HTML Overview

What is a web page?
The web page with which we are all familiar is usually not a single document; it is composed of multiple files, the primary being the HTML document. The only native content in an HTML document is text. An HTML document is actually just a text-only document. So where does all the rich media in a web page come from? Based on instructions in the HTML document, it is actually the browser that imports all the media-rich content (i.e. images, video, audio, etc.) and builds the web page.

What is HTML?
HTML is an initialism for HyperText Markup Language. HyperText allows the end user to advance to another web page by clicking on hyperlinks (a.k.a. "links"), which can be either text or images. This is what enabled the World Wide Web to take off so quickly when it was introduced in 1990. Prior to HTML, navigating the Internet was a laborious task. Markup is the programming (a.k.a. "code") that gives structure and meaning to the text content, provides instructions for downloading rich media (images, video, audio, etc), and determines how the browser should display this rich content.

HTML Software
Since HTML is really just a text document, it can be composed in any simple Text Editor (such as NotePad on the PC platform). Most programmers use an HTML Editor, which is a software program that recognizes both HTML programming plus many other web-based languages. Listed below are two free programs which you are welcome to use for this class.

PC Platform, NotePad++ : http://notepad-plus-plus.org/
Mac Platform, TextWrangler: http://www.barebones.com/support/textwrangler/updates.html

You are free to use the HTML editor of your choice. Do note, however, that since this is a programming class I will not cover how to work with any particular HTML or Text editor. If you need assistance with your tool of choice, you should consult its Help menu or the online support resources at the software's website. **Also, please note that those who are seeking a grade or certificate cannot** use Adobe's Dreamweaver, since that is a visual HTML editor which in many instances does the programming for you.

What is HTML code?
HTML programming is commonly called merely "code." Since an HTML document is a text-only document, how do you differentiate its native text content from the HTML code? Quite simply, the HTML code is contained within the greater than ">" and less than "<" symbols, which I will refer to as angle brackets. Any text outside of these angle brackets will be read as native text content. HTML has a vocabulary of elements, often (somewhat incorrectly) called tags, which are used to mark up the document. The example below uses the **P** element to show basic HTML markup.

```
<p>The P element creates a body copy text block.</p>
```

The **P** element above is composed of a start tag ** ** and an ending tag ** **. Notice that the ending tag contains a forward slash just after its opening left angle bracket. This informs the browser that you are closing your **P** element. With most tags – but not all – you must code where an element begins its markup with the start tag, and where its markup ends via the end tag.

Container elements vs. Empty elements

The **P** element above is an example of a container element, specifically a text container for establishing a body copy text block, because it contains something. There are many types of container elements. Although many can contain text content, there are some container elements that cannot contain text directly; rather they serve as containers for other elements like the **P** element. An empty element has a start tag but no end tag. Because these types of elements lack an ending tag, they cannot function as containers. The **BR** (line break) element below is an example of an empty element.

```
<p>The BR element is placed within a text block to create a line
break,<br>
 forcing any text that follows the BR element to drop to the
next line.</p>
```

HTML Hierarchy.

HTML is a hierarchical language in that some elements can serve as containers for other elements, and in turn those other elements may themselves contain additional elements. A container element that holds another element is called a Parent; an element that is contained within a Parent element is called a Child. The **BR** element in the above code is an example of such a relationship. The **P** element is the parent; the **BR** element is the child because it's contained inside the **P** element. Below is a more complex example.

```
<article>
  <p>An example of a P element with a phrase that needs to be marked
  up as having <strong>strong importance</strong>. This text block
  is marked up with both the P and Strong elements.</p>
  <p>In the text block above, the P element is the parent and the
  Strong element is the child. But the P element is also a child of
  its parent, the Article element.</p>
  <p>The Article element is the parent of three P elements as well
  as the grandparent of the one Strong element in the first P
  element. Or we could say that the Strong element is a child of the
  P element and a grandchild of the Article element.</p>
</article>
```

You cannot alter the parent/child relationship of the elements. The **P** element can never be the child of the **STRONG** element. And the **ARTICLE** element can never be a child of the **P** element. In the example above both the **P** and **STRONG** elements are descendents of the **ARTICLE** element.

Think of the hierarchical relationships of HTML elements in comparison with the relationships within your own family. Your offspring are your children, and they can never be your parents. You in turn are a child of your parents, but you can never be their parent. Both you and your children are descendents of your parents. Just as you may have more than one child, so may your parents. In HTML, Parent elements may have many descendents. Also in HTML, elements that are on the same level in the hierarchy have a sibling relationship in that they cannot be in a parent/child relationship with each other. In your own family, you may have one or more brothers or sisters, which is a sibling relationship. In the example above, the three **P** elements are siblings of each other and therefore can never be in a parent/child relationship.

Notice that in the coded example above the **P** element is indented within the Article element. This is a common programmer construct but is not required. The indentation helps you to visualize the parent/child relationships.

Basic markup for every HTML document
The code below must be used for every HTML document and is the minimum required markup. You can use more markup, but not any less.

```
1.  <!DOCTYPE html>
2.  <html>
3.    <head>
4.      <meta charset="UTF-8">
5.      <title>A topic for Search Engine Optimization (SEO) goes
    here</title>
6.    </head>
7.    <body>
8.       Text content and its markup along with markup for importing
         rich media goes here.
9.    </body>
10.  </html>
```

You may notice that the HTML code above is in color, often referred to as “color-coding.” (The better HTML editors provide color-coding.) Notice that the elements are color coded in blue, while on line 4 there is other color-coding — which we will cover in just a moment.

The coding above has also been given line numbers, which the better HTML editors also provide. There are tools called HTML Validators that will help you find errors in your HTML code by returning the line numbers where your chosen Validator believes those errors have occurred. Having an HTML editor that provides line numbers makes it easier to find those errors in your HTML code.

Line 1 above is called a “prologue” because it precedes your HTML programming. The prologue sets the Document Type Definition (DTD) which identifies to the browser the version of HTML in which your document is coded. The current version of HTML (identified above) is HTML5.

Line 2 above opens the HTML document. This is the HTML element’s start tag. Notice that its ending tag is on line 10. All other elements are descendents of the HTML element.

The HTML document is then divided into two sections, namely the head and the body. Line 3 opens the Head element, which closes on line 6. The **BODY** element opens on line 7 and closes on line 9. The **HEAD** and **BODY** elements have a sibling relationship that is very specific. The Body element must follow the Head element; it can never precede it.

The **HEAD** element, can only contain other elements specific to the head section; it can never contain body content (see “Where does my content go” below). The head section can contain meta information which the browser needs to interpret the document, SEO information, and/or other non-HTML programming such as Cascading Style Sheets (CSS) to style the document’s content and JavaScript to provide for interactivity.

On lines 4 and 5 of the coded example above are two elements which are children of the **HEAD** element. The **META** element on line 3 is an empty element in that it is not a container element and has no end tag. On line 4 we see the **TITLE** element, which is a container element. **TITLE** elements contain text which a browser uses to label a bookmark; Search Engines often look to the title text for the topic of a document. The text in the **TITLE** element does not display within the browser's view port. (The view port is that area in a browser which contains the Body element's content; it is located below the Menus, Address field, and Toolbars, but above the bottom scroll bar.)

Notice on line 4 that the Meta element has additional programming. After the Meta name there is a space followed by "**charset"**; charset is an attribute of the Meta element. Many HTML editors give attributes a different color-code than the one used for an element name. And there must always be a space between the element name and its attribute. In the case of a container element the attribute must only be found in the start tag. Many elements have attributes, but only the Meta element has the **charset** attribute. The purpose of this attribute is to identify the character set that encodes this document. Notice that following the **charset** attribute is an equal sign, which assigns the string "UTF-8" to this attribute. An attribute's string is called a **value,** and many HTML editors give it a color different from the color used for the attribute. Also notice that the UTF-8 value is inside a set of double quotes. The double quotes terminate the value of an attribute by marking the beginning and end of the attribute's value. Common coding errors are caused by failure to assign a value to an attribute with the equal sign, not properly terminating a value, and/or not placing a space between the element's name and its attribute.

What is UTF-8?
UTF-8 is the current standard for encoding the character set of an HTML document. UTF-8 is a nearly universal character set in that it will support all the characters (glyphs) used in many different languages. This is quite an improvement over the previous ISO character sets that only supported glyphs from a single language.

Where does my content go?
The Body element is the ascendant element of a document's entire content plus its markup. Here is an example using the earlier Article example.

```
1.  <!DOCTYPE html>
2.  <html>
3.    <head>
4.      <meta charset="UTF-8">
5.      <title>Body content and the Article element</title>
6.    </head>
7.    <body>
8.      <article>
9.        <p>An example of a P element with a phrase that needs to be
          marked up has having <strong>strong importance</strong>. This
          text block is marked up with both the P and Strong elements.
          </p>
10.       <p>In the text block above, the P element is the parent and the
          Strong element is the child. But the P element is also a child
```

```
        of its parent, the article element.</p>
11.     <p>The Article element is the parent of three P elements and
        the grandparent of one Strong element in the first P element.
        Or we could say that the Strong element is a child of the P
        element and a grandchild of the Article element.</p>
12.   </article>
13.  </body>
14. </html>
```

In line 5 we find the **Title** element, which contains, for bookmarking and SEO purposes, the topic of this HTML document.

The **Article** element on Line 8 and its descendents are all descendents of the **Body** element. This markup and its text content can never be descendents of the **Head** element.

HTML Syntactical Rules

- HTML code should be written in lowercase.
- Container HTML elements must close via their end tag.
- Empty HTML elements do not have an end tag and so are not closed.
- There cannot be a space before the Element name and its opening angle bracket.
- There must be a space between the start tag name and any attribute.
- Values are assigned to an attribute via the equal sign.
- Attribute values must be enclosed in quotes.
- Attributes must only be coded in the opening tag and never in the closing tag.
- If an Element has multiple attributes there must be a space between them.
- HTML is a hierarchical language, so parent-child relationships between elements must be observed.

Now that you have read the syntax rules, go back to the last coding example see how all the rules were applied.

Naming Conventions

Extensions
All HTML documents must end with the ".html" extension.

Case Sensitive
HTML document names are case sensitive. A web server will treat a document named bio.html, Bio.html, bIo.html, and BIO.HTML as different documents. Confusingly, the Mac and PC platforms are not case sensitive, so on those platforms the four different cases used for the Bio HTML document are considered to represent the same document. On the Mac and the PC, if you have a folder containing a document called bio.html and you try to save another document called BIO.HTML into that folder, both platforms will ask you if want to replace your bio.html document. But on a Web Server if you upload a file called bio.html and then later upload a file called BIO.HTML, the Web Server will accept both files, allowing both to co-exist. Why is a file name's case such an issue? Because when you reference a file in your HTML code, whether you are trying to import it or link to it, you must match the filename's case exactly; otherwise you will get a browser error. With a link you will get a "page not found" error message. When importing an image you will get a

broken image icon. Mismatched cases will not create a problem when testing pages on the Mac or PC; but will be a problem when your site is uploaded to a web server. Note that folder names (called "directories" on a web server) are also case-sensitive.

No Spaces in file names

You will use a URL when you're linking all your HTML files or want to import images and other rich media,. Since spaces are not allowed in a URL, you cannot have spaces in your file names. If your file names are comprised of multiple words, Google recommends separating the words with hyphens like so: my-long-wordy-file-name.ext. Note that 'ext' would be replace with the correct extension for a file, such as '.html' for an HTML file, '.css' for a CSS file, '.js' for a JavaScript file, '.jpg' for a JPEG image file, etc. Lastly, folders (i.e., directories) cannot have spaces in their names as well.

Final Thought

Please be solid on the HTML rules laid out in this lesson. If not you will have trouble creating HTML documents that have clean code, are error free, and are properly named.