HTML and CSS – What you Need to Know

Whenever you look at a website on the Internet, it is evident that there is some sort of underlying structure which governs how the website appears on your browser. The code that governs how content is rendered is called **HTML**.

What is HTML?

HTML stands for Hypertext Markup Language. There are two components of HTML; **tags** and plain **text**. Tags are used to format the content inside depending on how you want it to look on the page. Tags generally follow the following format:

```
<tagname>content that will show on the page</tagname>
```

Tags are enclosed in left and right carets, and *usually* come in pairs. The first tag is known as the **opening tag**; the second tag is known as the **closing tag**, and is denoted by a forward slash followed by the tag name. Tags format the content in between them depending on which tag you use. For instance, take the following:

```
<p>I am a paragraph.</p>
```

<p> is the **paragraph tag**. In this piece of code, the text "I am a paragraph" is situated between the opening and closing paragraph tags. When the browser reads this code, it will output the text "I am a paragraph" formatted as a paragraph. Fairly simple.

Next, consider the following code:

```
<strong>I want to be BOLD!</strong>
```

 is the **bold tag**. Text that appears between the opening and closing tags will be bolded upon output.

Notice that we are certainly allowed to use tags within tags (and in fact is completely necessary to achieve any semblance of a good-looking websites). For instance, what if we wanted to output a paragraph, but make one line of it bold? Then the code that we would want to use would look like:

```
<p>I am a paragraph. <strong>I am really important!</strong></p>
```

This would output:

I am a paragraph. **I am really important!**

Note that it is extremely important for tags to be **balanced**. What does this mean? Let's look at the following snippet of code:

```
<p>Something is wrong. <strong>Can you figure out
what?</p></strong>
```

From the browser's perspective, the browser will first attempt to render the paragraph element. When it does so, it sees the following:

```
<p>Something is wrong. <strong>Can you figure out what?</p>
```

The browser is now confused. Where is the closing tag? In an attempt to resolve this confusion, different browsers will take different actions. Some might throw out what it deems to be an extraneous tag, and thus nothing might get bolded at all. Alternatively, the browser might "guess" as to where the closing is and automatically try to fill it in. Furthermore, once the browser deals with the extraneous tag, it still has to deal with the mismatched tag, since once the browser renders the paragraph, it sees something like:

```
<p>[Content of paragraph]</p></strong>
```

The browser then must determine what to do with the tag, which appears to lack a corresponding opening tag. This leads to inconsistent behavior across browsers, which could turn off potential visitors. So it's always important for your tags to be balanced! Can you see why the following code is incorrect?

```
<p>I am a paragraph. <strong>The ending of me is
important.</p><p>I am the important beginning of a new
paragraph.</strong> I'm not quite as important.</p>
```

Attributes

Sometimes, you can't rely entirely on a tag to entirely describe what's inside of it. For instance, imagine if you worked at McDonald's, and someone comes up to the register and says, "I want a burger". You'd be confused; what kind of burger? What is its number on the menu? What is its price? How many slices of cheese does it have? In the same way, some HTML tags need more information in order to function property. For this, we have **attributes**. Attributes typically take the following form:

```
<tagname attr1='description' attr2='stuff'>Content</tagname>
```

For instance, let's take the code:

```
<a>Link me please!</a>
```

<a> is the **hyperlink** tag, which means that any text in between the opening and closing tags will be formatted as a hyperlink. But wait a minute; where is this link even supposed to go? We didn't specify a destination for it! For this, we need the **href** attribute. Don't worry about what href is supposed to mean; just know that it specifies where the hyperlink should redirect to. So a proper link would be formatted like:

```
<a href='http://www.facebook.com'>Click here to go to FB</a>
```

Notice that the phrase "Click here to go to FB" is what is between the opening and closing tags; this is what will appear on the page. When you click on the link, the browser checks for the href attribute, and then sends you to the appropriate site (in this case facebook.com).

Certain attributes can appear on any type of tag (we will discuss some of these attributes later), while other attributes are specific to a certain type of tag. For instance, there is certainly no point in putting a href attribute on a <p> tag; the browser wouldn't know what to do with it! Indeed, the href attribute only sees usage with the <a> tag for visual elements.

Let us look at another important tag which will require us to provide attributes. This is the **image** tag, which as you can guess displays an image on the page. A typical image will take the following form:

```
<img src='someimage.jpg' alt='an image description' />
```

Now, the first thing you might notice is that there doesn't seem to be an opening or closing tag; rather, it's an unusual combination of both. This is because the image tag is a **self-closed tag**; simply put, there's no need for an opening and closing tag because there would be nothing to be put between them! Self-closed tags take the following form:

```
<tagname />
```

Self-closed tags can have attributes, of course; the image tag we showed above is one of them. The *src* attribute determines where the image to be displayed comes from. The *alt* attribute is used in the event that the image fails to load, in which case the alt text will be displayed, or for assistance for visually-impaired people navigating a site. **The *alt* attribute is a required attribute**; although your site will still render properly if you don't include it, it's nonetheless a requirement for valid HTML, just like balanced tags are.

The image tag is an example of a tag which has a bunch of optional attributes we can use. For instance, the image tag has a *width* and a *height* attribute; if you set these attributes, then when the image is displayed on the rendered page, it will scale itself according to the attributes.

For instance, you can force a large 1280x800 picture to fit within a page by specifying that the width attribute equals 320.

```
<img src='someimage.jpg' alt='description' width='320' />
```

Thus, when the image shows up on the page, it will be a 320x200 picture.

A quick reference to important tags we will be using

We have covered the basics of tags. As such, here is a list of the tags which are most commonly used, and their important attributes:

Name	HTML	Description	Important Attributes
Hyperlink	`<a></a>`	Creates a hyperlink.	href – where the hyperlink goes
Paragraph	`<p></p>`	Creates a paragraph.	None
Div	`<div></div>`	Creates a container element.	None
Image	`<img />`	Creates an image.	src – the image's URL to be loaded alt – a description to use in various purposes

Heading	`<h1></h1>`
	`...`
	`<h6></h6>`
		Creates a heading, which is used to specify sections on the page.	None

Line Break	` `	Creates a line break.	None
List	`<ul></ul>` `<ol></ol>`	Creates a list (unordered for , ordered for).	None
List Element	`<li></li>`	Creates a list element. Must be located between or tags.	None

There are lots more tags out there; you can find tutorials on how to use them elsewhere. For the most part, you are equipped to deal with all tags.

CSS (Cascading Style Sheets) – A Primer

So far, we've discussed a lot of tags that will help define your content just the way you want it. But hold on a second; what about formatting? What if I want to make a paragraph's text blue? Is there a <blue> tag that I need to use? What if I want to make the paragraph's *background* blue? Would you use <bgblue>? Are there tags for every single color out there? Clearly, this cannot be the case. For this, we need to introduce **CSS**. CSS provides us a standardized and easy way to stylize elements. For instance, take a look at this snippet of code:

```
<p style='color:blue; font-size:16px;'>I am styled.</p>
```

The syntax here should be fairly explanatory. Here, we have a new **style** attribute. The style attribute can be used on *any* element, because every element is modifiable by CSS in some way. In this instance, our desired effect is that our paragraph has blue text and has an increased font size. CSS rules are declared with **properties** and **values**. For instance, in the code snippet above, color is the *property*, and blue is the *value*. Similarly, font-size is a *property*, and 16px is the *value*. Each individual line is called a **declaration**. So, "color:blue" is a declaration, and "font-size:16px" is a declaration. Declarations are separated by semicolons.

Using the style attribute, you can style specific elements on the page to your liking. But there's a problem with this. What if you want multiple paragraphs on each page to have the same styling? You could copy and paste the same style attribute into multiple paragraph tags. It's a bit cumbersome, but it works. But let's say you change your mind; blue is out of fashion, and green is what's in. You would need to find every blue paragraph on your site and manually change the value to green! If you have a lot of pages on your site, this is going to take forever. For that, we have an incredibly important concept:

Classes

Imagine that you're building a website, and each part of the page has a different purpose. For instance, perhaps you're building a website that will post a news article and then let people comment on it at the bottom. Suppose you're thinking, "I wish I could tell the page that the paragraphs at the bottom are supposed to be for comments". Well you're in luck; **classes** help you do just that! Let's take a look at the following snippet of code:

```
<p class='comment'>I am a special paragraph for comments.</p>
```

We have introduced a very important new attribute here, the **class** attribute. The class attribute is also a global attribute, so we can use it on any HTML element. Alone, the class attribute does absolutely nothing. How is the browser supposed to know what a class called "comment" is supposed to mean? To figure this out, first, we need to explore some more tags.

If you open up any web page, you'll notice something interesting. First, the entire page will always be enclosed by an <html></html> tag. This is fairly self-explanatory; the HTML indicates that what follows in between the tags is, well, HTML. However, the contents inside are divided into two sections; the **head** and the **body**. Accordingly, we have corresponding tags <head></head> and <body></body>. In between the <body></body> tags is where you will place all of the *visual* HTML elements; paragraphs, text, tables, all of that goes in the body. However, there are elements of a page that are not necessarily directly displayed on the page; those usually go in the head. For instance, at the top of any web page, you'll see that there's a title doesn't necessarily correspond to the name of the web site. Guess what tag that's defined

in: the **title** tag! The title tags, denoted by <title></title>, must be placed in the head of the document. In other words, your document must look something like:

```
<html>
     <head>
          <title>My Awesome Webpage!</title>
     </head>
     <body>
          <p>Here's some cool content of mine!</p>
     </body>
</html>
```

Notice the indenting that I'm using in my code; this helps keep clear which elements are contained within other elements. For instance, it is easy to see that the <title> is totally surrounded by <head>, and that the <p> is totally surrounded by the <body>. This will help you avoid unbalanced tags.

There's a whole host of other stuff that typically goes into the head. Most commonly, you'll see <meta>, <title>, <link>, <script>, and possibly <style> in the head. Don't worry about what most of those mean; we'll get around to those in due time. However, the one we are now going to be using is <link>. Now, contrary to what it sounds like, <link> does not indicate a hyperlink. Rather, it's used to load information about the page from an external source. Why would we want to do that?

Let's go back to our whole comment styling situation. What if, instead of having to copy and paste a style attribute into each and every paragraph, you could define what a comment is *supposed to* look like on some external source, and then reference that source whenever you need to style a paragraph? That's just what we're going to do!

Creating a Style Sheet and Linking it to your Web Page

In your code editor, go ahead and create a new document. Save it as "style.css". Now, in explaining how to code a style sheet, it would probably be beneficial to look at an example. Here's what a CSS file might look like:

```
.comment {
     text-size: small;
     color: grey;
}
#header {
     width: 600px;
     background: blue;
}
```

A good portion of this should be familiar to you. Declarations work exactly the same as they did before, with the exception that all of the declarations are enclosed within a curly brace. The text outside of the curly braces are called **selectors**. In the example given above, the two selectors are ".comment" and "#header".

Selectors define what elements of a page the declarations should apply to. Selectors that begin in a period, such as ".comment", apply to classes. In other words, they essentially state, "The following rules (declarations) should apply to any element which has a class of 'comment'". The pound sign, such as in "#header", refers to a new attribute, called the **id** attribute. The ID attribute works in a similar way to the class attribute, except that **ID ATTRIBUTES MUST BE UNIQUE**; no two elements on the same page can have the same ID.

There are many other unique ways to specify selectors. Perhaps, for some reason, you want all of your tables on the site to have a pink background. You could specify that using the following:

```
table {
     background-color: pink;
}
```

Notice that there is no period or pound sign before the selector. When this is the case, the selector refers directly to the tag. In this case, it's saying, "any <table> elements on the page should have a pink background". **BE VERY CAREFUL TO INCLUDE THE PERIOD WHEN DEFINING THE RULES FOR A CLASS.** For instance, imagine if we had a class called "header-button", but when writing the CSS rule for it, we forgot to include the period. The resulting code would look like:

```
header-button{
```

```
    width:30px;

    height:30px;
}
```

When the browser reads this, it thinks: “Well, there’s no period or pound sign, so it must be referring to a tag. I’m going to look for any instances of <header-button></header-button> on the page”. Of course, such a tag does not actually exist, and thus the rule would not actually do anything. The corrected code would look like:

```
.header-button{
    width:30px;

    height:30px;
}
```

Notice that the wrong code looks very similar to the correct code. Writing correct CSS is just something that you need to get used to. I’ve spent hours debugging what was wrong with my CSS just to find out that I forgot a period on a class. So be ever vigilant! If something isn’t working, chances are you made a silly mistake like forgetting a period.

There are a TON of selector rules, but there are a few that are worth mentioning. Let’s take the following example:

```
div.comment a{
    color: green;
}
```

Notice two things here. First, in regards to “div.comment”, notice that we have combined both a tag and a class. This makes the rule more specific; for instance, perhaps you don’t want the styling to apply to all divs, or you put the class “comment” on multiple types of elements and only want to stylize the divs with class “comment”. That’s all and well, but what about the hyperlink tag following it? This rule, as shown above, is essentially stating, “For any hyperlink that is contained within a div with the class ‘comment’, make it green”. When two selector elements are separated by a space, it means that any the CSS rule will apply to any instances of the second element that are contained within the first element. This is in contrast to the following:

```
div.comment, a{
     color: green;
}
```

Now, we've placed a comma between the two selector elements. When the comma is present like this, this means that the CSS rule is applied to BOTH of the elements. Without the comma, the CSS rule only gets applied to the hyperlink (specifically, any hyperlinks that are contained in a with class "comment"). This can be useful when certain elements have similar properties but aren't exactly the same. For instance, imagine we were building a table which had alternating row colors. Regardless of the rows' background colors, their height and width would still be the same though, so we could do something like:

```
tr.odd, tr.even{
     height: 50px;
     width: 600px;
}
tr.odd{
     background-color:blue;
}
tr.even{
     background-color:purple;
}
```

Let's return to our example of the "comment" class. Imagine that your style.css file is as follows:

```
.comment{
     font-size:8px;
     color: grey;
}
```

So now we've got a nice, central location for what our comment paragraphs should look like. How do we tell our web page to use this information? For this, we need to use the <link> tag in the head. Here's an example of how you might include your CSS file:

```
<link rel="stylesheet" type="text/css" href="style.css" />
```

Notice that the link tag is a self-closed tag. The **rel** and **type** attributes simply indicate that we're attaching a CSS style sheet to our document. The href attribute, similar to how we saw it used in the hyperlink tag, specifies where the style sheet is located.

Once you include this tag in your head, assuming everything is done right, when you load the page, you should be able to see the new styles being applied to any elements with the "comment" class. Here's what a properly formatted page might look like:

```
<html>
    <head>
        <title>Welcome to my page!</title>
        <link rel="stylesheet" type="text/css"
href="style.css" />
    </head>
    <body>
        <p>This is a paragraph. I want it to be normal.</p>
        <p>This is another normal paragraph.</p>
        <p class="comment">Here is a comment!</p>
        <p class="comment">Here is another comment!</p>
    </body>
</html>
```

And the corresponding style sheet:

```
.comment{
    font-size:8px;
}
```

What now?

At this point, I have taught you almost all the basics of what it's like to build a website. If you understand HTML and CSS, you are ready to get going. The best way to learn, of course, is not by reading tutorials, but by going out there and actually trying stuff out! Try to replicate the visuals of a site you already know, and in the process learn about a bunch of CSS attributes that I haven't taught you yet.