specify the HTML directly instead of a page… so it can be more interactive and generated on the fly) When you set the height and width to smaller than the page content, you get **scrollbars**.

1. To avoid appearing out of the page design, you can use **seamless** so that it appears as part of the page,
2. Sandbox lets you set **restrictions in the iFrame content**: **allow-forms**, **same**-**origin**, **scripts**, and **top-navigation**.

**<iframe src="http://www.yahoo.com" seamless="seamless" height="300" width="300"></iframe>**

# Obsolete and Deprecated Tags

Although browsers continue to support many tags, officially the standards do not recommend developers to use these tags. [W3C list of Obsolete Elements](http://dev.w3.org/html5/html-author/#obsolete-elements). Some elements and attributes are [obsolete](http://www.w3.org/TR/html5-author/obsolete.html#obsolete-but-conforming-features) but do conform and are sometimes used.

## Border property

On images if you use the border property you must use the value 0.

<img border=”0” src=”image.gif”/>

## Script

The attribute values are case sensitive. Do not specify language or type. If you do, use these values only.

<script language=”JavaScript” type=”text/javascript” />

# Completely Obsolete Tags

1. Tags such as **<tt>, <i>, <b>, <big>, and <small>** were **allowed** but **not** recommended in XHTML.
   1. However, **I and b and small remain** in the list for **HTML 5** for now.

**Non-Conforming Tags**

* 1. These tags should not be used because they are [not conforming](http://www.w3.org/TR/html5-author/obsolete.html#non-conforming-features).
  2. If you use these in your projects, you must have a reason documented with EACH instance!
  3. There are many attributes that are not to be used because they are non-conforming. It’s best to assume that all attribute are disallowed unless they are explicitly allowed, by checking the HTML 5 standards.

1. **Applet** - Use **embed** or **object** instead. This may not be possible with some browsers. Reference: <http://www.w3.org/TR/html5/obsolete.html#the-applet-element> The W3 wants all plug-ins to be managed in a consistent manner but that is not often the goal with vendors like Oracle/Sun. They continue to post on their site to use different methods. We will cover this in more detail when you learn about applets.
2. **Center** - This is often inserted if you use an editor to create your web page!
3. **Frame, frameset, noframes** – **Use iframe and CSS or server side code**. Many companies used and are using frames. Frames have not been supported in the strict versions of the standards and are now obsolete in HTML 5.
4. **Font** – This is often inserted if you use an editor to create your web page! Previous versions of HTML used the <font> tag. Then, used attributes in the font tag to modify the font face and size. Later you will learn about styles, and we still configure the font family and size. So this might be useful as a reference.

**<font face="Arial" size="6">Fish</font>**

|  |  |
| --- | --- |
| Relative | Absolute |
| 1 | 8 |
| 2 | 10 |
| 3 | 12 |
| 4 | 14 |
| 5 | 18 |
| 6 | 24 |
| 7 | 36 |

* 1. The **font family** is identified with **face**.
  2. Here is the catch. The only fonts the user can see are what are on their system. So only use the basic fonts on your site, except for images.
  3. You can use the **absolute** or **relative** font size
  4. This is the listing of the **font size** and **relative** font numbers

Here are the rest of the tags that you should **avoid**.

1. **Blink** - Although blink animations were the rage in the early days of web programming, they were quickly deprecated and were thought to be a potential risk for people with epilepsy conditions. They are not consistent with accessibility guidelines today and should be avoided even if a browser supports them.
2. **Marquee** – The marquee created an animated rolling text which for a while only worked in Internet Explorer and not Netscape. But the preferred method today is to use CSS transitions or plug-ins. Attributes included start and stop, scroll, slide, alternate, text, direction (up, down, right, left), truespeed and interval delay, scrollamount and loop. Reference: <http://www.w3.org/TR/html5/obsolete.html#the-marquee-element>**Acronym** - Use abbr instead.
3. **Bgsound** - Use **audio** instead.
4. **Big** – consider h1 heading, strong or mark. Small is still supported in HTML 5.
5. **tt** – Was used for keyboard teletype, and not use kbd for keyboard input, code for computer code, and samp for computer output

Older tags and not often used. Avoid these.

1. **Dir** - Use ul instead.
2. **Isindex** - Use an explicit form and text field combination instead.
3. **Listing** - Use pre and code instead.
4. **Nextid** - Use GUIDs instead.
5. **Noembed** - Use object instead of embed when fallback is necessary.
6. **Plaintext** - Use the "text/plain" MIME type instead.
7. **Rb** - Providing the ruby base directly inside the ruby element is sufficient; the **rb element is unnecessary**. Omit it altogether.
8. **strike** - Use del instead if the element is marking an edit, otherwise use s instead.
9. **Xmp** - Use code instead, and escape "<" and "&" characters as "&lt;" and "&amp;" respectively.
10. **Basefont**
11. **multicol**
12. **nobr** and **spacer**

# Frames - deprecated

Frames are discussed in your readings. They are useful in sites with no access to server side programming, but need some way to manage multiple web pages. The header, menu, and footer are usually single pages that do not change. Only the main content area changes in the web page.

The main page contains a frameset. Think of the Online Classroom. The menu on the left, the header on the top are fixed, but the content area changes on each page. They are physically, three separate pages.

You can have nested framesets but it's not often used. The code is not hidden from users. They can easily get to your page outside of the frame unless you protect the page with programming code.

How do you know if the site uses frames? Look at the source code of the home page in the online classroom and see if you can locate the frameset and frame tags.

The book presents common layouts for frames. You can set the size and frame properties in the main web page. You can set the target for the frame, so when the user clicks on the menu, the content in the main section changes. However, you can change this in your code.

If you do work with frames, you may want to use a web development editor. If you have complex and multiple layers of frames you should consider alternate methods of organizing your site.

IMPORTANT: Please, do not use frames for your homework! You will lose serious amount of points if you do! Make sure to read the directions for the homework. I included a few comments about frames, but it’s best not to use them unless there is no other option.

In other courses you will learn about master content pages in ASP.NET that provide an easier way to create sites that appear to be frames, but the browser doesn't know that the content was generated from multiple source files. This is a much better way, more efficient way, to create web sites than with frames.

The next page contains information on Forms. Forms are not covered until JavaScript in your book. However, they are important, so I also provided some information below. This is optional reading for now but will be required when we cover Forms and JavaScript. SO, you can review it now, but the homework does not require the form until homework 3.

# HTML Tutorials

There are so many great sites for beginning web tutorials. Here are just a few links you may want to visit. Be aware that many of the tutorials on the web today are older, because the companies are now selling their resources as "courses" online. However, they are usually a great place to start!

1. A great place to start is a great tutorial called [HTML for the Conceptually Challenged](http://www.arachnoid.com/lutusp/html_tutor.html" \t "_blank).
2. [W3Schools](http://www.w3schools.com/html/default.asp" \t "_blank) – has one of the most extensive lists of HTML references and samples. Start here for the [Introduction to HTML](http://www.w3schools.com/html/html_intro.asp" \t "_blank).
3. [HTML Code Tutorial](http://www.htmlcodetutorial.com/" \t "_blank) – For a great beginning tutorial [start learning HTML here](http://www.htmlcodetutorial.com/document/" \t "_blank).
4. Other useful sites that contain references, resources, and links to more resources.
5. [HTML Writers Guild HTML Resources](http://www.hwg.org/resources/?cid=14" \t "_blank) Contains a link to excellent material on [HTML](http://www.hwg.org/resources/?cid=26" \t "_blank) although it’s older
6. [HTML Goodies](http://www.htmlgoodies.com/primers/html/" \t "_blank) – another site with lists of links and also has some reference material on many web related topics including HTML.
7. [HTML Sandbox](http://htmlsandbox.com/) – used to test web pages. If you have a problem, this is a useful tool!
8. [WDVL HTML Tutorial](http://www.wdvl.com/Authoring/HTML/) – Web Developers Virtual Library tutorial
9. [HTML Color Codes](http://html-color-codes.info/) – contains a chart of colors and Color Picker
10. [HTML Tutorials](http://www.webmonkey.com/tutorial/tag/html) – [WebMonkey Tutorials](http://www.webmonkey.com/tutorial/) are also good for beginners.

# Basic Web Design

You can create a great looking site with some basic HTML and graphics. Great examples of web site design can be found on the [Newark1.com](http://www.newark1.com/) web site. Also you can look at web award sites to see examples of good web page design.

1. [Interactive Media Awards](http://www.interactivemediaawards.com/)
2. [Webby Awards](http://www.webbyawards.com/)
3. [Clio Awards](http://www.clioawards.com/)
4. [Favorite Web Site Awards](http://www.thefwa.com/)

###### Graphics

Locate lots of free and shareware graphics online. [Microsoft Office ClipArt and Media Home](http://office.microsoft.com/en-us/clipart/default.aspx) has lots of great free art and photographs for online use. There is an agreement you must comply with to use the graphics. But, for this course, you may use the clip art and graphics from the Microsoft site.

Here are links to design related web sites.

1. [Newark1.com](http://www.newark1.com/) – Great examples of web site design can be found on the [Newark1.com](http://www.newark1.com/) web site.
2. [Style Guide for Online Hypertext](http://www.w3.org/Provider/Style/Overview.html) – contains some useful tips, although the document is old.
3. [Goodpractices.com](http://www.goodpractices.com/" \t "_blank) contains design tips and best practices and [HyperText Markup Language](http://www.goodpractices.com/html.htm" \t "_blank) tutorials.
4. [Coolhomepages.com](http://www.coolhomepages.com/" \t "_blank) is useful to get ideas on how to layout your web pages.
5. [Web Pages That Suck](http://www.webpagesthatsuck.com/" \t "_blank) - sorry, but that's the name of the site by Flanders Enterprises. They do have some web pages that really are good examples of poor design.
6. [Web Design Group Reference Section](http://www.htmlhelp.com/reference/" \t "_blank) - more reference links including to an [HTML 4.0 Reference](http://www.htmlhelp.com/reference/html40/" \t "_blank) guide.
7. [WDVL CSS tutorial](http://www.wdvl.com/Authoring/Style/Sheets/) – Web Developers Virtual Library tutorial
8. [RGB to Hexidecimal Color Converter](http://www.telacommunications.com/nutshell/rgbform.htm)

# HTML Editors

Web pages are created using markup code called HTML and style rules – CSS

1. The W3C (<http://www.w3.org>) maintains standards

Tools for creating Web pages

1. Any basic **text editor** such as Notepad.
2. **Web page editors** - make document creation easier, correct more errors at design time, and help you identify choices such as what attributes and values are available
   1. You do not need not know XHTML to create XHTML using editors
   2. Shortcuts to typing tag names, spell-checker,
   3. There are WYSIWYG XHTML editors
3. Examples
   1. **Early editors** such as HotDog and FrontPage, Netscape Gold, HotMetalPro, America Online
   2. **Word** – **Convert** into HTML
   3. **Content Management System** – MySpace and Facebook – creates the HTML for you
   4. Tucows - freeware, shareware and commercial software programs [www.tucows.com](http://www.tucows.com) and CNET.downloads.com and www.sourgeforge.net.
   5. Adobe/Macromedia **Dreamweaver/CS4**
   6. Microsoft
      1. Visual Studio .NET Web Developer Express Edition free - <http://www.asp.net>
      2. Visual Studio .NET Professional
      3. Web Matrix – a replacement for the Express Edition.
      4. Microsoft Expressions Web

###### *Site Builder Software*

You cannot use content manages systems, site builder or templates for this course for your homework assignments. However, you should know about them. Many third party hosting providers provide site builder tools and templates such as [IBuilder](http://www.sitebuilder.com) by Site Builder, WordPress, and DotNetNuke.com.

# Accessibility

Note: Web sites are now also being required to follow additional standards such as Accessibility. Check out <http://www.w3.org/standards/webdesign/accessibility> and <http://webaim.org/> for the latest information on accessibility.

One of the things you can do to help accessibility, mobile devices and program compatibility is to maintain consistency with standards. As you deviate from standards, you will find the application may work now, but in the future they may be more difficult and expensive to maintain.

As part of my role at other companies, I developed the policy and procedures required so that courses were compliant with the latest accessibility standards. Many programs like Adobe PDF and Dreamweaver, provide tools to help validate the documents with most of these standards. These are not perfect but they are a good start. Accessibility has always been important. But meeting needs of our military and wounded warriors is something educational professionals are invested in. So, while this is a course on basic web development, we want to make sure you are going out into the world implementing the best web site architecture, which includes support for accessibility and mobile devices. Although I am not working anymore in administration, these policies are in place and the University is committed to them. So, if you know of anyone with accessibility needs, point them to the instructors and counselors who can help them. The web developers and instructional designer can help modify web site content if you find it’s not compliant.

**Section 508 of the Rehabilitation Act, §1194.22**

The Section 508 is a list of recommendations to meet government accessibility standards:

1. Short description of standards: <http://webaim.org/standards/508/checklist>
2. Government reference: <http://www.section508.gov/index.cfm?&FuseAction=Content&ID=12>

**Web Content Accessibility Guidelines (WCAG) 2.0**

This standard contains a subset which meets the Section 508 standards, plus additional standards.

1. The official guidelines: <http://www.w3.org/TR/WCAG20/>
2. Summary of the guidelines: <http://webaim.org/standards/wcag/checklist>

# Additional Resources

Tutorials - there are so many great sites for beginning web tutorials. Here are just a few links you may want to visit. Be aware that many of the tutorials on the web today are older, because the companies are now selling their resources as "courses" online. However, they are usually a great place to start!

1. A great place to start is a great tutorial called [HTML for the Conceptually Challenged](http://www.arachnoid.com/lutusp/html_tutor.html" \t "_blank).
2. [W3Schools](http://www.w3schools.com/html/default.asp" \t "_blank) – has one of the most extensive lists of HTML references and samples. Start here for the [Introduction to HTML](http://www.w3schools.com/html/html_intro.asp" \t "_blank).
3. [HTML Code Tutorial](http://www.htmlcodetutorial.com/" \t "_blank) – For a great beginning tutorial [start learning HTML here](http://www.htmlcodetutorial.com/document/" \t "_blank).
4. [HTML Reference](http://msdn.microsoft.com/workshop/author/html/reference/elements.asp" \l "ie40_htmlref" \t "_blank) - a list of HTML elements from Microsoft
5. The [W3C](http://www.w3.org" \t "_blank) is really the one site to start with learning about all web related standards. You should become very familiar with their site.

Other useful sites that contain references, resources, and links to more resources

1. [HTML Writers Guild HTML Resources](http://www.hwg.org/resources/?cid=14" \t "_blank) Contains a link to excellent material on [HTML](http://www.hwg.org/resources/?cid=26" \t "_blank) although it’s older
2. [HTML Goodies](http://www.htmlgoodies.com/primers/html/" \t "_blank) – another site with lists of links and also has some reference material on many web related topics including HTML.
3. [Case Western Intro to HTML](http://www.cwru.edu/help/introHTML/toc.html" \t "_blank)  - this is really old, back to 3.x. But, sometimes you need to know what happened before to understand where you are now. If you go to their web page at [http://www.case.edu/help/](http://www.case.edu/help/" \t "_blank), you can see additional HTML reference guides.
4. [Goodpractices.com](http://www.goodpractices.com/" \t "_blank) contains design tips and best practices and [HyperText Markup Language](http://www.goodpractices.com/html.htm" \t "_blank) tutorials.
5. [Coolhomepages.com](http://www.coolhomepages.com/" \t "_blank) is useful to get ideas on how to layout your web pages.
6. [Web Design Group Reference Section](http://www.htmlhelp.com/reference/" \t "_blank) - more reference links including to an [HTML 4.0 Reference](http://www.htmlhelp.com/reference/html40/" \t "_blank) guide.
7. [Web Pages That Suck](http://www.webpagesthatsuck.com/" \t "_blank) - sorry, but that's the name of the site by Flanders Enterprises. They do have some web pages that really are good examples of poor design.
8. Broken links? If you see a broken link in this course, send an email to: **KKalata@oakton.net** and put Broken Link in the subject line. Identify the page and the link that is broken.