**Basic HTML (and XHTML) Tags v21 81/10/2021**

**By Margaret S. Menzin © May be used for non-commercial purposes with credit.**

1. **Absolutely Essential**

**Html, head, body, title, h1-6, and comments tags and !DOCTYPE declaration; meta tag**

1. **Organizing Text
a. At the detail level**

**Paragraph, break line, span and div tags,**

**The id and class attributes for paragraphs, headlines, spans, and divs**

**Blockquote, and Preformatted tags**

**Horizontal Rule tags**

**b. At a page level**

 **Nav, main, and optional header, footer
Article, section, aside**

1. **A very little styling**

**Bold, italics, underline, superscript and subscript tags**

1. **Lists**

**Unordered lists (bulleted), ordered lists, and list item tags**

1. **Links**

**Anchor, hypertext reference tags, id attribute**

**Relative and absolute addressing**

1. **Tables**

**Table, table row, table header, table data item tags, thead, tfoot, tbody**

## Colgroup, Column span and row span tags

1. **Inserting Graphics**

**Img tag with src and align attribute**

**Tiling**

**NOTE:** Also as much styling as possible should be done with CSS (Cascading style sheets), but these notes have some examples which show you how to over-ride that with in-line styling.

**Other Reading:**

By the time you finish the HTML coding you should have read all of Robin Williams’
The Non-Designers Design Book. It’s a fabulous little book, on reserve in the library.
You will want to read parts of it more than once. Many of the ideas are also found at [www.peachpitpress.com](http://www.peachpitpress.com/) The Robin Williams book is available thru the ACM book site at <https://learning.oreilly.com/library/view/the-non-designers-design/9780133966350/>

**Preface: You may ignore this if you have never written HTMLor XHTML.**

These notes teach you how to write HTML5.

For older browsers, almost all of HTML5 is backwards compatible to earlier versions of HTML. All modern browsers support HTML5.

In terms of style, I believe in using the syntax rules which allow you to turn an HTML5 page into XHTML in case you need namespaces (e.g. for ChemML etc.) by changing the DOCTYPE and MIME.

So, all tags are closed, all tags are lower case, and all attribute values are quoted. This is also considered to be 'best practices' for HTML5.

A lot more information about XHTML may be found online. If you write HTML as in this course it will be easy to convert to XHTML should you ever need to. The Appendix discusses those matters.

1. **Absolutely Essential**

**Tags:** In (x)HTML5 almost all tags come in pairs – an opening tag and a closing tag –e.g. <p> and </p>. Notice that the closing tag has a slash at the start.

A very few tags are empty and do not need to be closed or may be closed in minimized form – notably <br /> for line breaks.

Because I believe it is easier to learn XHTML and XML if you develop the correct habits with HTML5 I will always close tags. *These notes however use HTML5*.

**< !DOCTYPE…> The DOCTYPE preprocessor information and**

**<html > </html>**  **…….. The HTML tag**

**<!DOCTYPE html>
<html lang=’en’>**

**All your web pages will begin with these 2 lines**

Everything in between the <html> and </html> tags is interpreted as HTML.[[1]](#footnote-1)

In other words, the html tags mark the beginning and end of your web page.

The page is further divided into a head and a body.

**<head> </head> …….. The head tag**

Again, opening and closing tags.

 The header contains the title and will often contain your JavaScript code and
 Cascading Style Sheets (CSS) or links to them..

 Often the header also contains meta-tags (e.g. keywords about the content of
 your page to make it easier for search engines to find it.)

**<meta> ….the meta tag *which has no closing tag*** Every head section should begin with
 <meta charset = "utf-8">

 There are other meta tags used to describe content etc. These used to be used
 to optimize search engine results, but are so over-used they are now ignored.

**<title> </title> …… The title tag**

 Opening and closing tags.

The title is what is displayed at the bottom of your browser or in the tab of the browser. It should be informative and short. Since only the first part of the title will show if the user has multiple tabs open, it should start with words which distinguish it.

Do not add spaces between the tags and the title:

 <title>The right way to make a title</title>

<title> The wrong way to make a title </title>

**Having met <head> and <title> you know that your pages will all begin:**

**<!DOCTYPE html>
<html lang=’en’>**

**<head>**

 **<title>My Informative Title</title>**

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

 **</head>**

Please also notice that the meta tag is NOT closed, nor is the !DOCTYPE (which is strictly speaking not a tag.)

**<body> </body> …… The body tag**

The body has everything that's not in the head.

 It comes after the head, so that by the time the body is executed anything in

 the head has been read.

**At this point you have the skeleton of all web pages:**

 <!DOCTYPE html>

 <html lang=’en’>

 <head>
 <meta charset="UTF-8">

 <title>MyPage</title>

 : possibly more stuff, certainly info about who wrote the pa**ge** when, etc
 </head>

 <body>

 : more stuff

 </body>

 </html>

 Now let’s see what we can do in the head and body.

**Do Problem 1 and email it to me.**

**<h1> </h1> thru <h6> </h6> ….. Heading tags**

Any text between the tags will be in **bold** face.

There will be a blank line after your headings.

Heading sizes go from **<h1> </h1>** (biggest) down to **<h6> </h6>** (smallest).

Best practices are to make your most important heading <h1>, the next most
 important <h2>, etc. You will then use CSS to set the sizes for <h1>, <h2>, etc.

 It is important for screen readers that your page have *only one*  h1 tag.

 If you have a heading a subheading you should group them in <hgroup>.

 <hgroup>
 <h1>HTML5, CSS3 and JavaScript</h1>
 <h2>Three technologoies working together</h2>
 </hgoup>

**<!-- --> …… The Comment tag**

 Anything between these tags is ignored by HTML. Use them!!

 This is where you put important information to document the code :

* Your name
* The date you wrote this code and the date of any subsequent revisions
* References - This code from such and such a book, page ….etc.

You will also use to enclose JavaScript code, so that HTML doesn't try to execute it.

**Do HTML Problem 2 and email it to me.**

1. **Organizing Text**
2. **Organization at the detail level**

**<p> </p> ……. Paragraph tags**

These mark the beginning and end of a paragraph.

 Each paragraph will automatically start on a new line, with one blank line inserted

 after the last paragraph. [[2]](#footnote-2)

**The class and id attributes** Sometimes we wish to identify a paragraph so that later we may style it – e.g.
 display it with a different font or color, or even move it around.

 To do this we give the paragraph an **class attribute.** Multiple paragraphs may
 have the same class.If we want to identify it *uniquely* we give it an **id attribute.**

To give a paragraph an id we write:

 **<p id = ‘intro’> … </p>** It is important to know that:

* **An attribute is a property of an element.** If you think of an element as a noun ( a thing), then the attribute is an adjective
* **An attribute is always specified** in the opening tag with:
 **attributeName = ‘attributeValue’**
* **The name of the attribute should be given in lower case or camelCase**.
* You may use either single or double quotes around the attribute value, but they must match.
* The value for an id should **not** have any whitespace (blanks) in it.
* **An id attribute may be given** to paragraphs, headings, and other elements which you will meet.
* **The id must be unique.** You may not use the same id for a paragraph and a heading or for two paragraphs.
* As in any naming in any language, the id you give should be descriptive – e.g. intro for introduction.

**<br /> ….. Line break tag**

This inserts a line feed (start new line). [[3]](#footnote-3)

**<div** id =”TableOfContents ”**> </div> The div tag**

A div is used to delineate a large block of code. Typically this is done so that
 you may style it separately or to place a block of code somewhere on the page.

This is useful to center several paragraphs, heading, etc. at once.

The id attribute is optional, but very common.

 In AJAX a <div> may also be used as a placeholder for where text will appear.

.

**<span> </span> The span tag**

 The span tag is used to identify a small amount of text – often in the middle of a
 paragraph, almost always for styling.

 A span may have an id or not.

Some typical uses of span are to change the appearance of a word or two (e.g.
 to display them in red type).

**NOTE: <span**> is used for short amount of text(e.g. a few words) and **<div>**  is used for large blocks of material. For example, a div may include several paragraphs.

 In the lingo of HTML and CSS we say that:
 **div is a block element and span is an in-line element.**

Block and in-line elements have different styling and layout capabilities.
 (Stay tuned.)
 HTML5 has introduced new block level elements (similar to <div> but with a hint
 as to their meaning). For a clear description of these, see part b below
 on organizing general level. This is expanded on at
 <http://www.w3schools.com/html/html5_semantic_elements.asp>

**The class attribute (again)**

Just as the **id** attribute identifies an element **uniquely,**  the **class attribute** is used to identify a bunch of elements for purposes of treating them the same way. For example, our code might contain:

<div class=’finePrint’>

 :

</div>

 :
<p class = ‘finePrint’> …….</p>

The next 3 tags are used much less often:

**<blockquote> </blockquote> ……. Blockquotes**

For long quotes. The quote will be indented or italicized or otherwise set off.

**<pre> </pre> …. Preformatted text**

Everything in between will appear exactly as you typed it - indenting,

paragraphs,etc. Useful for quoted material, poetry, etc.

**<hr /> …… Horizontal Rule**

 This draws a line across your page. With the rise of small screen devices, this
 has pretty much disappeared from web pages.

You may specify an id or a class for a line:

 **<hr class=”wideline” />**

You may collect some fancy horizontal rules for your pages, but these days it is
 more common to use a border or an image (see below).

**Ref: read Robin Williams The Non-designer’s Design Book p. 11-42**

**Special text characters, also called special entities, (e.g. @ and those used in French and Spanish) may be found at places in the** [**Web Centric References section on HTML documentation**](http://web.simmons.edu/~menzin/WebCentricResources.html#XHTML_Documentatin) **(or in** [**bookmarks**](http://web.simmons.edu/~menzin/IT320/IT320_Bookmarks.html#XHTML_Documentatin) **for an older version of this course - in the 2nd paragraph ) and at**

[www.w3.org/MarkUp/Guide/Advanced.html](http://www.w3.org/MarkUp/Guide/Advanced.html%20%20) **and on the page on superscripts, etc. in the Chapter 0 folder.** This is reference material to use when you need it – not to learn.

You will find that you use some of these so often that you naturally learn them. Here are a few I use often:
 **HTML ignores more than one blank space in a row. If you want an additional
 blank space use the special entity &nbsp; (non-breaking space)
 &mdash;** is a dash the width of the letter m
 &**apos**; is a apostrophe (looks like a single quote, but it is used inside words.)

**Do HTML HW Assignment 2 and email it to me and yourself.**

1. **Organization at the more general level** *This section is more related to designing sites and complex pages, and you may defer reading it until then.*

**<main> </main> ……The main tag**Inside the body, every page should have one and only one main tag. It delineates the main content of the page.

**<nav> </nav> …… The navigation tag**This will hold the content for navigation around the page or web site.
Later you will learn how to place it where you want it to got.
Very often the nav content is an unordered list (see below.)

**<header> </header> …. The header tag
<footer> </footer> ….. The footer tag**These two tags are optional. If they are at the start and end of the body then they pertain to the whole page; if they are inside an article then they pertain to the article.

It is common to see a footer tag on a large site with links such as About, Contact Us etc. You may, however, just put such a section at the end of a page.

**<article> </article> ….The article tag
<section> </section> …. The section tag
<aside> </aside> …. The aside tag**

If you have a web page which is best divided up into pieces, then each major piece would go inside an article tag. If the major piece is divided up further, then each of those pieces would go inside a section tag.

The aside is used for the kind of content which would go in a sidebar.

Here is an example of how this kind of organization works: (The bold facing is just to make the structure stand out)

<**body**>
<**header**>
 <h1>Majors at Simmons University</h1>
 This page has an overview of all the wonderful majors
</**header**>

<**main** id='allMajors'>
 <**section** id='MCS'>
 <h2>Majors in Mathematics, Statistics, Data Science, Computer Science. Informaton
 Technology and Web Design and Development</h2>
 <**header**>Why MCS is fabulous- this header pertains only to this section</header>
 <**article** id ='CS'>

 <h3>Computer Science</h3>
 <p>Paragraph about our CS major></p>
 <p>Another paragraph about CS</p>
 <*aside* id ='CSJobs'>
 <p>Enticing info about what our CS majors do</p>
 </**aside**>
 </**article**>
 <**article** id = 'DsandA'>
 <h3>Data Science and Analytics</h3>
 <p>Verbiage about DS&A</p>
 </**article**>
 :
 </**section**>

 <**section** id = 'NatSci'>
 <h2>Natural Sciences</h2>

 :

 </**section**>
</**main**>

<**nav**>
 :
</**nav**>

</**body**>

 Think about this example:

* Each chunk which I might want to style differently or move around has an id, so I can do that with CSS.
* As the chunks become les and less general the font size of the headings decreases – e.g. h1 for the whole page, h2 for each section (broad group of majors), h3 for each article (single major) etc.
* There is one & only one h1; there is one & only one main.
* This example is a little extreme. Other than pages in a reference manual, on a government /regulatory web site or similar page we rarely put that many levels of detail on one page. Most people just won't read it.
1. **A very little styling<span> </span> The span tag**

**<b> </b> …….. The bold face tag** (keywords)

**<i> </i> …….. The italics tag** (alternate voice)

**<sup> </sup>…….. The superscripts tag**

**<sub> </sub>……. The subscripts tag**

**<b> and <i>** are really about presentation, not content, and, strictly speaking, they don’t belong in HTML. But, they have been around for a very long time, and they are so widely used that they have been allowed to stay in HTML.

In general, it is better to use **<strong> … </strong>** than <b> …</b>, and it is better to use **<em> …</em>** than <i> … </i>. (‘em’ stands for emphasis.) This is because readers for the visually impaired can render ‘strong’ and ‘em’ but not b(old) and i(talics).

In fact, the w3c came close to deprecating <b> and <i>, but they are just too common.[[4]](#footnote-4)

<sup> and <sub> have been left in HTML because they are useful for equations.

1. **Lists**

Lists may be either ordered (1, 2, 3 etc) or unordered.

In either case, the items on the list go between **list** **item** tags **<li> </li>**

**<ul> </ul>……. Unordered List tag** (Unordered means not numbered).

The list is indented, and you may nest lists to get levels of indentation.

 **<ul>**

 **<li>**My first item **< li/>**

 **<li>**My second item**<li />**

 **<li>**My third item**< /li>**

 **<li>**My last item

 **</ul>**

 An unordered list may be given an id or a class – and that may be used to style them. For example, you may change the shape of the bullet or get rid of them.

**<ol> </ol> … Ordered List tag** (Numbered lists)

Ordered lists are numbered sequentially.

 Put each item between <li> and </li> tags..

 The numbers and new lines are automatic.

 Ordered lists may be nested, and you may mix ordered and ordered lists.

**<ol>**

 **<li>**My first item**</li>**

 **<li>**My second item**</li>**

 **<li>**My third item**</li>**

 **<li>**My last item**</li>**

 **</ol>**

An ordered list may be given an id or a class – and that may be used to style them. For example, you may change whether the list is enumerated by numbers or letters (A, B, C, etc.) This is discussed further in the CSS notes.

Examine the program **p4\_lists.html**

1. **Links**

**Absolute Links or Links to Other Pages**

**<a href=”http://the\_URL”>Words to Underline</a> The anchor tag – absolute**

The text in between the two tags is underlined. When the user clicks on it the browser transfers to the URL in the first tag.

NOTE: the quote marks are the usual ones --- not facing in any particular way.

<a href=”http://www.simmons.edu/~menzin”>My Favorite Professor</a>

This example (above) is an absolute reference.

Notice that it gives both the protocol (HTTP ---- as opposed to FTP etc.) and the complete address. These are specified in the **href** attribute.

Notice that the complete address is enclosed in quotation marks.

There is a convention that when a path name is listed (as above) without a file name at the end, then the browser will look for a file called **index.htm** or **index.html.** So your opening page should be named index.

There is also a convention that user directories (those that start ~username) will

have all their public files in a directory called public\_html.

In other words, when a viewer clicks on the text in the example, her browser will actually get the file www.simmons.edu/~menzin/public\_html/index.htm

In this case (the absolute URL) the URL completely defines where the browser is to go.

Look at the program **p5\_goodMorningAbsoluteLinks.html**

**Links to Places on the Same Page The anchor tag – same page**

**<a href =”#NamedSpot”>Words to Underline to go up or down the page</a>**

**<a id =”NamedSpot”>Where link will go</a>**

 **NOTE:In HTML4 this was coded as <a name =”NamedSpot”>** The name attribute is theoretically not recognized in HTML5,
 tho’ most browsers will handle it. **Use the id attribure.**

In order to link somewhere else on the same page you need two anchor tags –

 <a id=”ShortNameForTheSpot”>Text to link on</a>

 defines a name for the place you wish to go to.

 <a href=”#ShortNameForTheSpot”>Text to click on to go there</a>

 does the actual linking.

 Notice that both the **a id =** tag and the **a href =** tag have the address in quotation

 marks.

**In XHTML1 and in HTML4 you did not need the id= part, but beginning in HTML5
 you need the id= and further if there may be both a name (for really old legacy
 browsers) and an id attribute (and then they must have the same value.)**

In other words, in order to create a link from one place on your page to another place
 you must do all of the following:

* Decide where you want to link to and name it with
<a id=’NamedSpot’> </a>
* Be sure you have an id attribute in the tag (and sometimes a name attribute with the same value)
* Create the place(s) you want to link from and for each put
< a href=’#NamedSpot’> …</a> around the text you want to link from.

Notice the use of **#** inside the anchor where the linking is done ---this alerts the browser to look for a named place, not an absolute or (see below) relative reference.

Your link may go either up or down the page. See the links8a.html and links8b.html examples.

You may also combine links to other pages and links to named spots on those other pages. For example, let us suppose that you have built a page at with the URL

SomeComputer/MyBook/Intro.html

And that somewhere in that file you have a named anchor

 <a id =”contents”>Table of Contents</a>

Then, on some other page, if you wish to link to the Table of Contents you would code:

<a href=”http:// SomeComputer/MyBook/Intro.html#contents”>MyBook’s Table of Contents</a>

Notice that there is the usual anchor with an href (in quotes) but that the #namedSpot

comes at the end of the URL.

**Relative Links or Links to Other Pages on the Same Site**

**<a href=”OtherFileInSameDirectory.htm”>**Check Out My Other Pages**</a>**

In this case you will link to a different file (one named OtherFileInSameDirectory.htm).

Relative links allow you to keep all related files in the same directory or folder. If you decide someday to move the whole folder to another computer or another spot on that computer,

then the relative hrefs will still work, but absolute ones will need to be re-typed.

As usual, there are no spaces in URL file names, and file names are case-sensitive.

Relative references may be combined with named anchors, too, as above.

It is possible to do a limited amount of navigation in a directory using relative URLs.

Suppose that I have a directory (folder) named menzin and in it I have sub-directories named

cs101 and html\_programs. Further, suppose that my html\_programs directory has a file called fonts4.html

menzin

 cs101 html\_programs

index.html hw.html bookmarkJan00.html index.html fonts4.html

In html\_programs/index.html, a link to the fonts4 file is **href=”fonts4.htm”**

In cs101/index.html a link to hw.html is **href=”hw.html”**

To get to the html\_programs/fonts4.html file from cs101/hw.html, I first need to go up to the html\_programs directory, and then to the fonts4 file.

The **../** means go up one level in the directory tree. So the link is

**href=”../html\_programs/fonts4.html”**

The ../ gets us from the cs101 directory to the menzin directory.

From there we go to the html\_programs directory, and in it to the fonts4 file.

We will see this again with graphics links.

You may insert a link to your email with:

**<a href=”mailto:menzin@simmons.edu”>Or contact me by e-mail</a>**

1. **Tables**

***In HTML tables are used for creating charts and tables, but are no longer recommended for controlling page layout***. In the bad old days, a table with two columns (which need not have the same width) is one way to create the familiar side-bar with links to other parts of a web site. Today, using CSS is the proper way to achieve this result, and is important when you want a site to work on a wide range of screen sizes (e.g. laptops and cell phones).

Position on the page/page layout may be controlled with CSS. This is discussed in the CSS notes. Controlling position with external files for CSS works better for pages which may be ‘read’ in many formats (e.g. on hand-held devices), but has the disadvantage that an external style-sheet is not always downloaded from a web page (i.e. the layout is not saved). None-the-less, CSS is the preferred method for laying out pages.

*On the other hand, tabular data is presented in a table.*

**<table> </table> The Table Tag**

Every table begins and ends with these tags.

A table has rows (which run left to right) and columns (which go up and down, just as on a building).

A table is described by reading across the first row, then reading across the next row, etc.

All rows of a table are of the same width, and all the cells in a given column are the same width.

In HTML5, tables have a sections for the head (indicated with **thead** tags), the footer (indicated with **tfoot** tags) and the body (indicated with **tbody** tags.)

When we define a table, the order is:

<table>

 <caption>….caption here </caption>

 <thead>

 : thead stuff goes here

 </thead>

 <tfoot>

 : tfoot stuff goes here

 </tfoot>

 <tbody>

 : the main part of your table goes here

 </tbody>

</table>

A few important comments:

* The caption, thead and tfoot tags are all optional, but when used they must be in the order given.
	+ In other words, if there is a caption tag it comes immediately after the table tag (and it will be centered unless styled otherwise.)
	+ If there is a thead section it comes after the caption tag, if there is one.
	+ If there is a tfoot section it comes after any thead section and **before** tbody**.**
* The big advantage of using thead, tfoot and tbody is that the head and foot sections stay put if the user needs to scroll down the table. This is useful for very long pages.
A good example of this is at <https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_tbody>
* You may see older pages without thead etc. but you should always use tbody tags.

**<tr> </tr> The Table Row Tag**

<tr> marks the beginning of a row's description.

 Table rows can go in thead or tfoot or tbody, but most of the main data will be in tbody.

 <table>
 <tbody>

 <tr>

 *The description of the entries in the first row goes here*

 </tr>

 <tr>

 The description of the entries in the second row goes here

 </tr>

 <tr>

 The description of the entries in the third row goes here

 </tr>

 </tbody>

</table>

Notice that I have indented the table rows. Table descriptions can get complex (you can even put a table inside another table!) and it is a good idea to do this!

**<th> </th> The Table Header Tag**

**<td> </td> The Table Data Item Tag**

Each entry in a table is either a header (which is in bold) or a data item. The beginning and end of each entry is surrounded by these tags.

By default, the contents of a <th> are in bold and centered in the cell.
By default, the contents of a <tr> are left aligned.
These defaults can be over-ridden (See below). For example a column of numbers is always right-aligned.

To make your site aria-compliant (accessible for screen readers) any <th> opening tag should specify if the <th> goes across the row or down the column. This is very easy to do – you code <th scope="row"> or <th scope="col">.

**Beginning with HTML5, all attributes of tables (border, cellspacing, cellpadding, and width. etc.), table rows and table cells must be set through CSS.** That is, you use the style attribute inside the <th> or <td> tag when the stylimg applies to just one cell, as in the alignment example below. If your styling applies to the whole table (e. g. for borders) then the styling is done either in the <table> tag or using a CSS stylesheet (stay tuned.) **A very common problem is to have a column of numbers which you wish to right align.**Unfortunately, other than using some obscure CSS, the easiest way to do this is to write

<td style=’text-align:right’> …</td> for every cell where you want the data to be right aligned or to use colgroup – see <https://www.w3schools.com/tags/tag_colgroup.asp> for an example.
Similar information is at <http://htmldog.com/guides/htmladvanced/tables/> .

See the listing for **tables10b.html** (Remember that file names are case-sensitive!)

You may **align a table or a td or th cell** for purposes of wrapping text.

 <table style=”text-align:left”>…</table> Puts the table on the left side of the page, and
 the text to the right.

 The only choices are left and right.

See the various tables pages for examples, and examples of coloring both all the background and

individual cells.

Sometimes you want a cell to stretch across several columns (e.g. for a heading) or down several rows.

 <tr style=”text-align:center”>

 <th>This is the first column.</th>

 <th colspan=”3”>This occupies the next 3 columns.<th>

 <th>This is the last column</th>

 </tr>

If you are doing something complex, it is a good idea to make a simple sketch of it before you start coding. That way when you have a column or row span you will remember which cells have already been taken described. See **tables11c**.**html a**nd other programs with tables in their name. We will revisit this topic later in the course when we talk about Responsive Web Design.

Putting this all together, here is a template from <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/thead> which you can easily modify to have more rows or columns. Remember that caption, thead, and tfoot are optional

 <table>

 <caption>Council budget (in £) 2018</caption>

 <thead>

 <tr>

 <th scope="col">Items</th>

 <th scope="col">Expenditure</th>

 </tr>

 </thead>
 <!—if there is a tfoot it goes here 🡪

 <tbody>

 <tr>

 <th scope="row">Donuts</th>

 <td>3,000</td>

 </tr>

 <tr>

 <th scope="row">Stationery</th>

 <td>18,000</td>

 </tr>

 </tbody>

</table>

**Another common situation is to have a column (or row) heading which goes across several columns.**
To think about this you need to understand how a table in your code getsrendered on the page. The browser reads through your code one row at a time, starting with the first row. It that row has 8 columns, then the browser will expect each row to have 8 columns. That's fine, except that suppose the first 2 columns are about the East, the next 2 columns are about the Mid West, etc. What we really want is a table which starts out like:

|  |  |  |  |
| --- | --- | --- | --- |
| **East** | **Mid West** | **South** | **West** |
| **State** | **Pop.** | **State** | **Pop.** | **State** | **Pop.** | **State** | **Pop.** |
|  |  |  |  |  |  |  |  |

In order to do this we need the <th> which holds East to span 2 columns, likewise for the <th> which holds Mid West, etc.

We accomplish this by using colspan. When the browser comes to a cell that says

<th **colspan="2"** scope="col">East</th>

then the browser knows that there will be two columns here, but in this one row they are merged. That way the browser can keep track of how many columns there are in each row.
Of course, rowspan works the same way --- and the cell which does the spanning is in the first row. Look at [http://web.simmons.edu/~menzin/CS321/Unit\_1\_Basic\_HTML5\_and\_CSS/Chapter\_0\_Basic\_HTML/html\_programs/p8\_tables\_the\_row\_and\_column\_span.html](http://web.simmons.edu/~menzin/CS321/Unit_1_Basic_HTML5_and_CSS/Chapter_0_Basic_HTML/html_programs/p8_tables_the_row_and_column_span.html%20%20)  for an example with both rowspan and colspan cells.

 **One final word of warnin**g: **If you want to set the background color or an *empty* cell you must put some character such as &nbsp; in it**. For example:

 <td style = "background-color:yellow">&nbsp;</td>

**7. Inserting Graphics:**

**Please read the pages I e-mailed you about gif’s and jpeg’s and about large files.**

**<img src="fileLink.ext" alt=”Stop sign” /> The Image Tag**

Let us suppose you wish to insert a clip art file that is in the same directory as this html page, and that the file is named StopSign.gif At the place where you wish the image to go you code:

 <img src=”StopSign.gif” alt = “Stop sign” />Here is the text that goes next to it

You may refer to the src (source) file using absolute or relative addressing (as for links).

**NOTE**: You should always include the **alt** attribute <img src=”stopSign.gif” alt=”stop sign” />

to get a written description for visually impaired users (and those too impatient to wait for the

image to load) and for search engines.

 <img src=”StopSign.gif” height=”100” width=”100” />What a big stop sign!

The height and width attribute specify the size of the image in **pixels.**

Obviously, if you change the height and width to a different ratio than your original gif or jpeg you will distort the image (which you may choose to do.)

You also may give an id or class attraibute to an img and use css to style it (set size, position, etc.)

NOTE: As of HTML5 you are supposed to always set the border. While the border attribute may still be used inside the <img> tag, it is preferred to set the border with CSS, (coming soon).

See **InsertingGraphics**.**html** for examples.

You may (of course!) include the image in an anchor tag so that the whole image becomes a link:

<a href=”http://web.simmons.edu/~menzin/cs101”>
 <img=”smiley.gif” alt=”smiley face” />To the source!</a>

Finally

<body style=”background-image:url(“awfulStuff.gif”)>

will cause the entire background of your page to be tiled with the gif you specified.

There may also be assigned reading from two other books (on reserve):

Olin Lathrop’s The Way Computer Graphics Works and Lynda Wyman’s Designing
Web Graphics (See also [www.lynda.com](http://www.lynda.com)). I suggest you browse in these books
beyond what is assigned.

**APPENDIX About compatibility of HTML and XHTML**

**Please note: these notes now close ALL tags so as to be compatible with CSS and XHTML.**

**Also, all styling is consistent with HTML5. For example, the <font>and <align> tags are obsolete in HTML5 and have been replaced with CSS (as a separate sheet or in-line styling.)**

* If you have never written HTML you may ignore this section.
* If you have written HTML then the notes below will explain the rather minor differences.
* If you have written HTML and have HTML pages which you wish to convert to XHTML1, then go to [http://cgi.w3.org/cgi-bin/tidy](http://cgi.w3.org/cgi-bin/tidy%20%20%20)  , enter the reference for your page, and HTML Tidy will do the conversion for you. You then need to save the source (view it, highlight all, copy and paste it into NotePad on a pc or TextEdit on a Mac and save it). HTML5 may be validated at <http://validator.w3.org/check>. This process works!

**XHTML is HTML re-written in XML . For a complete discussion of this see**

[**http://www.w3.org/TR/xhtml1/introduction.html#xhtml0**](http://www.w3.org/TR/xhtml1/introduction.html#xhtml0)

**.**

 **If you are used to writing HTML, then please note the following items:**

**(See** [**http://www.w3.org/TR/xhtml1/diffs.html#diffs**](http://www.w3.org/TR/xhtml1/diffs.html#diffs) **for more details.)**

1. **All tags and attributes are in lower case.**
2. **Certain tags are “empty” and in HTML did not require a closing tag- notably <hr>, <br>, <img>. In XHTML all tags must be closed. So, for example <br />.

 Empty tags should be closed in the minimized format, and with a space before the slash: <hr />, <br />, <img src=”myPhoto.jpg” alt=”me!” />

In HTML5, because many people learned to close empty tags when they wrote XHTML1, it is permitted but not required to close empty tags. In XHTML5 (and XHTML1) closing empty tags is required. The same is true in polyglot HTML5/XHTML. Accordingly I recommend closing all empty tags.**
3. **HTML is casual about not requiring that some tags be closed – e.g <p>.
XHTML requires that all tags be closed. If you have an empty tag which may, in other pages, have content, you are advised (by w3c) not to use the minimized format. That is, use <p></p>, rather than <p /> because some old browsers may handle the minimized form incorrectly.**
4. **All attribute values must be in quotes – e.g. color=”black”, size=”3”**
5. **You must use a <!DOCTYPE…..> before your <html> tag. There are basically five good and legal choices: See the guidelines below for more information.**

 **For XHTML1.0 Strict**

<!DOCTYPE html

 PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

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

**For XHTML1.0 Transitional**

<!DOCTYPE html

 PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

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

**For XHTML1.0 Frameset (not recommended)**

<!DOCTYPE html

 PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"

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

**For HTML5 --- and the <meta ….> line is optional but recommended and goes after the <head> tag.**

**<!DOCTYPE html>
<html lang=’en’>**

<meta charset="UTF-8">

XHTML1.1 is (very) strict and less popular than XHTML1.0. There are doctypes for HTML4.01, but not always used and will be inferred if omitted. . Note that XML 2.0 was published in January 2006 and put on hold in 2009 so that resources could be devoted to HTML5. More importantly, HTML5 is in the works, so please read [Coming Attractions:HTML5 and XHTML5](file:///C%3A%5CUsers%5CM.S.Menzin%5CDocuments%5CSimmons%5CSimmons%5CIT320%5CFall_2010_CD%5CIT320_CD_2010%5CUnit1%5CCh0%5CComing%20Attractions%20HTML5.html) in these notes.

.

1. **If you are using the strict DTD (as opposed to the transitional) then you must include an XML Namespace. Writing in English you might put**<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

**More complete information about compatibility with XML agents can be found at** [**http://www.w3.org/TR/xhtml1/#guidelines**](http://www.w3.org/TR/xhtml1/#guidelines)
2. **HTML5 is here and XHTML** is less popular. Briefly, HTML5 is completely supported by new browsers (more or less),but not by legacy browsers. HTML5 is supported by IE9, but not by earlier versions of IE.
 HTML5 may be written as either HTML or XHTML, and the w3c recommends that most authors use HTML.

 **a. If you wish to write HTML5 (serialized as HTML)** you will begin your pages

 <!doctype html> or <!DOCTYPE html> followed by :
 <html lang=’en’>
 <meta charset = ‘UTF-8’>

 <head> etc……

 **The meta line is optional and will be inferred if missing; it is recommended that
 you include it.**
 Pages that begin this way will always render with the most current standards.

 **b. If you wish to write HTML5 serialized as XHTML** you may begin your pages
 <?xml version="1.0" encoding="UTF-8"?>
 <html xmlns="<http://www.w3.org/1999/xhtml>" xml:lang="en" lang="en">

 **but the s3c recommends that you omit the < ?xml …> line.**

 In both case you may, of course, choose a different charset.

**c. The main reason to continue to write XHTML is for namespaces (beyond those for MathML and SVG, which are supported in HTML5) or because you are using an extension such as ChemML.

d. It is possible to write polyglot code – which will have the same DOM
structure whether rendered as HTML or as XHTML**. If you are interested in doing this, you should go to

 <http://www.w3.org/TR/html-polyglot/#dfn-polyglot-markup>
This reference is not for newbies.

1. Remarks for the curious:

	* UTF-8 is the char encoding (which sequence of 0s and 1s represents which character.)
	* This page will be in English (‘en’.)
	* The MIME (media type) is not specified and defaults to **text/html**, which is the mime type for HTML. For XHTML the MIME type is **application/xhtml+xml**The DOCTYPE declaration is different for XHTML – see the Appendix for details [↑](#footnote-ref-1)
2. Remark: The closing </p> tag is optional in HTML, but not in XHTML. Use it! [↑](#footnote-ref-2)
3. HTML5 will accept <br> or <br /> but XHTML requires <br /> [↑](#footnote-ref-3)
4. The following is no longer available in HTML5:

 <u> </u> …….. The underline tag

 See below for how to do this with CSS. [↑](#footnote-ref-4)