1. A web page is not likely to appear the same on different computers. Web pages are not static images such as photographs or PDF files. Instead they are freshly "**rendered**" (drawn from instructions) each time they are displayed. The appearance of the resulting page can depend on many factors, including: screen size, screen resolution, browser window size, browser settings regarding font sizes, etc. Also each brand and version of web browser interprets "source code" (stored instructions) of the web page according to its own rules. Although web languages such as HTML and CSS are standardized, proprietary browser programs such as Windows Internet Explorer, Mozilla Firefox, Google Chrome, and Apple Safari (to name a few) each interpret these languages in unique ways which will affect the finished look of the page in different ways. Web authors must recognize this - test view their pages with many browsers - and accept that the finished look of a page is not always certain. Any computer can have multiple browsers installed on it. Any one of them can be used to test view a web page locally before it is uploaded to a web server.
2. Most languages used in web pages are text-based, meaning that the languages are stored as "**plain text**" rather than as some binary code such as in picture files or programs. As such, the safest type of program to use to edit web language is a "**plain text editor**", such as the Notepad program in Windows, TextEdit on a Mac, or Emacs or Nano in Linux. There are many free third-party text editors (from simple to sophisticated) available on the Internet. One versatile one is jEdit. There is a version of it for Windows, Mac or Linux. So it is a good one to learn if you move between many different platforms. You should avoid using more complex software such as HTML tag editors, word processors, or other graphic web authoring tools until you understand their limitations particularly with regard to data types. It is very important that web authors recognize the data types of the files that they are using. To this end, be sure that your operating system is showing you the filename "**extensions**" for each file when viewing folders on your computer. To learn more about this, read the web page about file association. HTML or XHTML files should be saved with a filename extension of either ".htm" or ".html" (which are synonymous). External CSS files should be saved with a ".css" extension. External JavaScript files should be saved with a ".js" extension.
3. The source code (text-based browser instructions) contained in web page files is composed as a collection of data objects referred to a "**elements**", including but not limited to: paragraphs, lists, tables, divisions, images, anchors (hot links), etc. The foundation of web language is HTML (HyperText Markup Language). "Hyper-text" (or far-fast text) is simply text that is spread over great distances that can be retrieved quickly. The source code of all web pages starts as plain text. That text is "marked up" using "**tags**" composed from angled brackets (the symbols "<" and ">") containing a label that defines the nature or characteristics of the text. For example, a paragraph is written encapsulated within the tags and , creating a "paragraph element". Elements can be classified as either "containers" or "empty elements". Most elements are containers, meaning that they are coded using a pair of tags which enclose content, such as paragraphs or headings. The first tag opens the element and the following one closes it, as in: content . The tag opens and identifies the element as a paragraph; the tag closes the element. Empty elements such as line breaks and horizontal rules do not contain content, they separate it. Such elements are now coded with a single "**self-closing tag**", such as `<br />` rather than as an empty pair of tags, as

in
</br>. Note the use of the slash (/) to self-close the single tag. The blank space ahead of the slash should be used to accommodate older browsers which might not recognize the notation.

4. Another way in which elements are classified is within two basic categories based on how they are rendered. "**Block-level**" elements such as paragraphs, lists, tables, and divisions are rendered as rectangular blocks with author-definable properties such as width, height, margins, borders, alignment, etc. "**Inline-level**" elements such as plain or stylized text, images, and anchors are displayed along an imaginary baseline and wrapped as they encounter a margin. Rules exist regarding how elements must be sequenced and whether they can be "nested" (positioned within another element). For example, anchors and images must be nested within a block-level element, and cannot be used outside of one. (See: http://en.wikipedia.org/wiki/HTML_element)
5. As source code is read and interpreted by web browsers, the formatting (layout and appearance) of the rendered text depends primarily on the embedded tags rather than on the layout of the source code. Carriage returns in the source code are not rendered as carriage returns, but rather treated as "**whitespace**" (a character that separates content – just like a single blank space). Whitespace characters include: blank spaces, carriage returns, tabs, line feeds, and some other layout control codes. In most web languages, the appearance of any quantity of successive whitespace characters in the source code will be interpreted and rendered as a simply a single blank space on the rendered web page. As such, multiple lines of simple plain text will not be rendered as expected, because the carriage returns at the end of each line of source code will not be displayed as carriage returns, but rather as blank spaces, causing one long line of text to be rendered. A browser will perform a carriage return only when it reads a tag that generates one, such as the
 (line break) tag, or when it encounters a margin. Text containing tab characters will not render properly either, treating each one as a single space rather than as a command to advance to a new column position. One way to override the behavior of the browser and force it to render all whitespace characters as written, is to enclose such "pre-formatted text" inside of the container tags <pre> and </pre>.
6. Computers typically offer a set of characters far larger than those normally available on a keyboard. Web language includes a large set of "**character entities**" (special character coding sequences) to produce characters such as a copyright symbol (©). The character entity that can be typed in source code to represent a copyright symbol is "©". Notice that the typical format for a character entity is to precede an identifying label (such as "copy") with an ampersand (&) and terminate it with a semicolon (;). Another very useful character entity is the "non-blank space"   which will render as a blank space, but not be collapsed with whitepaces.
7. Some elements are sufficiently complex that they must be defined with specific "**attributes**" (characteristics or supporting data) such as: width (for a table) or web address (for a link anchor). For example, a link anchor containing the text "IRSC" that should link to the college web site (www.irsc.edu) when selected (clicked upon), would be coded as the element:
   `<a href="http://www.irsc.edu">IRSC</a>`
   The text "href" is the name of an attribute that defines the anchor's "hypertext reference" (or web address). That attribute's "value" is the URL "http://www.irsc.edu". Note that attribute values should always be written enclosed in quotes (either single or double, but always a pair of the same style) in modern web source code.

8. The languages used to write web pages evolve. This means that web browsers must also evolve. The continual cycle of language and software revision affects the choice of which language version will be used to author each web page. Authors must realize that pages written using the most advanced language might not render properly in older browsers. Surprisingly, the reverse might also be true. As browsers appear in smaller devices such as cell phones and PDA's their conformance to language standards is becoming more important. Small devices require very efficient browser programs that do not have the luxury of being able to compensate for poorly written source code that might have earlier been handled by sophisticated browsers on large computers. This is why so much emphasis is now being placed on writing "**valid**" source code that passes inspection related to its use of web language syntax. It is also why newer versions of web language such as HTML 5 are not always backwards compatible with earlier versions and actually "**deprecate**" (remove from the language) inefficient or outdated tags or attributes that previously were recognized in earlier versions. The leaner the language, the smaller the browser.
9. Language validation must be performed separately for each language used in a web file. Modern web files typically contain more than one web language. For example, most ".html" files contain XHTML, CSS, and JavaScript. Separate validation resources (programs, plug-ins, web sites, etc.) exist for each language. Some can be installed on your computer and used there. Others are available on web sites and you can upload your file to that site for validation or copy and paste your source code there. The most widely known **validation site** is at the home of the Web on a server named validator.w3.org. You can learn how to use it by clicking Help & FAQ link on that site or by watching some tutorial videos on YouTube such as the ones at:
http://www.youtube.com/watch?v=Pt8GZjhExlc
or http://www.youtube.com/watch?v=--5jwQptFQI (for Mac users).
Similar (but separate) validation resources exist for CSS, JavaScript, Link Checking, Accessibility, etc. Links to these can be found on our course home page.
10. After your web files are created and verified, you will probably want to upload them to a web server so that they will be permanently accessible on the Web. For our course projects, you will submit your file (or files) for grading via the college's eLearning server. When your projects involve more than one file, you will be required to package your files (and possibly related folders) into a single compressed archive and then send submit that single file. For step-by-step directions, read the web page at: http://www.gibsonr.com/classes/howto/onlinesubm.html
After your first project is complete, a login account will be created for each student on the IRSC Student Web Server. For directions about how to upload files to this server using FTP (File Transfer Protocol), read the web page at: http://www.gibsonr.com/classes/internet/ftp.html
Students will not be required to upload to this server, but might find it useful to practice doing so. Also files that have been stored on the server can have their source code validated more easily than by upload. Note: this server is world accessible and is provided to students as a courtesy. Each student using the IRSC Student Web (Linux) Server account must agree to abide by the Indian River State College Student Standards of Conduct and District Board of Trustees (Info.Tech.) Policy 6Hx11-9.12.