CODING BY DUMMIES,
CODING FOR DUMMIES

What I am about to embark on is unchartered territory for me. I've deciphered code, hand-written code, and copy-pasted code on my own for about two years now. I've got a pretty good handle on the basics. Through all of my learning though, I really wished that there was a resource that offered not only the syntax, but also a little understandable explanation of what the code was doing, and how to decipher it. So, having attained a certain measure of "experience" with HTML, I am putting together this article for beginners, written by a beginner! I will use words and explanations you can understand!

Now it is a little dry, and you have to actually be interested in reading about it, because there are not a lot of pictures in code, but that's just the nature of the beast. So, if you want a picture tutorial, this is not the place to be. Likewise, if you already have an understanding of the basics, this will be a snooze fest for you too, because that's what this is all about: the basics. This is essentially a primer to get your feet wet in the shallow end of the coding pool. I will look at some of the basic concepts of coding. Once the concept behind each is addressed, you'll get to see a couple concrete examples of each. I'll show you how to "de-code" expressions into regular English, so you know what each does, and what the format means. Finally, I'll give you some key web references where you can go and really get your feet wet.

So, with that in mind, let's talk first about the various formats that you can use to create web pages. While there are many formats, one of the most common is the "HTML" format. HTML stands for Hyper Text Markup Language. This format is actually what I would call the most basic of building blocks for web pages. You really cannot glaze over understanding some basic html if you want to do anything on the web where you do at least some coding. A parallel would be to say you want to run wind sprints before you even learn how to crawl or walk – you just can't do it.

HTML documents are basically text documents with html elements in them, and saved with an html extension (.html) rather than the text extension (.txt). The key thing to understand here is that the html element is what defines our content. Elements are the parts that comprise the "code" of html. Okay, so what exactly is an element?

HTML elements have two key components: an opening tag and a closing tag. The opening tag will be just the element name, positioned in between a greater than and a less than bracket < is the less than, and > is the greater than bracket. (Remember your high school math?)

Here are some examples of some HTML tags and what they really mean, followed by the closing tag:

Opening Tag	Meaning	Closing Tag
<title>	Indicates the text to show in the title of a page	</title>
	Indicates a paragraph of text	
<h1>	Indicates the first heading of text	</h1>
<h2>	Indicates the second heading of text	</h2>
<head>	Indicates you are in the header section of a document	</head>
<body>	Indicates you are in the body section of a document	</body>
	Indicates items that follow are part of an ordered list	
	Indicates items that follow are part of an unordered list	
	Items within an ordered or unordered list	

<table> Indicates items that follow are part of a table </table>

Hopefully, in seeing the table of some of the basic HTML tags and their meanings, you'll also notice a trend. Each tag will be surrounded by the same brackets, the <>. Similarly, an opening tag will only have the content descriptor inside its tags, while the closing tag will always start with a "/" before the descriptor. So, in a general sense, we can say that HTML code will always follow this format:

```
<descriptor> Text to be displayed </descriptor>
```

As with anything that is this generalized, there will be numerous exceptions. HTML is no different. For instance, some tags are both opened and closed within the same <> set. A good example of this is the line break tag. The line break format is simply the brackets surrounding the text "br". To make the code work though, we still need the "/", so browser knows where the line break ends. The end result will look like the following:

```
<br />
```

Another good example of this is the image tag, where the image is declared at the beginning of the bracket "<img", then additional criteria for that image are specified, such as image location, size, position to place in the document, borders, etc. (also known as attributes), before the tag is closed with a "/>". The image tag would thus look something like this:

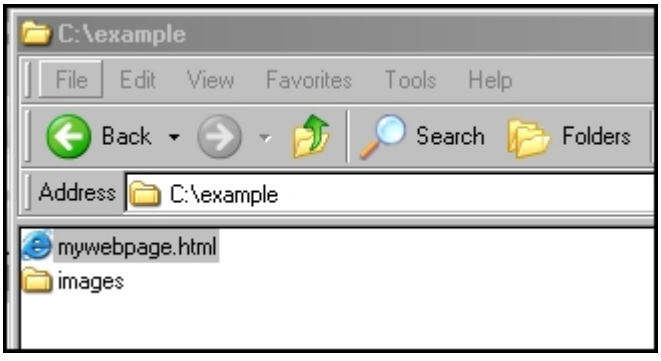
```
<img picture.jpg located in My Pictures folder />
```

I chose the image tag for a reason, because I just introduced a new concept here in html, and that's the one of an html attribute. Attributes define characteristics of a particular tag. Attributes include things like (as indicated) a filename, the location to find that file, what size a file show be displayed at, whether it has borders, what size borders, and a host of other things. But we can't just revert from html to our normal English conventions, because that would leave too much unclear for the browser to render. Instead, we need some uniform formats to keep everyone on the same sheet of code, as it were.

So, conventions have been developed stating where attributes can be declared and the format they should follow. Using the img tag as an example, the first attribute we need to declare is the file location. To do that, we need to start with the term "src" (short for source), and an = sign. This says to the browser "Hey, I've got an image file I want you to display, here's the source location for that file, go get it and display it on the page" So, thus far, the image tag looks like this:

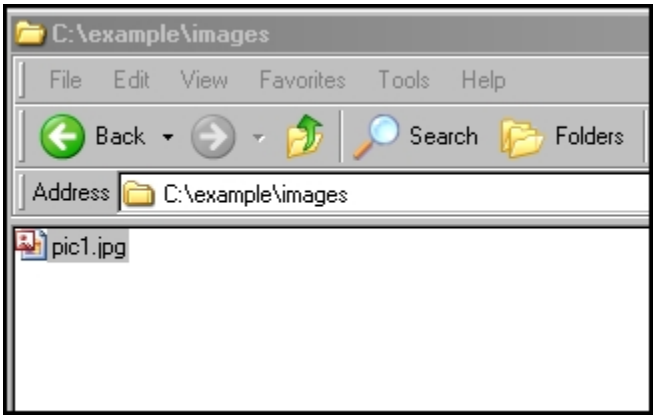
```
<img src="file location"
```

Here's the tricky part… when you specify a file location for an html page to reference, you have to make sure that the file location is relative to where the html file is being read from. Let's look at an example to see what is going on here.



In the example website to the right, we've set up the web page "mywebpage.html" and a folder to store all the images for the site. We can see the source is our C drive, and everything is in the C:\example folder.

Inside the images folder, we've also got the picture we want to reference.



So, if we were going to add an image reference tag to "mywebpage.html", it might make sense to create an image tag that looks like the following:

```
<img src="C:\example\images\pic1.jpg">
```

Here's the thing… this could work during testing! However, when you are finished and upload everything to your web server, the file path will change because it's no longer on the "C" drive! Whether your web server is on a Unix/Linux type of host or a Windows type of host will determine the type of path that you would need to specify, so this will have to be checked through your hosting company (although most web servers that I've come across have been hosted on some variant of a Linux or Unix operating system).

Instead of using the locations for your own computer, it's best to enter file references like this in relative terms of where they are stored relative to one another, all within a main folder (like our **Example** folder). In addition to using this method for easier referencing when writing code, it's just good design practice, because it allows for faster page loads. This means using the "./share/folder/file" convention. In this example website here, the correct code would look like this:

```
<img src="./images/pic1.jpg" />
```

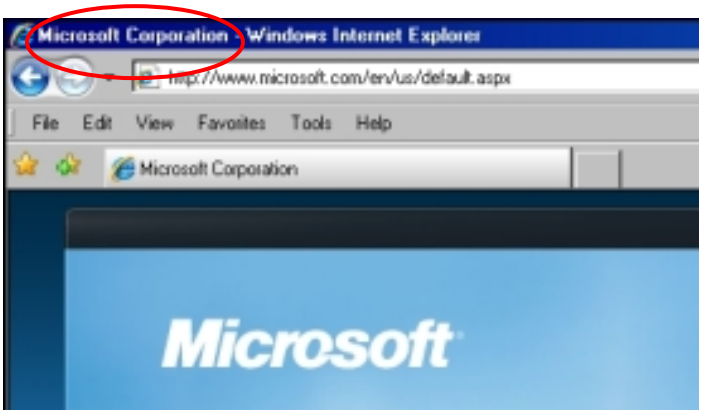
For those of you not familiar with the dot slash concept, you may want to brush up on Linux and Unix.

So, we've determined that HTML is what defines the content of our website – our titles, our headers, and things like that. Some of the more easily recognizable content descriptors and their HTML designation are listed below:

- Title <title>This is my Website</title>
- Head <head>elements to be read that specify instructions for how a page is loaded</head>
- Body <body>elements to be read when the page content</body>
- Heading 1 <h1>Heading 1</h1>
- Heading 2 <h2>Heading 2</h2>
- Heading 3 <h3>Heading 3</h3>
- Paragraph content

We could look at each and every one of these (and more…there are six different heading sizes!), but that would defeat the purpose here…this is just to give you an idea of all that is out there. So, the only one I'll talk about here is the first one, the one that is most commonly **missed**! That's right, the title….many people miss specifying their page title. This is a bad thing because now search engines (Google and all the rest) have no idea what your site is about until they read the content, and they don't read very much…)

So, what is the title? The title is just that, the title that you want to be displayed on your website. Where does the viewer see this? Typically the title is located in the very top colored bar of your browser window. In Internet Explorer, it is on the top blue bar, as shown by the circled text below:



You'll notice that the title is almost always followed by the name of the web browser that you are using – in this case, Internet Explorer. So, if you are using a different browser, yours will obviously look a little different.

By specifying a title, not only are you giving Google and other search engines an idea of what your site is about, but you are also making it easier for people to identify you in their browser window. It's a very handy method for differentiating between well-designed sites and poorly designed ones.

Designing Your Site

Having gotten the grasp now of a few of the basic HTML concepts and terms, let's leave the more advanced stuff to some pretty well-documented materials that are widely available online, and move from the coding of content, to the next phase of coding: designing the layout of your site.

You may have noticed that we didn't talk at all about how to specify the colors of anything, or how to position your content on a website. While it is possible to do this in HTML, advances in web design have brought about something called Cascading Style Sheets (CSS). What these files do is specify how all the designs (or styles...go figure) are laid out. Design refers to all the color schemes, positioning and other aesthetics that are part of a well-designed web page. Some of the more common examples of CSS elements are: font sizes, font colors, background colors, heading positions, and alignment. You see all these elements on a style sheet.

So, what is a style sheet? A style sheet is simply a file that HTML uses as a reference so it knows how to display the content. Let's take a look at an example HTML page below.

```
<html>
<head>
<title>My Website</title>
<link rel="stylesheet" type="text/css"
href="mysite.css" />
</head>
```

So, you notice that the HTML page starts with a declaration saying what type of code it's written for. Then on the second line, you have the head declaration. Next, you can see the title. On the fourth line, you now see that the HTMLpage is accessing a link which is another page – our style sheet! I should say at this point that you can insert CSS styles into HTML, but industry best practices recommends placing CSS in its own dedicated file, so we'll proceed with that approach here.

So, the HTML file references a CSS file which contains all our colors, positioning, and design characteristics. But we still haven't seen a style sheet. Let's take a look at one now! A style sheet is

very much like an HTML file in that you have elements where you specify information. However, whereas an html file has the textual content, the CSS sheet defines the qualities of that element.

In CSS though, the format is a little different. The syntax for style sheets is as follows:

```
element {property: value}
```

Now in CSS, the element section is the same as the element in HTML, but of course, to confuse you, CSS terminology calls it a selector. So, the format becomes:

```
selector {property: value}
```

Let's look at each of these. The "selector" in CSS is the same as the "element" in HTML, just different terms for the same thing. Next up, the "property". This is what style component you want to adjust. Do you want to adjust the font-type? What about the font-color? What about it's vertical position? What about its horizontal position? All these are properties of that selector (or element). The value then would be the number for whatever property you are defining. Let's take a look at an example.

```
#h1  {color: white }
```

This should be pretty straightforward. We have declared the selector to be heading1, and then said we want the color to be white.

So, now it's going to get tricky…we're going to add a second property and value to this selector (element)! Look at the following code:

```
#h1 {
        color: white;
        font-family: times;
    }
```

Now hang on there, Scooter! I changed everything…instead of putting it all on one line, I am taking some text and putting it on another line. That's okay…it's still going to display the web page the same. When you press the enter key (also known as the carriage return, CR for short), that does not register as a positioning property, so we're fine. I just moved things down a line to make it easier to read everything that is going on with the heading. If you get into advanced CSS styles, you will want some way to visually off-set each selector so you can see it better.

So, we've kept the color value at white, and now we've added a font property, saying that we want it to use the Times font family. Notice how each line also has a semi-colon (;) after each value. This is significant because it indicates that more CSS information is coming about this particular element. Just like HTML, we still have to close it when we're done, but as indicated, we use the other bracket, (or the }) to close CSS selectors (elements.)

Conclusion

So, you now know a few snippets of HTML code and are ready to start writing a web page, right? Excellent! Now, head on over to a great website, called W3 Schools to get more detailed tutorials on the specifics of coding not only HTML but also CSS and other web languages like Javascript, PHP and much, much more! I would venture to guess that W3 Schools is probably the most renowned (and free) resource to learning the details of web design.

On that note, here are some of the better web resources I have come across that are useful either for inspiration in design schemes and layouts, as well as for resources in learning how to do different things. Thanks for reading this article, and remember to stop over to my website for more articles, tutorials, tips, tricks and learning materials relating to photography.

Resources

www.w3schools.com
www.alistapart.com
www.csszengarden.com
www.oswd.org
www.good-tutorials.com