HTML Primer

Table of Contents

HTML Primer

Getting Started

The following document will describe the basics of the Hypertext Markup Language and list the tagging elements and their attributes.   This primer follows the HTML 4.0 and XHTML Standard established by the World Wide Web Consortium (W3C: http://www.w3c.org).

Every web page starts and ends with the start and end tag, respectively, for an HTML Document; `<html>` and `</html>`.   These tags must be present to signal the web browser that information located between the tags must be translated by the HTML tagging code that appears in the text document.   Current XHTML requirements stipulate that the HTML code should be in lower case.

The HTML document is divided into 2 main sections, the head section and the body section.   The tags `<head>`, `</head>`, `<body>`, and `</body>` delineate these sections and are placed between the start HTML tag, `<html>`, and the end HTML tag, `</html>`.

Only text that appears between the start and end body tags will actually appear in the web browser window. Information placed in between the start and end head tags will define general information for the browser. For example, the start and end title tags, `<title>` and `</title>`, will place any text between these tags in the window title bar at the top of every browser window.   The header portion of the web page code is also appropriate for JavaScript tags, Meta tags, and Link tags which are described in another section of this document.

Of the 4 tags described so far, all are considered to be elements of the web page code. Elements can have attributes which define certain parameters of the web page.   The only element of the 4 described so far that has attributes is the body tag.   The attribute of a tag always appears in the start tag.   For example, one of the attributes of the body of a web page that can be altered is the background color.   In the code below, the body tag defines the background color to be red.

```
<body bgcolor="red">

</body>
```

There is another means of defining color for any attribute that alters the color of the web page, the text on that page, or the borders around objects.   By inserting the hexadecimal code for the color to be displayed, there is greater control over the color being displayed. The term "red" might be interpreted differently by different operating systems, different computer platforms, and different browsers. Defining the color more accurately by the

hexadecimal code will make certain that the only way a user will see something other than the color you intended is if their monitor is not displaying color correctly. Consult the section labeled "Hexadecimal Code" to learn more about color definition, but for now accept that the color "red" can be displayed with the hexadecimal code: "`#ff0000`." So the attribute definition would change to, `<body bgcolor="#ff0000">`.

Content on a Web Page

A web page is nothing more than a page of information intended for the visitor to read and navigate to additional information. This information or content is separated into paragraphs much like the content in this primer. To signal to the browser that you want blocks of content separated into paragraphs you use the paragraph start and end tag; `<p>` & `</p>`. Please refer to the section "Example HTML File – Simple" to see the implementation of this tag.

There are other elements similar to the paragraph tag that automatically determines the layout properties of the text placed within the tags. An excellent example of this is the headline tag. There are six to choose from, numbered from 1 to 6. The largest headline tag is `<h1> </h1>` and will automatically separate the text between the start and end tags from the rest of the text on the web page. The text will be increased in size from the rest of the web page text and will be bold. Headlines are meant to be headings or subheadings in a document. You should always use them in sequence meaning that `<h1> </h1>` is always first followed by `<h2> </h2>` etc. This will assist screen reader software in organizing the page for a person that cannot view the web content directly. Mixing the order of the headline tags will result in confusion for both groups of users, the sighted and un-sighted. View the section "Example HTML File – Simple" to see implementation of this tag.

Another excellent example of elements that define the layout of text on a web page is the list tags. There are several different types of list tags. The ordered list tag, `<ol> </ol>`, causes list items to be automatically numbered and spaced in outline format on the web page. The list items are defined with the `<li> </li>` tags. The other option for listing items is with a bulleted list known in HTML code as the unordered list, `<ul> </ul>`. Again the `<li> </li>` tag is used to indicate the items in the list. The web page author can also generate a definition list, `<dl> </dl>`, containing definition terms, `<dt> </dt>`, and definition definitions, `<dd> </dd>`.

Often a visual separation between content or features on a page can be achieved with a horizontal ruler (or line) that runs from the left to the right margin of the web page. The `<hr>` tag places a horizontal ruler (or line) across the page. It is also our first example of a tag that has no end tag, only a start tag. Because of this the proper way to designate an `<hr>` tag is with the end tag, forward slash at the end of the tag label, `<hr />`. Doing so means that your coding will follow the current convention of XHTML standards. It is important because it is possible that your web pages will not display properly as newer browsers are no longer compatible to former standards. The tag `<hr>` will still display a

horizontal ruler, but it might not in the future. Also the `<hr />` tag will pass HTML validator and accessibility web sites such as Bobby. The `<hr />` will separate the text above and below it with one line of space.

What do we know so far?

Well, we know that each and every web page starts with the `<html>` tag and ends with the `</html>` tag. Everything in between is read by the browser to be a web page and is decoded appropriately. The `<head>` tag contains the `<title> </title>` tags and other tags yet to be mentioned. The `<body>` start tag follows the end tag, `</head>` and can contain attributes. Page layout and formatting elements include `<p> </p>`, `<ol> </ol>`, `<ul> </ul>`, and `<dl></dl>`.

Elements versus Attributes

Every HTML document contains tagging elements that indicate to the browser how to display the content included in the HTML file. These elements form two broad categories; layout elements and text formatting elements. Layout elements predefine how the text will be arranged on the page. For example, the paragraph tag `<p> </p>` tells the browser to separate the text between the start and end tag from any previous or following content. The text itself does not carry a special font or color.

Text formatting elements like `<strong> </strong>` do not tell the browser how to arrange the text on the page but describe certain features to be applied to the text. For example, the weight of the text can be altered from the page default of "normal" to "strong" for emphasis. In this case, `<strong> </strong>` causes the text to be bold in appearance and the tagging element `<em> </em>` cause the text to be italicized. Other examples include the superscript tag, `<sup> </sup>`, and the subscript tag, `<sub> </sub>`.

In the past, bold text was coded for by the tag `<b> </b>` and italicized text was coded for by `<i> </i>` but these tags have been deprecated. A deprecated tag is nothing more than a tag that has been discontinued and in the future will not likely work with newer versions of browsers. The reason some HTML tags have been deprecated is that tags like the bold tag, `<b> </b>`, were quick conventions in earlier versions of HTML code that enabled the coder to format the text. Tags have been deprecated primarily because they no longer format text in a method that enables the content to be separated from the formatting of the text, and it is important to not use tags that the World Wide Web Consortium have ceased to authorize.

The latest version of HTML straightens out the language and moves us to a modern approach of page layout and text formatting. In essence HTML 4.0 is about separating the formatting and layout of content from the content itself. For example, in this text presenting this information covering the basics of HTML, there is the letters that make up

the words that make up the text. Then there is the separation of groups of words into sentences and sentences into paragraphs, a transition from the content to the layout. The style or formatting of the text has been changed to differentiate HTML Code from the rest of the text. While the content doesn't change, the formatting of the text plays an important role in the display of the content and, therefore, is a means of communicating information.

The information that we display electronically through a web page can be separated into the content, the formatting, and the layout allowing different browsers, operating systems, and platforms to display that information in a means relevant to each instance. This development means that a developer can code the content once and enable different devices such as a desktop computer, cell phone, and personal data assistant to all access the same content, but the layout and formatting codes differ to handle the content for each specific device. This will also allow people with physical and learning disabilities to access the same content using devices that enable them to interface with the content. See the section on "Accessibility for Web Sites" to learn more about this.

There are other tags used for formatting text including the `<font>` `</font>` tag. This tag uses attributes inside the tag to describe a specific version of the tag. We mentioned another attribute called `bgcolor=""` when we spoke about color. Nearly every tagging element, layout or formatting, has one or more attributes that can be placed inside the start tag to further control the display properties of the tag. The `<font>` tag has three attributes; `color=""`, `size=""`, and `face=""`. The color attribute, `color=""`, defines the color of the text that appears between the start and end tags. You would use the hexadecimal color code described in a previous section.

The attribute that changes the size of the text relative to the rest of the text on a web page is the `size=""` attribute. In this attribute there are two ways to define the size. You can define the font size using the values 1, 2, 3, 4, 5, 6, or 7. The value "3" is the default size used by the browser to display the text in the browser window so defining a value of three will result in no obvious change compared with the text found outside of the font tag. To make the font size smaller you specify a value of "1" or "2." To increase the font size specify a value of "4", "5", "6", or "7" with "7" being the largest that you can increase the font size.

The `face=""` attribute is a little harder to explain, but can be extremely effective when formatting text. The attribute `face=""` refers to the look of the font, in this case the font itself. The font used in this document for the main content is "Times New Roman." The `face=""` attribute is used to make a change in the text font from the default use by the browser for any text located between the font tags. Care must be applied when selecting a font because the font will only appear the way you code for it if the viewer has the same font. For example, I could set the attribute to display the "Arial" font, `face="Arial"`. If the computer I am developing the web pages on has the font, then I will see the text between the font tags changed to Arial when displayed in the browser. If I then post those pages on the Internet and view them using a computer connected to the Internet that does not have the Arial font, then the text between the font tags will not appear different

from the default font used for all of the text. See the section on "A Word about Fonts" to learn more about this.

One of the nice things about attributes is that you can place as many as you want in the same start tag so you do not have to repeat a start tag to apply a different attribute. For example, with the `<font>` tag you can place the attributes `face=""`, `color=""`, and `size=""` inside the one start tag.

```
<font face="Arial" color="#ff0000" size="5"> </font>
```

You cannot place more than one reference to a single attribute in the same tag, however. In essence the browser would be getting two commands for the same thing, and the browser will ignore one attribute and use the other. For example, the code below will result in a red text color and not the blue color specified in the first color attribute.

```
<font color="#0000ff" color="#ff0000"> </font>
```

Text is not a simple matter of Typing

One of the problems with a tagging code for presenting content is that the tag itself is coded for by text. This means that there are a few characters and letters that cannot be typed directly into the code of the HTML file. For example, the "less-than" sign, "<", cannot be represented in text with out coding for the ASCII character. This is because when the browser sees the "less-than" sign, it is triggered to expect a tag. If it does not receive a "greater-than" sign, ">", it will assume all of the text following a "<" is a tag. The only way around this is to represent the "less-than" sign with its code equivalent so that the browser will display it as text. In this case the code for the "less-than" sign is: "`<`". The ampersand tells the browser to interpret the code that follows as an ASCII text character. The pound sign identifies it as a hexadecimal number and the number specifies the character. The whole thing is ended with the semi-colon so that the browser returns to normal interpretation of the content for display.

There are a lot of these special characters and you must code for them with either the hexadecimal code or a name abbreviation that is available for some of the characters. The abbreviation is usually some word or group of letters that when placed between the ampersand and the semi-colon tells the browser what ASCII character to display. See the table below for a list of essential characters that need this special treatment if you want to display them in the content of a web page.

Character	ASCII Code	Abbreviation		Character	ASCII Code	Abbreviation
<	<	<		>	>	>
"	"	"		'	'	'
&	&	&		~	~	

#	#			-	-	­
$	$			_	–	–
%	%			——	—	—
*	*			_	_	
(	(			\	\	
)	)			/	/	
!	!			|	|	
(space)				^	ˆ	ˆ
@	@			`	`	`
+	+			´	´	´
±	±	±		¨	¨	¨
¿	¿	¿		~	˜	
¡	¡	Iexcl		™	™	™
‰	‰	‰		©	©	©
ƒ	ƒ	ƒ		®	®	®
¥	¥	¥		°	°	&degree;
€	€	€		«	«	«
£	£	£		»	»	»
¢	¢	¢		µ	µ	μ
¤	¤	¤		2	²	²
¶	¶	¶		½	½	½

An example of an implementation of the copyright symbol, which is common on web pages, is below:

```
All content is copyrighted and all rights reserved, 2004&#169;
```

One of the great things about ASCII text is that you can code for accented letters that are essential for certain words or the pronunciation of certain words. For example, the word café is accented with a mark above the letter "e" and the word should always appear with an accent. In word processors such as MS Word, the auto-correcting features of the software, like spell check and grammar check, automatically add this accent for me after typing the word. If you are using a different word processor or this option is turned off, you can add the accented "e" by going into one of the start menu and adding or inserting a character or symbol. In MS Word, I use the Insert Menu, select the Symbol Option and the Symbol Window appears. I can then select the unique character to add.

To perform this feat in a page of web content, I must code specifically for the accented character. The sentence, "I ate at a small café in the airport," would be coded as below to appear the same on a web page:

```
I ate at a small caf&#233; in the airport.
```

See the table below for a list of accented letters and the code to use to have them displayed in a web page properly.

Character	ASCII Code	Abbreviation		Character	ASCII Code	Abbreviation
à	à	à		î	î	î
á	á	á		ï	ï	ï
â	â	â		ñ	ñ	ñ
ã	ã	&altilde;		ò	ò	ò
ä	ä	ä		ó	ó	ó
å	å	å		ô	ô	ô
æ	æ	æ		õ	õ	õ
ç	ç	&ccedit;		ö	ö	ö
ð	ð	ð		ø	ø	ø
è	è	è		ù	ù	ù
é	é	é		ú	ú	ú
ê	ê	ê		û	û	û
ë	ë	&emul;		ü	ü	ü
ì	ì	ì		ý	ý	ý
í	í	í		ÿ	ÿ	ÿ

Comment Tags are Essential

Often when coding a web page a section of the code can be commented with text that is not intended to be displayed for the reader. In this case it is important to include a tag around the text so that the browser will know to ignore the text between the tags. This is referred to as "commenting out" text and is performed with the tag `<!--   -->`. Any text placed between the less-than-exclamation-point-dash-dash and the dash-dash-greater-than characters is commented out.

The comment tag can be used for a variety of purposes. First, you should comment the code in your web page HTML document so that you know what sections in the code are for what sections on the web page. This will make it easy to trouble shoot the problems that you are experiencing when the web page does not display properly. You will be able to go directly to the section of code that is the problem. Refer to the section "Example HTML Files" at the end of this document for examples.

You can also place the comment tag at the beginning and end of a section of the web page that you want to remove temporarily. This is very valuable when trouble-shooting a specific line of code or section of code that might not be working properly or might be influencing the document in a strange way. Removing the section with comment tags means the browser will ignore everything in between. Doing this also means you will not have to re-type a lengthy section of code.

You can also use the comment tag as part of several people working on the same pages or web site. The comment tag could direct an individual to place content in a specific place,

other coding elements, or perhaps provide a place holder for future graphics that you do not want to appear yet on the web page.

I often use the comment tag to hide repetitive lines of code that I use in a single page so that I can copy-and-paste them into the web page repeatedly. The advantage is that I will not have to retype the code and I can hide multiple or incomplete examples of code from being expressed by the browser.

Programmers and coders will tell you that it is essential that you mark your code with comment tags. These stand as reminders of the value of a line of code within the program. The same is true for web pages. You might forget why a line of code is placed in the document and the comment tags serve as a reminder. Also, when others take over the job of revising and updating your web page, the comment tags will remain there to explain what, where and how you did something within the web page. Refer to the section "Example HTML File – Advanced" at the end of this document to see examples of this.

A Word about Fonts

The look of a font can be a powerful means of communicating information and facilitating the reader. Studies have shown that reading speed and comprehension increases when using a serif font. A serif font is a font that has frills, enders, tails, serifs, whatever you want to call them, on the letters. The font you are reading here is a serif font called Times New Roman. Arial is an example of sans-serif (without serif) and this sentence has been altered to appear in the Arial font. When looking at identical letters between a serif and a sans-serif font face, you can readily see the serifs that appear in the letters. Researchers agree that the serif part of each letter communicates additional information that improves reading speed enabling readers to skip portions of a letter or word in the decoding process of the brain. The Arial font by comparison is fairly common from one letter to another and slows the readers speed down.

Publishers and researchers have indicated that the content of a passage should be in serif fonts and that headings and sub-heading section titles are best presented in sans-serif fonts where the reader must slow down to read the section titles.

Another important aspect of the fonts we use is that none of the ones we commonly use were developed for the electronic environment. There are now a few fonts that have been specifically designed for the unique nature of the web browser and the computer monitor, which ultimately displays the font chosen. Believe me, not all monitors are created equally, and while the Times New Roman font is the default used by browsers, it does not always look the best. The serif font created specifically for computer monitors is the font "Georgia." The font was developed to display the serif part of each letter on a square pixel computer screen. The sans-serif font developed for electronic display is "Verdana." While it is not necessary to code for these fonts specifically, they are the fonts that will

look the best regardless of the font size chosen when displaying a web page. See the section "Advanced Topics" for more information about setting these fonts for each web page.

The Anchor Tag

The anchor tag, `<a>  </a>` is probably the most significant aspect of the hypertext markup page since it is the anchor tag that makes the link to other content pages. There are many different types of anchor tags or more specifically, there are links to many different web based content and services that the anchor tag provides. There are 4 basic different types of links that the web page coder can put on a web page. These are discussed in detail below.

The link to another web page on the same server

The typical link on a web page is to another web page on your web server or in your web site. This is the simplest link requiring the name of the file or the name of the file and its directory tree structure. For example, if you want to link someone from your home page to your Public Education Classes on a file named "classes.htm" then the link would be:

```
<a href="classes.htm">Public Education Classes</a>
```

If the file was located in a folder called "PE" then the link would be:

```
<a href="PE/classes.htm">Public Education Classes</a>
```

If the file was located above the current location in the directory structure then the link would be:

```
<a href="../classes.htm">Public Education Classes</a>
```

The dot-dot-slash (../) tells the browser to look for a file above its current directory folder. Repeating this code will tell the browser to look up two levels in the directory structure. The analogy that fits best here is that the "../" tells the browser to take one step back and to look in that folder to find the file. Each "../" is a step back. You can step back and forward again, meaning that you can open another folder at a level above your current location by using the following code:

```
<a href="../PE/classes.htm">Public Education Classes</a>
```

Here the browser will step up the directory structure, open the folder "PE" and look for the file "classes.htm." You have already probably surmised that if you do not accurately type the location of the file, or look in the wrong directory structure, then the browser will not be able to find the file.

The link to another web site

The second most common link on a web page would be a link to another web site. The link in the `href=""` attribute needs to specify the protocol of the communication that the browser should expect to find. In the case of the World Wide Web, the first four letters of the Uniform Resource Locator are HTTP which stand for HyperText Transport Protocol. There are many other protocols including File Transfer Protocol (FTP) and HyperText Transport Secure (HTTS). An example of a link connecting the user to another website would be:

```
        <a href="http://www.cgaux.org">The United States Coast Guard
Auxiliary National Web Site</a>
```

The link to an email address is a convenience for the user so that they do not need to re-type the email address, a tiresome chore. Providing an email link is simple, but it must be remembered that the client or visitor must have an email program configured to run when an email link is clicked on in the browser. For example, my computer has Microsoft Outlook configured to start whenever an email link is clicked on in my Internet Explorer browser. If my email was not configured in my browser then it might inform me of the need to configure an email software or prevent the link from working. The anchor tag and the `href=""` attribute should look like this:

```
        <a href="mailto:todd.thuma@gactr.uga.edu">email todd</a>
```

The named anchor tag

The named anchor tag is another convenient aide to provide the visitor to your web site. Some web pages extend below the scroll for several "stops" and providing visitors a way to jump around on that page is great. The named anchor tag performs this service. The `name=""` attribute is used instead of the `href=""` attribute to mark in the document where you want to make the jump. For example, often you may find a "TOP" link in a page which jumps you to the top of the web page. To perform this on your web page, place the following code within the <body> </body> tags nearest to the start tag:

```
        <a name="top"></a>
```

Then place a link somewhere near the bottom of your web page to the named anchor with the following code:

```
        <a href="top">TOP</a>
```

You may have noticed that the `href=""` attribute is used in this case. If you think about the `href=""` attribute as the hypertext reference link, then this starts to make sense. The end result is that wherever the link to top is placed, clicking on it returns the browser window to the top of the web page. This technique is often done for glossaries and other

content that is easily segmented and benefits from having a list of links at the top of a page that jump a visitor to the exact location of that section.

One important point to understand is how the browser prioritizes the scroll. The named anchor tag will place the location of the named anchor at the top of the screen as long as there is content to place at the bottom. The preference is to put the last line of text in the bottom of the window. If the named anchor is only halfway up from the bottom, then that is where the browser places the named anchor, halfway up from the bottom. Experiment with your code to experience this yourself. View the section "Example HTM File – Advanced" for a closer examination of how this named anchor tag works.

A general note about links

All of the mentioned anchor tags have attributes other than the `name=""` and `href=""` attribute. The `title=""` attribute is a way to name the link and refer to it by other code in your document and provides a hover hint when the mouse is rolled over the linking text. It is also a way of communicating information to screen readers especially when using the `target=""` attribute to open the next click in a new browser window or a specific frame. The `target=""` attribute tells the browser where to open the link. Defining the target attribute to be "_top" (`target="_top"`) means that the link will open in the whole, currently active browser. The "_blank" value opens the link in a new window on top of the current browser window. This is typically the command that is used to open pop-ups. The "_self" opens in the current browser window and "_parent" opens the link in the parent browser window and is especially important when working with fames. Below are a couple examples of anchor tags with defined target and title attributes.

```
     <a href="http://www.cgaux7.org" target="_blank" title="link will
open in a new browser window">7th District Auxiliary Web Site</a>

     <a href="classes.htm" target="_self" title="link will open the
Public Education Classes Schedule web page">Public Education
Classes</a>
```

Any text placed between the start and end tag for the anchor is the text that appears as the linkable text. In the examples above, "Public Education Classes" would appear as underlined and blue text. The text "7th District Auxiliary Web Site" would also be in blue and underlined, but the "th" would not appear as a smaller font and superscripted as seen in this document, but it would instead appear as "7th". MS Word did that automatically for me, but the browser will not. If I want the text to appear that way, I have to use either a special character or a series of HTML tags. The series of tags would be:

```
     7<sup><font size="1">th</font></sup>
```

Here the <font> tag is used to reduce the size of the text "th" to the smallest font size. The <sup> tag is the superscript tag and will elevate the text off the baseline. Note how

the font tag is started and stopped within another series of tags, the `<sup> </sup>` tags. This is called nesting and all tags must be nested properly in order to work in the HTML 4.0 and XHTML standards. Below is an example of tags that are not nested properly and will not work correctly when displayed:

```
7<sup><font size="1">th</sup></font>
```

Notice that the font tag is started but not ended prior to ending the superscript tag. To nest you must end (or close) the last tag opened and follow the sequence until all tags are ended. The `<html> </html>` tag is an example of this because all tags for web page coding are nested between the start and end tag.

The Image Tag is really a Link

The image tag is technically a specialized link because images are not technically in the web page. The browser codes for space to be allocated to the image when the web page is drawn and the image is "included" from the source. A simple implementation of the tag is below:

```
<img src="aux_logo.gif" alt="The Official Logo of the United
States Coast Guard Auxiliary" />
```

In this example the image tag `<img />` has a source attribute, `scr=""`, and an alternative text attribute, `alt=""`. The `src=""` and `alt=""` attributes are mandatory, one to display the image and the other to provide a text equivalent for the image in case the image does not load or a screen reader is used. You will also notice that the image tag does not have an end tag and requires a forward slash, "/" just before the greater-than sign.

Images occupy space and in the example above, no dimensions are specified. The lack of specific dimensions will cause the web page to load slowly. This is because the browser must calculate the need for space in the web page prior to drawing the text and the graphics. If the dimensions, height and width are specified as attributes, then the browser is told how much space to allocate and will actually draw the text of the web page while the images download to the user. The code below is an example of a more complicated image tag:

```
<img src="aux_logo.gif" width="20" height="20" alt="The Official
Logo of the United States Coast Guard Auxiliary" />
```

The `width="20"` attribute and the `height="20"` attribute specify the number of pixels that the graphic will occupy in the horizontal and vertical directions. Just about every image viewing and editing program will have the ability to tell you the size of the graphic. Leaving this attributes blank means that the browser will get this information from the image itself.

There are a couple of other important attributes that can be defined here. The `border=""` uses pixels, expressed as integers placed between the quotation marks in the attribute, to

define the amount of border thickness to place around the graphic. The default is zero or none and can be manually set with `border="0"`.

The space reserved around the graphic, the space where the text or other objects would go, can be defined in attributes as well. The `hspace=""` sets a empty space around the graphic in the horizontal dimension and the vertical dimension of space is defined with the `vspace=""`. There are other attributes that are outlined in the section, "HTML Code Listing."

Different Image Types

There are different types of images and each file format is specific for a particular type of image. The *.gif file format and the *.jpg file format where the first formats capable of being decoded by the browser. The *gif file format was created by CompuServe back in the late 80's early 90's. They patented the file compression format but never pushed for the royalties that were owed to them and the *.gif file format became the main graphic format for viewing on the Internet. The *.jpg file format came out in the middle to late 90's and was an attempt by the Joint Photographic Experts Group (JPEG) to create a compression scheme that did better on photographic style images than the *.gif format was doing. They succeeded in creating an open source standard that exists today as the best means of compression for photographs. The *.gif compression format is best for cartoon, line drawing, and simple colored images.

Unfortunately, patents and intellectual property are causing the *.gif image compression format to die off. The company that bought CompuServe realized that they could profit from the patents that they now owned and spent the last couple years in court trying to get a legal ruling that would allow them to recoup royalties from companies using the *.gif compression standard. They achieved this ruling in the early 2000 but unfortunately the ruling is a little like mandating that the barn door be closed even though the horses are long gone.

The replacement, *.png, is better anyway and is an open source standard that has no patent or royalty baggage associated with it. The *.png file format stands for Portable Network Graphics and browsers already have the code necessary to decode the file format and display that in the browser window. The *.png format is best for *.gif style images meaning line drawing, simple colors, and cartoon or clip art style graphics. Basically anything that is not a photograph is best to be compressed into a *.png file.

Another file format on the horizon and will be important in the near future is the *.svg file format or Scalable Vector Graphics. The advantage of a scalable vector graphic is that it can be scaled up or down without a loss of quality. The file format will even allow animation of the graphic on a web page and morphing into other shapes and colors. This is an open source standard to what can be done in the Macromedia Flash software and when *.svg compression is built into programs like Microsoft's PowerPoint or Adobe Photoshop you will see widespread acceptance in the web community. Both the

Netscape 7.0 and Microsoft I.E. 6.0 browsers can read *.svg files without adding a plug-in.

Tables Arrange the Text

Tables present an easy and efficient means of laying out text on a web page. Defining columns, rows, and cells make the work of laying out text both beautiful and organized. Unfortunately, this is true for only the sighted reader. A screen reader has a terrible time navigating the code to get at the content, and the use of table tags for content layout, except when laying out data in a tabular form, is deprecated. As we move towards separating content from the formatting and layout it is important to eliminate the use of table tags for the purpose of layout text, unless the information like data is suited to a table. The good news is that table tags are allowed under the 508 Guidelines as long as you indicate that the table is for layout purposes only. The section below illustrates both uses of the table, one to layout tabular data and the other for layout elements on a page.

A Table of Data

The table tag, `<table> </table>` is a straight forward tag with a several attribute tags. It starts the table and every tag that is presented in this section must fall between the start and end tags. See the code below followed by a detailed discussion:

```
<table align="center" width="80%" border="1" cellpadding="3"
cellspacing="2" summary="this table displays data representing the
mission and the number of hours engaged in that mission for the
operational year 2001">
<thead>
<th width="80%" align="center">
Mission
</th>
<th width="20%" align="center">
Hours
</th>
</thead>
<tbody>
<tr>
<td width="80%" width="left">
Operations: Surface
</td>
<td width="20%" width="center">
250
</td>
</tr>
<tr>
<td>
Operations: Air
</td>
<td>
148
```

```
</td>
</tr>
<tr>
<td>
Public Education
</td>
<td>
221
</td>
</tr>
</tbody>
</table>
```

From the beginning you can see that several attributes are defined in the table start tag. The `align=""` attribute defines whether or not the table is placed in the center of the page or in the left or right margins. The `width=""` attribute defines the size of the table and can be done in either percent or pixel widths. Percent is preferred and here the example is 80% of the window width. Greater control can be defined by assigning a pixel width such as 640 pixels or 800 pixels. I always design for an 800 by 600 screen resolution, which means the screen is 800 pixels by 600 pixels. I typically define my tables to be no more that 750 pixels wide.

The `border=""` attribute tells the browser to place a border around the table and define each cell with a border. The default value is 0 and the value can be set fairly high, but a border thicker than about 5 looks bad. The `cellspacing=""` and `cellpadding=""` attributes define the space between cells and the space within a cell on all sides before text is displayed, respectively. The `summary=""` attribute is also important as that is where you tell screen readers and other special browsers what the table is used for.

The table is defined into sub-sections, the head (`<thead> </thead>`) and the body (`<tbody> </tbody>`). Inside the `<thead> </thead>` tags is the `<th> </th>` which is the head of each column. This table is 2 columns because I defined two `<th> </th>` combinations. Had I specified a third, then there would be a third column. Be careful to specify no more than 100% of the tables width or if using pixels, define only enough to equal the pixel width of the table as a whole. In either case you cannot excide the specified width of the table. The alignment of each cell can be specified too or in the first instance of each column as seen in the code above.

The rows in a table are offset with `<tr> </tr>` tags and the table data tags, `<td> </td>` are placed inside these. Use the same number of `<td> </td>` as you used in the `<th> </th>` or the table may not display in the browser. There are a few tricks that you can employ if you want to combine or eliminate cells, but that is reserved for an advanced section on the topic.

Here too in table data tags you can define the alignment attribute as well as the width. Note that once the first series of `<td> </td>` tags are assigned attributes, it is not necessary to repeat that code unless something changes later in the table. The text that you want to appear in the table is placed between the <td> </td> tags and you can place

any tags there that you would use outside of the table including <p> </p>, , , among others.

Here's what the table should look like in a web browser:

Mission	Hours
Operations: Surface	250
Operation: Air	148
Public Education	221

One attribute not seen in the code above is the `bgcolor=""` attribute that allows the color of the cell, row, or table to be specified. You can place the `bgcolor=""` attribute in the `<table>`, `<tr>`, `<th>`, or the `<td>` tags. Using the hexadecimal code for the color desired you can define the background color.   You should never use color as part of the information that you are communicating on a web page, because color is not consistently applied in every browser, the importance of color is lost on screen readers, and color-blind individuals may not see the colors you intended.

Laying it all out with Tables

Using table tags to layout your web page is a great way to maintain control over where text, graphics, and other objects are placed. Keep in mind this is no longer a preferred means of laying out information on a web page, but the newer method used to layout elements on a web page is an advanced topic reserved for a later discussion. I often place a header at the top of the page followed by a navigation menu to the left and the body of text on the right. See the code for this below and compare it with the table used for data layout.

```
<table width="600" align="left" summary="the table is used for layout
purposes only and lays out a page header on the top with a navigation
menu on the left and a main body of text on the right under the
header">
<tr>
<td colspan="2" width="600" title="header">
<img src="header.jpg" width="600" height="100" alt="a graphic header of
several images from missions of Flotilla 28 in the Bay of Volunteer" />
</td>
</tr>
<tr>
<td width="150" align="left">
<a href="home.html" target="_self" title="link to the home page"><img
src="home.gif" width="150" height="80" alt="home" /></a><br />
<a href="operations.html" target="_self" title="link to the operations
page"><img src="operations.gif" width="150" height="80"
alt="operations" /></a><br />
<a href="pe.html" target="_self" title="link to the public education
page"><img src="public_ed.gif" width="150" height="80" alt="public
education" /></a><br />
```

```
<a href="vsc.html" target="_self" title="link to the vessel safety
check web page"><img src="" width="150" height="80" alt="vessel safety
checks" /></a><br />
</td>
<td>
<p>
Welcome to the web page of Flotilla 28, America's Volunteer Lifesavers
in the Bay of Volunteer.  Explore the links in the navigation menu on
the left to learn more about the U.S.  Coast Guard Auxiliary and our
commitment to your safety on the water.
</p>
<p>
Just in case you were wondering, the USCG Auxiliary is part of the
Department of Homeland Security, too.  As members of this department we
are ever diligent to the security of assets on the water.  Follow this
link <a href="http://www.dhs.gov" >here</a> to visit the Department of
Homeland Security.
</p>
</td>
</tr>
</table>
```

The code above defines a table with 2 rows and 2 columns with the first row occupying both cells in the row. The `colspan=""` attribute in the first `<td> </td>` tag merges the two cells that would have been here on this row. The attribute `colspan=""` literally tells the browser to have the cell, defined by the `<td> </td>` tags to span the number of columns specified in the attribute. The same trick can be applied across rows with the `rowspan=""` attribute. Be certain to omit the extra `<td> </td>` tags nested with in subsequent <tr> </tr> tags to prevent to many cells from appearing within a row. If the table looks funny or has extra, compressed cells at the left or right sides of the table, this is typically because of too many cells being defined for a row.

The `<img />` tag codes for the header graphic and defines the alternative text that will appear in the first row of the table.

The next row is divided into two columns, a left navigation menu and the main body of text. The navigation menu in this case is 4 image buttons where an image is made to be a link by placing it inside the `<a>` tag. Because the links are images be certain not to forget the `alt=""` attribute so that the screen reader will indicate which button links where.

The `<br />` is a line break tag that has no end tag and therefore requires a forward slash in the back half of the tag. The line break tag will force the next graphic to be on a new line. This is necessary because the browser likes to make decisions about where to place graphics and text and it might "accidentally" place two image buttons side-by-side rather than one on top of the other. The `<br />` tag is a way to force the line break.

The main body of text in the last cell in the second row is fairly straight forward. Any tags can go here and in the example above, two paragraph tags are seen. Basically, the table coded for above should look like the table depicted below. Note that the table below has a "weak" border, but the table in the code above would have no border. The

MS Word program places a "weak" border line around the table as seen below, despite the table lines being turned off. When the document is printed the table seen below in the printed document will not have a border. Also the graphics were not presented in the table below so that you could attend to the nature of the table and not worry about the graphics. The text stands in for the graphics that would be placed there.

Graphic Header (not shown)

Home	Welcome to the web page of Flotilla 28, America's Volunteer Lifesavers in the Bay of Volunteer. Explore the links in the navigation menu on the left to learn more about the U.S. Coast Guard Auxiliary and our commitment to your safety on the water.
Operations	
Public Education	
	Just in case you were wondering, the USCG Auxiliary is part of the Department of Homeland Security, too. As members of this department we are ever diligent to the security of assets on the water. Follow this link here to visit the Department of Homeland Security.
Vessel Safety Checks	
(graphical buttons, not shown here)	

Hexadecimal Code

There is nothing magical about hexadecimal code. The numbering system of 0 to 9 is a system based on 10 digits in the "ones" column before starting into the "tens" column. So the number 10 represents one value in the "tens" column and one value in the "ones" column. The hexadecimal system uses 16 values in the "ones" column. Since 10 is a two digit value, we need someway of representing the 10th, 11th, 12th, 13th, 14th, 15th, and 16th values in the "ones" column. Since we start the numbering with 0, the value 9 is actually the 10th value in the hexadecimal system. So we use the letters A, B, C, D, E, and F to represent the 11th, 12th, 13th, 14th,15th, and 16th values in the hexadecimal system respectively. So the hexadecimal system is number as: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, & F.

The basic reason for this is that computers work in bytes which are 8 bits of information, ones and zeros. Each byte codes for a character that can be represented on the computer screen. With respect to the web browser and color, the color of any object in the browser is defined by the hexadecimal numbers described above. The attribute `color=""` and `bgcolor=""` use a hexadecimal coding system to define the color for the attribute. The variable placed in quotes must start with the # (pound sign). Following the # sign is two hexadecimal digits for Red, two for Green and two for Blue. Coding for a level of red, green, and blue you are able to reproduce over 25,000 different colors. This is because you are actually telling the monitor what value of red to apply to the screen as well as the

value of green and blue to apply. By adjusting the levels of the respective colors, you can create the other colors in the spectrum; orange, purple, maroon, cyan, magenta, etc.

The two hexadecimal digits actually are multiplied by the computer to obtain the color value of each color segment. For example a color value for red only, #ff0000, would mean that the red is turned on all the way to highest value of 256. Each "F" represents a value of 16 and 16 multiplied by 16 equals 256. In Photoshop you might have noticed that color can be defined in values of 0 to 255. Well, if you count 0 as the first value, then the 256th value is 255. To achieve the color of black you must turn all of the colors to off or #000000. To set the color to white you turn all of the colors to full on, #ffffff. A small color table below illustrates the color possibilities.

Color Table

One important point to state about color is that while your web browser can display between 25,000 and 16 million colors, this also depends on your computer monitor, the human eye can recognize about 1000 distinct colors and shades of a color. Early browsers could only differentiate between 216 colors. You might even see the term "web safe" colors which refer to the 256 colors that every browser can display regardless of computer platform.

Rule of 51's

There is an easy way to remember the 256 web safe colors that can be coded for within a browser that will reliably and faithfully be displayed on all computers equally. That is by remembering the rule of 51's and its relationship to the hexadecimal code. If you start with zero and add 51 for each step you get 6 steps as your reach 255. If you related that to the hexadecimal code, then a pattern emerges. The first value of 0 corresponds to 00 in hexadecimal code. The next step, 51, corresponds to 33 and 102 equates to 66. The small table below illustrates this relationship.

Rule of 51's	
Step	Hexadecimal
0	00
51	33
102	66
153	99
204	CC
255	FF

By combining the 6 possible hexadecimal values in the "Rule of 51" across the red, green, and blue color assignment, you can achieve 216 possibilities. So what happened to the other 40 colors? It turns out that the different browsers, Netscape and Internet Explorer the two more popular ones, on different platforms deal with those additional colors differently. Suffice it to say that it gets more complicated to use those 40 other

colors, so just stick with the 216 that the Rule of 51's can produce. In essence what this means is that you can only use the following combined hexadecimal values in your color attributes: 00, 33, 66, 99, cc, or ff.

Accessibility for Web Sites

One of the great things about content available from the Internet is that it can be made available to all people regardless of their learning or physical abilities. A published physical book cannot meet that goal, because a different version of the book would need to be produced to meet everyone's unique situation. A blind person with a screen reader, however, can benefit from the content placed online without the developer having to do anything different for the content. The key is that the developer not place code "in-the-way" of the screen reader so that the visually challenged visitor to the web page is not frustrated when attempting to get at the content on the page.

What this basically means is that some of the tags that the web page developer chooses to use can place blocks in the way of the screen reader. This gets back to the topic that content should be separate from the format and layout of the content. As a result, some of the code for layout and formatting text on a page has either been moved off the content page or has been changed to code that is more friendly towards browsers and devices used by visually changed visitors.

The government was challenged that the American Disabilities Act, a law which mandated physical accessibility and other equalities providing access to structures and government services for all people, extended to government web sites. Since the Auxiliary is part of a government entity, our web sites must meet the ADA guidelines for accessibility. The guidelines were added to the law and are known as the 508 Guidelines for their section in the law.

The 508 Guidelines are fairly simple to follow and implement. The difficulty comes from reading an understanding these guidelines and implementation with your web pages. Web page editors, software used to generate web pages without the hand coding of HTML, vary in their ability to implement the accessibility requirements.

A short list of 508 Guidelines is below:

Advanced Topics

Setting the font face to Verdana or Georgia is a simple matter complicated by the number ways that it can be performed. One method is to use the `<font>` tags to define the `<font face="">` for the text between the start and stop tags. Unfortunately, this method is deprecated and strongly discouraged. Not to fear, however, there are other ways of performing this task with varying degrees of difficulty.

The technique common to all the methods is called style sheeting. You can apply the style sheet as an inline element, as an attribute in the layout element, to the whole body, or as a cascading style sheet. Using the `style=""` attribute you can define the font face, color, and size among other characteristics, using the tagging element `<span> </span>`. The span element defines the span of application of the attributes described in the style="" code. For example, if I wanted to switch from the default values of the web page from Times New Roman and black to Arial and red text I would use the following tag:

```
<span style="color:#ff0000; font-family:Arial;"> </span>
```

All of the text placed between the start and end tag will be red in color and the font face, or family, will be Arial. Note that the style sheet defines the property and then separates the value of the property with a colon. Also notice that multiple properties can be defined by separating their call for expression with a semicolon. The `style=""` attribute is the sole container defined by the start and end double quotation marks. There are no spaces between the property and the colon and the value applied to the property. The `<span>` element is an inline element that changes only the text within the start and end tag. There are other uses for the `<span>` element discussed elsewhere in this document.

The paragraph tags, `<p> </p>`, can also be used to define how an entire paragraph of text will appear. This is important because the span element is intended for short sections of text not an entire paragraph. An example might be to use the span element when you need to change the color of one word in the paragraph versus setting the entire paragraph of text such as in a quoted passage from a book. An example of this is below.

```
<p style="color:#000000; font-style:italic; font-family:Verdana,
sans-serif; font-size:larger ">
```

The style attribute properties above define a paragraph that will be black text in color, be italicized, use the Verdana font if available and if not use the default sans-serif font, and will be larger in size than the default font. Below is a short list of properties that can be applied to text.

Style Property	Optional Values
font-family	Arial, Helvetica, Times, Courier, Verdana, Trebuchet, Georgia, Comic Sans, serif, sans-serif, monospace
font-size	xx-small, x-small, small, medium*, large, x-large, xx-large
font-style	normal*, italic
font-variant	normal*, small-caps
font-weight	normal*, bold
Color	#rrggbb, #000000*
text-align	left*, center, right
text-decoration	none*, line-through, underline

text-indent	number of pixels in integer values, only works on the first line, second line not indented, default is 0
*default value	

If you defined the property values for an entire paragraph of text and needed to define a short phrase differently from the paragraph, then you would use the span element to change from the values set by the paragraph to the property values for the specific word or phrase inside that paragraph.

To identify the style attribute for an entire HTML document you can specify the style attribute in the body tag or you can define the styles the style element tag, `<style>` `</style>` and place those start and end tags nested in the `<head>` `</head>` tags. An example of the first option is identical to the paragraph style attribute described above. The `<style>` `</style>` tag is a little different and requires a different attribute in the element tags in order to refer to the style attribute. The short example below is taken from the section "Example HTML File – Advanced."

```
<style type="text/css">
      .quote {color:#000000; font-style:italic; font-
family:Verdana, sans-serif; font-size:larger; text-indent:20px; }
      .title {color:#0000ff; font-weight:bold; font-
family:Verdana, Arial, Helvetica, sans-serif; text-
decoration:underline;}
      .normal {color:#000000; font-weight:normal; font-
style:normal; font-family: Georgia, Times, Time New Roman, serif;
}
      .important {font-weight:bold;}
</style>
</head>
<body>
<h1 class="title"> From the Operations Officer</h1>
<p class="normal">I would like to start this section on the
importance of training with a quote from the admiral.</p>
<p class="quote"> The key to success on a mission is <span
class="important">training</span>, because it is training that
prepares us for the inevitable emergency that through <span
class="important">diligence to training</span> comes
infrequently.  </p>
```

All of this code above enables us to examine several key aspects of style sheeting. The call for the styles starts in the style element. Here the start and end tag delineate the classes that will be called to by individual element and the `class=""` attribute. Classes are nothing more than a set of properties pre-defined in the web page. This eliminates a great deal of repetition in your code from layout element to layout element because you can simply refer to a set of properties, in this case a class, than have to re-type the string of properties and values that make up the style.

The style element has a `type=""` attribute that must be defined, but it is always the same `type="text/css"`. The class is defined by starting with a period and following with a unique name. In the example above the classes are "quote", "title", "normal", and "important". The brackets, {}, contain the properties and the values that define the class. Any number of properties can be defined here but the same property can not be defined twice.

The elements place between the body start and end tags must call to the head section using the `class=""` attribute. In the `<h1> </h1>` tag the class attribute is defined as `class="title"` and the text will appear blue, bold, and appear in sans-serif font. The class prioritizes the font family defining Verdana first, Arial second, and Helvetica third. If none of these fonts are available, then the default sans-serif font available on the client's computer will be used. The first paragraph uses the `class="normal"` specification where the text appears black, non-italicized, normal weight, and will appear in the font family Georgia, Time, or Times New Roman before using default serif font. This step is not necessary if you want to stay with the page defaults or you can define the page defaults in the body tag with the `style=""` attribute.

The quote class will italicize the text, specify black for color, use the Verdana font if available and indent the first line of text 20 pixels. The span element with the `class="important"` will do something unexpected with the text that falls in between the span tags. The text will no longer be Verdana or be italicized because the class, "important" does not specify the text the way that the `class="quote"` does. This is because the span element placed within the paragraph element will use all default values for the web page unless specified in the code. The paragraph gets is class coding from the `class="quotes"` specifying italicized and a special font. Both deviate from the default values of the document. So when `class="important"` is employed in the span element, the span element trumps the paragraph tag class attribute.

The final method of style sheeting is to not place the class definitions in the web page at all. You place the definition for each class in a separate file and call to that file from the web page documents that you create. This means that instead of repeating the classes in the style element, `<style> </style>`, of each header in each document you define it only once in a Cascading Style Sheet. As your web site grows you will really appreciate what this means because only one file, the *.css file, will need to be changed to change the format of all the web pages in your web site. Without the *.css file, you would need to change dozens, even hundreds of documents to make changes to the format of each document.

To complete this step you place all of the classes defined between the `<style>` `</style>` tags in the example above and put that code in its own text document saved with the *.css file extension. You must then place a `<link />` element in the `<head>` `</head>` region. There are several essential attributes that must be placed and defined there in the `<link />` element. The `rel=""` attribute must be defined as stylesheet thusly; `rel="stylesheet"`. The type attribute is also used: `type="text/css"`. Finally, the `href=""` attribute is used to define the location of the *.css file.

Example HTML File – Simple

```
<html>
<head>
<title>
```
U.S. Coast Guard Auxiliary District 7 Web Site Homepage
```
</title>
</head>
<body bgcolor="#cc33ff">

<!-- Display Header at the top of the web page -->
<h1>
```
United States Coast Guard Auxiliary
```
</h1>

<h2>
```
District 7
```
</h2>

<!-- main body of text on web page -->

<p>
```
The 7th District of the U.S.C.G. Auxiliary encompasses Georgia, South Carolina, and Florida with the exception of a portion of the Florida panhandle and a bite out of western Georgia around the city of Columbus. Aside from the continental United States, the 7th District Auxiliary also includes the U.S. Virgin Islands and the islands that comprise Puerto Rico. Members in this District patrol the waters of the Gulf of Mexico, the waters around the Florida Keys, the waters surrounding U.S. Protectorates in the Caribbean, the Atlantic Coast as far north as the state border with North Carolina, and hundreds of lakes in these areas.
```
</p>

<p>
```
There are 16 Divisions in the 7th District and the 1st Division has the distinction of being located in Puerto Rico. The 2nd Division is land-locked in north and northeast Georgia, central Georgia, and southwest South Carolina. Division 12 is also land-locked in central and southwestern South Carolina, with the coastal South Carolina and Georgia border being designate the 10 Division. With the exception of the 16th Division which organizes the Flotillas in the U.S. Virgin Islands, all of the other Divisions are spread around the state of Florida. Most are on the coast, but some include Flotillas that patrol the lakes of Florida. The 7th District has over 6,000 Auxiliarists in this region and ranks as one of the largest District in the Coast Guard.
```
</p>

<p>
```
Our members perform many functions as part of Team Coast Guard. We conduct public education programs that teach adults and children about safe boating and environmental practices while on the water. We conduct free boating safety checks to educate the boater about necessary safety gear and safe practices of boating operation. Some of our

Auxiliarists fly their own aircraft in support of homeland security, port over-flights, and coastal safety patrols. While on the water, Auxiliarists provide safety patrols, verify the presence, position, and condition of Aids to Navigation, assist boaters in immediate need, patrols areas that the active duty Coast Guard might not frequent. Both the air assets and the surface assets of the Auxiliary provide additional search and rescue capabilities when the Coast Guard is called to rescue stranded boaters or downed planes. Auxiliarists also man some of the Coast Guard bases providing for administrative support or stand radio watches. In the area of Marine Safety, Auxiliarists work alongside the active duty personnel in the Coast Guard examining commercial vessels for safety and compliance with the laws of the United States. Just as the Coast Guard is tasked with multiple duties and assignments involving the navigatable waters of the U.S., the Auxiliary is there standing with the active duty Coast Guard personnel in support of their many missions.
```
</p>

<!-- disclaimer section -->

<p align="center">
<font size="1" face="verdana"><em>This web</em></font>
</p>

</body>
</html>
```

Example HTML File – Lists

```
<html>
<head>
<title>
```
U.S. Coast Guard Auxiliary District 7 List of Duties and Definitions
```
</title>
</head>
<body bgcolor="#cc33ff">

<h1>
```
Duties and Definitions
```
</h1>

<h2>
```
Duties
```
</h2>

<ul>
<li>
```
Education
```
</li>
<li>
```
Marine Safety
```
</li>
<li>
```
Safety Patrols
```
</li>
<li>
```
Search and Rescue
```
</li>
<li>
```
Administrative Support
```
</li>
</ul>

<hr align="right" size="10" width="50%" color="white" />

<hr color="#ffffff" />

<h2>
```
Definitions
```
</h2>

<!-- alphabetical list of terms and definitions used in this document -
->
<dl>
<dt>
```
Education
```
</dt>
<dd>
```

Education involves the passing of information from an experienced individual to an inexperienced individual often involving skill demonstration or information exchange. Typically the novice is judged by the expert in a manner of examination or testing involving practical skill assessment or written evaluation of knowledge and understanding.
</dd>
<dt>
Marine Safety
</dt>
<dd>
The marine environment is often ad odds with development and progress of humans. Maintaining our precious resource from polluters and those intent on profiting from practices that will render the environment unusable by others is the focus of the Marine Safety Offices in the U.S. Coast Guard. The U.S. Coast Guard monitors pollution clean-up in the navigatable waterways of the United States and their protectorates. The MSO Office also examines commercial vessels for compliance with safety of their cargo and passengers when in the navigatable waters of the United States and their Protectorates. The MSO Office also assists U.S. Customs and Port Authority agencies with the monitoring of cargo importation at the dozens of ports in the U.S.
</dd>
<dt>
Safety Patrols
</dt>
<dd>
Safety Patrols is performed on the water or in the air and generally involves monitoring a set area in the event of an emergency or a call for assistance. The vessel or aircraft during a standard safety patrol may discover a boater or a boat in trouble and can render assistance. Or perhaps the patrol asset while underway may be called to aid a boater that has called the Coast Guard in distress using a VHF radio. These safety patrols are likewise able to discern when conditions or situations deviate from the norm and can report these instances back to the department of Homeland Security.
</dd>
<dt>
Search and Rescue
</dt>
<dd>
The hallmark of the Coast Guard is the search and rescue mission. Either through the forces of nature or from a boaters failure to know or practice safe boating techniques, a boater may become distressed and need immediate assistance. The Coast Guard will attempt to locate vessels in distress and render assistance after locating a vessel in distress. The practice of search and rescue requires consistent training to become practiced and efficient at skills and techniques refined over many years of use. The constant need for training and practice keeps the U.S. Coast Guard Auxiliary ready to follow the example of our motto Semper Paratus, Always Ready.
</dd>
<dt>
Administrative Support
</dt>
<dd>

The Auxiliary is a force multiplier for the active duty Coast Guard personnel. While our members are not typically the young men and women the comprise the Coast Guard, their experience and skill enable them to provide support at many levels of Coast Guard operations. From standing watch in the radio room to processing many of the daily and necessary documents inherent in the U.S.C.G organization, an Auxiliarist can free a younger enlisted person or officer to perform more active or specialized skills. Even if an Auxiliarist can take over the grounds keeping duties of a small boat station, the active duty Coast Guard can spend more time on the water protecting the navigatable waters of the U.S.

```
</dd>
</dl>
</body>
</html>
```

HTML Code Listing

<html> </html>
- Starts and ends a web page. Everything placed between these tags will be interpreted by the web browser

Attributes
- xml:lang=""
  o human language of the document
  o for English the attribute would be: xml:lang="en"
  o this attribute can be omitted
  o this attribute is used in other tags
- xmlns=""
  o XML-namespace
  o This attribute can be omitted
  o This attribute is used in other tags

<head> </head>
- Delineates the head of the web page
- Between the head tags goes the element tags <meta>, <style>, <title>, <script>, and <link>.

Attributes
- profile=""
  o unknown
  o this attribute can be omitted
  o this attribute is used in other tags

<body> </body>
- Delineates the body of the web page
- Most of the HTML element tags available to the web page developer go between these tags

Attributes
- bgcolor=""
  o establishes the background color of the web page, default is white
- link=""
  o establishes the color of the hypertext links on the web page, default is blue
- vlink=""
  o establishes the visited link color on the web page, default is purple
- alink=""
  o establishes the active link color on the web page, default is red
  o this is the color the link takes when you click on hold on the link
- text=""
  o establishes the color of the text in the web page
- background=""
  o defines the background image to be displayed on the web page

o   if defined this will supersede the bgcolor attribute

\<p\> \</p\>
- separates content on a web page into paragraphs
Attributes
- align=""
  o   determines how the paragraph is to be aligned on the web page
    ▪   options: left, center, right, justify

\<h1\> \</h1\>
- defines section headings or headlines within the web page
- there are 6 different pre-defined section headings numbered from 1 to 6 in decreasing level of prominence with respect to the standard content on the web page placed between paragraph tags
Attributes
- align=""
  o   determines how the section heading is aligned on the web page
  o   options: left, center, right

\<ul\> \</ul\>
- generates an unordered list of items
- between the start and end tag goes the \<li\> \</li\> tags
Attributes
- type=""
  o   defines the type of bullet that the list will display, default is a solid circle
    ▪   options: disc, circle, & square

\<ol\> \</ol\>
- generates a numbered list of items
- between the start and end tag goes the \<li\> \</li\> tags
Attributes
- type=""
  o   defines the list style type of the numbered list
    ▪   options: 1, a, i, A, & I (roman numerals)
- start=""
  o   defines the value to start with
    ▪   options: depends on the type of style selected

\<li\> \</li\>
- defines the list item in either ordered or unordered lists
Attributes
- type=""
  o   defines the list style type of the numbered list
    ▪   options: 1, a, i, A, I (roman numerals), disc, circle, & square

- cannot contradict the "type" defined in the ordered or unordered list tag
- value=""
  - defines the value or bullet displayed in front of the list item
    - options: depends on the type of the style selected

<dl> </dl>
- definition list is a specialized form of list that comes with tags that define the items in the definition list
- between the start and end tags goes the tags <dt> and <dd>
Attributes
- compact
  - places the definition of the term on the same line if the definition of the term is short

<dt> </dt>
- defines the term to be defined in the list
- must be placed between the <dl> </dl> definition list tags
Attribute
- nowrap
  - prevents the line of text comprising the definition term from wrapping
  - if included, nowrap is on, to turn off omit from the tag

<dd> </dd>
- defines the definition of the term
- must be placed after the definition term to which it applies and within the <dl> and </dl> tags
Attributes
- nowrap
  - prevents the line of text comprising the definition from wrapping
  - if included nowrap is on, to turn off omit from the tag

<hr />
- places a horizontal ruler on the web page
- inserts a line break of space above and below the horizontal ruler position
Attributes
- align=""
  - aligns the position of the horizontal ruler from left to right
  - options: left, center, right
- size=""
  - determines the thickness of the horizontal ruler
  - any integer value greater than 0 in pixels
- width=""
  - determines the number of pixels from left to right that the line is drawn
  - any integer value greater than 0 in pixels
  - or a percentage of the window width greater than 0%

- noshade
  - o prevents a shadow from appearing behind the horizontal ruler
- color=""
  - o determines the color of the horizontal ruler displayed (Internet Explorer only, will not appear in the Netscape Browser)

- formats the text appearing between the start and end tags
- this is no longer the preferred method and has been deprecated, although it still works in current browsers
- Attributes
- color=""
  - o defines the color of the text
  - o use hexadecimal numbers to substitute for red (rr), green (gg), and blue (bb) in "#rrggbb"
- size=""
  - o defines the size of the text
  - o value options include
    - ▪ 1, 2, 3, 4, 5, 6, 7
    - ▪ 3 is the default size of text in browser window
    - ▪ or +, -, 2+, 2-, etc.
- face=""
  - o defines the font of the text
  - o default is "Times New Roman," a serif font
  - o options
    - ▪ serif fonts
      - • Times New Roman
      - • Georgia
      - • Times
      - • Tahoma
      - • *Others*
      - • serif
    - ▪ Sans-serif fonts
      - • Arial
      - • Verdana
      - • Helvetica
      - • Geneva
      - • *Others*
      - • Sans-serif
  - o The client must have the font on their computer in order to display the specified font defined in the font tag. If the font is not available, then the browser defaults to the font family, serif or sans-serif, if specified of the client's browser.

- inserts an image into the web page as the web page is drawn by the browser

- Note: image files must be placed in a location where the web page can access. While text is coded for in the web page with the HTML code, the image is not included into the code except for the reference to the link. This is different than a word processed document where the image is incorporated into the document and the image file itself does not need to be included along with the document. Web pages by themselves do not contain the image, so the file must be included on the computer, disc or disk along with the HTML file.

Attributes
- src=""
  - technically src or source is an attribute of the image tag. Place the location of the file in the src="" attribute. This will accept a URL enabling a link to another web server but its practice is discouraged since this linked-to server will be overburdened with image delivery.
- width=""
  - defines the width of the image
  - leaving this attribute out means that the browser will not know how much space to allow around the image. The browser must upload the image before preceding with text presentation. If the image width is defined then the browser is able to allot the space necessary for the image and the end result is that the text of the document will appear in the browser window before the graphics have completed uploading. This means that the web page uploads to the client faster. Height must be included to work properly.
  - Values are presented in pixels
- height=""
  - defines the height of the image
  - leaving this attribute out means that the browser will not know how much space to allow around the image. The browser must upload the image before preceding with text presentation. If the image height is defined then the browser is able to allot the space necessary for the image and the end result is that the text of the document will appear in the browser window before the graphics have completed uploading. This means that the web page uploads to the client faster. Width must be included to work properly.
  - Values are presented in pixels
- border=""
  - defines the thickness of the border that appears around the image
  - a value of "0" turns the border off and is the default value.
  - Values are presented in pixels
- alt=""
  - allows for a text description of the image and is required as part of accessibility guidelines
- hspace=""
  - defines the horizontal space around the image before text is presented
  - values are presented in pixels
- vspace=""

- o defines the vertical space around the image before text is presented
- o values are presented in pixels
- align=""
  - o defines how text is aligned with the image. There are many options and this attribute is deprecated in favor or a better method.
  - o Options
    - ▪ Left
    - ▪ Right
    - ▪ Top
    - ▪ Texttop
    - ▪ Absmiddle
    - ▪ Middle
    - ▪ Bottom
    - ▪ Baseline
    - ▪ Absbottom
- usemap=""
  - o used in creating an image map that allows different regions of the image to link to different locations
  - o place the URL in between the quotes and can link directly to the image map code in the document with "#map-name".
  - o Advanced Technique
- ismap=""
  - o defines the regions of the image map. This code is used to define regions of an image through shapes and x and y coordinates. Must be listed in the code with a map name.
  - o Advanced Technique
- longdesc=""
  - o for accessibility reasons, long description (longdesc) was included as an attribute so that lengthy descriptions of images could be created as a file and linked to by this attribute
  - o useful for describing an image in detail but not having to place it in the alt attribute.
  - o Place the URL to the long description file between the quotes

- Formats the text between the start and end tag into a bold font
- replaces the tag

- Formats the text between the start and end tag into an italicized font

<blockquote></blockquote>
- A layout element that is similar to the <p></p> tag but indents the paragraph to the right

- a layout element for text that codes for a line break

\<pre\>\</pre\>
- a text format and text layout element called pre-formatted meaning that the format comes from the way it is formatted in the HTML File.
- text that appears between the tags will appear as formatted in the HTML File even to the point of no carriage returns
- great for making text appear as it is typed by a typewriter or controlling where the lines breaks occur

\<tt\>\</tt\>
- a text format element similar to the pre-format tag and stands fro "typed text"
- text between the tags appears as typed text and the window size determines the carriage returns

Table Elements

\<table\> \</table\>
- Defines a table in the HTML file
- Has several related tags that must appear nested between the elements and these appear below
  Attributes
- border=""
  - o defines the thickness of the border that appears around the image
  - o a value of "0" turns the border off and is the default value.
  - o Values are presented in pixels
- cellspacing=""
  - o defines the distance between the cells, essentially the border thickness present between cells
  - o a value of "1" is normal, the default value, and a value of "0" eliminates the shadow in the border drawn
- cellpadding=""
  - o defines the distance between the text and the border of the cell
  - o a value of "0" places the text right on the border
  - o a value of "1" is normal and the default value
- bgcolor=""
  - o establishes the background color of the web page, default is white
- width=""
  - o defines the width of the column
  - o values possible
    - ▪ percentage of the window expressed as 0% to 100%
    - ▪ pixels expressed as integers
- height=""
  - o defines the height of the column
  - o values possible
    - ▪ percentage of the window expressed as 0% to 100%

- ▪ pixels expressed as integers
- align=""
    - o determines where in the browser window that the table will be aligned
    - o values possible are "left", "right", and "center"
    - o the default value is "left"
- summary=""
    - o enables screen readers to read a summary of the table allowing the user to determine whether or not the table is worth navigating through and can skip if they so chose.
    - o Accessibility standards require the summary attribute to be included
    - o If the table is used for layout purposes the summary must describe the table as used for layout purposes only.

<thead></thead>
- defines the header of a table
- important for the headings of each column
- must be nested within the <table></table> tags
- Code for this first in the table
- essential for accessibility standards, improves pagination and scrolling
- use <th></th> between tags and not <td></td> to define cells
- can have more than one row, use the <tr></tr> tag to add additional rows

Attributes
- align=""
    - o determines how the text in the cells will be aligned
    - o values possible are "left", "right", and "center"
    - o the default value is "left"
- valign=""
    - o determines the vertical alignment of the text in the cell
    - o values possible are "top", "middle", "bottom", and "baseline".
- bgcolor=""
    - o establishes the background color of the web page, default is the bgcolor="" specified in the <table> tag.
    - o use hexadecimal values

<tfoot></tfoot>
- defines the footer of a table
- important for footnotes with the data in the main body of the table
- must be nested within the <table></table> tags
- code for this second in the table
- essential for accessibility standards, improves pagination and scrolling
- use <td></td> between tags to define cells
- can have more than one row, use the <tr></tr> tag to add additional rows
- it may be omitted if now footer is necessary or table is used for layout purposes

Attributes
- align=""

- o determines how the text in the cells will be aligned
- o values possible are "left", "right", and "center"
- o the default value is "left"
- valign=""
  - o determines the vertical alignment of the text in the cell
  - o values possible are "top", "middle", "bottom", and "baseline".
- bgcolor=""
  - o establishes the background color of the web page, default is the bgcolor="" specified in the <table> tag.
  - o use hexadecimal values

<tbody></tbody>
- defines the main body of the table where the core of the data is presented
- important to separate the data presentation portion of the table from the header and footer information to assist with accessing the data for alternative browsers
- must be nested within the <table></table> tags
- code for this third in the table
- improves pagination and scrolling
- it may be omitted if table is used for layout purposes
Attributes
- align=""
  - o determines how the text in the cells will be aligned
  - o values possible are "left", "right", and "center"
  - o the default value is "left"
- valign=""
  - o determines the vertical alignment of the text in the cell
  - o values possible are "top", "middle", "bottom", and "baseline".
- bgcolor=""
  - o establishes the background color of the web page, default is the bgcolor="" specified in the <table> tag.
  - o use hexadecimal values

<tr> </tr>
- defines a row in a table
- must be nested within the tags <table></table>, <thead></thead>, <tfoot></tfoot>, or <tbody></tbody>
- contains the tags <th></th> and <td></td> nested inside <tr></tr>
Attributes
- align=""
  - o determines how the text in the cells will be aligned
  - o values possible are "left", "right", and "center"
  - o the default value is "left"
- valign=""
  - o determines the vertical alignment of the text in the cell
  - o values possible are "top", "middle", "bottom", and "baseline".

- bgcolor=""
  - o establishes the background color of the web page, default is the bgcolor="" specified in the <table> tag.
  - o use hexadecimal values
- width=""
  - o defines the width of a row
  - o values possible
    - ▪ percentage of the window expressed as 0% to 100%
    - ▪ pixels expressed as integers
- height=""
  - o defines the height of the row
  - o values possible
    - ▪ percentage of the window expressed as 0% to 100%
    - ▪ pixels expressed as integers

<th></th>
- defines the table header for a column
- must be nested within the <thead></thead> tags
- can be nested within the <tr></tr> to create more than one row of header information
- automatically formats all text as bold
Attributes
- same for <td></td>

<td></td>
- defines the table data cell for the table
- must be nested within the tags <table></table>, <thead></thead>, <tfoot></tfoot>, or <tbody></tbody>
- can be nested within the <tr></tr> to create more than one row of table data information
- essentially each "cell" within the table is defined by the information between the <td> </td> tags.
Attributes
- align=""
  - o determines how the text in the cells will be aligned
  - o values possible are "left", "right", and "center"
  - o "justify" also a value but only works in some browsers
  - o the default value is "left"
- valign=""
  - o determines the vertical alignment of the text in the cell
  - o values possible are "top", "middle", "bottom", and "baseline".
- bgcolor=""
  - o establishes the background color of the web page, default is the bgcolor="" specified in the <table> tag.
  - o use hexadecimal values
- width=""

- o defines the width of a row
- o values possible
  - ▪ percentage of the window expressed as 0% to 100%
  - ▪ pixels expressed as integers
- height=""
  - o defines the height of the row
  - o values possible
    - ▪ percentage of the window expressed as 0% to 100%
    - ▪ pixels expressed as integers
- colspan=""
  - o defines the number of columns that the cell spans making it appear as if the cells have merged within a row
  - o values possible
    - ▪ integer values; 1, 2, 3, etc.
- rowspan=""
  - o defines the number of rows that the cell spans making is appear as if the cells merged within a column
  - o values possible
    - ▪ integer values; 1, 2, 3, etc.
- nowrap
  - o defines the line of text to complete one line without wrapping
  - o no values, the presence of the word in the attribute portion of the tag turns the value from "off" or "false" to "on" or "true"

<caption></caption>
- defines a tables caption
- nested within the <table></table> tags

Attributes
- align=""
  - o determines how the text in the cells will be aligned
  - o values possible are "left", "right", and "center"
  - o "justify" also a value but only works in some browsers
  - o the default value is "left"
- valign=""
  - o determines the vertical alignment of the text in the cell
  - o values possible are "top", "middle", "bottom", and "baseline".

Form Elements
<form></form>
- defines a form for filling in fields on a web page
- contains the tags <input />, <textarea></textarea>, <select></select>, <button></button>

Attributes
- action=""
- target=""
- method=""

- o values possible
  - get
    - blah
  - post
    - blah
- name=""
- autocomplete
- enctype=""

<input />
- defines an input field in a form
- must be nested within the <form></from> tags
Attributes
- name=""
  - o defines the name of the input field, essential
  - o invisible to the user
- value=""
  - o defines the value present in the field at the time the form is drawn in the browser
  - o visible to the user
- id=""
  - o defines the id for the form input field
- type=""
  - o defines the type of the input field in the form
  - o possible values
    - password
      - encrypts the text typed in the field and displays bullets not the text typed
      - additions
        - o size=""
          - length of the field box in characters
        - o maxlength=""
          - number of characters accepted by the input field, visible or other invisible to the user
    - text
      - defines on line of text for an input field
      - additions
        - o size=""
          - length of the field box in characters
        - o maxlength=""
          - number of characters accepted by the input field, visible or other invisible to the user
        - o autocomplete
    - checkbox

- defines the input field to be a check box and allows more than one in a group to be checked
- additions
  - checked
    - pre-checks the box for the user and maybe unchecked by user prior to submitting the form
- radio
  - defines the input field to be a radio button and allows only one within a group to be selected
  - additions
    - checked
      - pre-checks the box for the user and maybe unchecked by user prior to submitting the form
- button
  - defines a button
  - the text in the value="" attribute determines the text displayed on the button and the text length determines the length of the button
- submit
  - defines a button with the text submit
  - required for a form to be submitted and an action to be taken
- reset
  - defines a button that resets the form back to the blank default values or the values defined by the value attribute for each <input /> defined
- file
  -
- image
  - as

<textarea></textarea>
- defines a multi-line textarea for an input field

<select />
- defines a drop down menu of items for a form input field
- options are defined with the <option /> tag
Attributes
- name=""
  - names the input field, necessary for communication with the server
- size=""
  - defines the number or rows visible when open
- multiple
  - allows more than one choice in the menu list to be selected

<option />
- defines options that make up the list of choices in the menu
- must be within the <select> tag

Attributes
- value=""
  o the text in the value attribute is the text that appears in the menu listing
- selected
  o pre-selects one of the values in a list of choices