Mary Naylor

Basic CSS Tips and Tricks

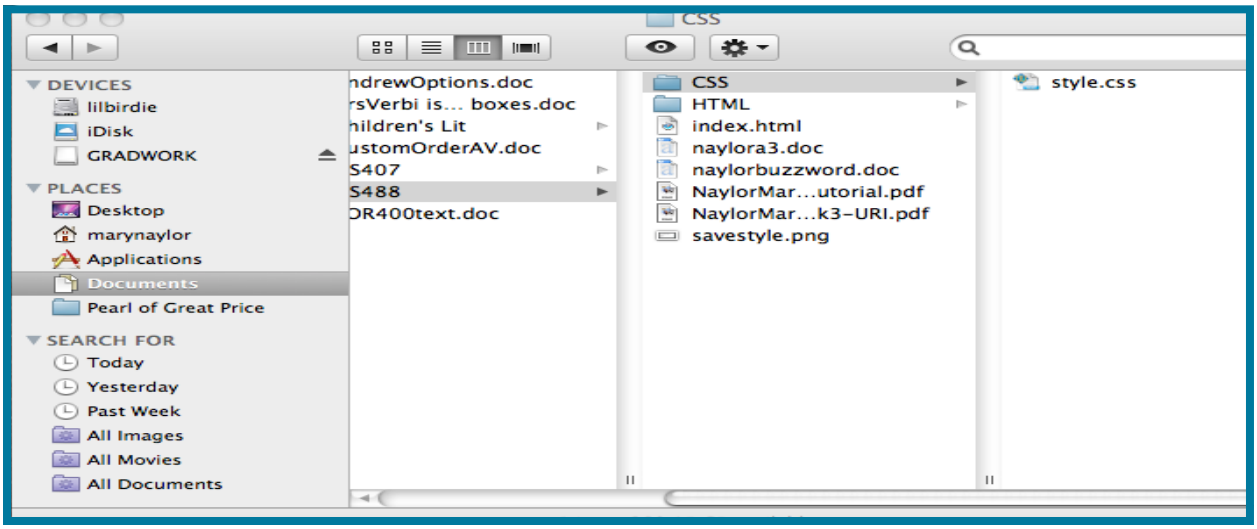
You have spent a lot of time getting the HTML for your library website ready to rock and roll. All your code has validated, classes specified, and paragraphs put in order. There is only one problem—it looks really boring. As it loads on the page you'll notice something: everything is black text on a white page with preset sizes for all your headings. All the paragraphs, images, and captions will load line by line down the page from top to bottom mirroring your HTML code just without the tags. This is not particularly inviting. So how do you make a web page look more than a type written piece from a word document? The answer is CSS (or Cascading Style Sheet). This tutorial is designed to get you started, and provides places where you can go for more advanced ideas and applications. I'll start with two spoilers—the more you tinker and toy with CSS the better you will be at it, and a lot of times, when it comes to design, less it more.

Alright, let's get started!

Step 1: Creating your CSS page

Making the page

First in the folder on your computer or flash drive where your index.html page is saved create a new folder title "CSS." Now open up your favorite code-writing text-editing program—Notepad, Text Edit, or Text Wrangler will all do the trick for free. Before you type any code save the document as "`style.css`" in the new folder you have just created. This will make the style sheet easy to find and name in your all HTML files, while at the same time, keep you organized. The file type "`.css`" tells the computer and browser to read the file as a CSS document and not something else.



Index.html with the CSS folder and style sheet saved properly.

Linking it to your HTML

Following the above steps you will have created an external style sheet—you won't ever have to dig through code to change how something looks, if you know the element you want to style you are golden, if you want to change something specific, you only need to find the ID, and not muddle your HTML, but add a specific style to the CSS page which is easier to navigate and manipulate. It this also keeps everything neat and tidy. One caveat is making sure your HTML knows to "talk" to your CSS Sheet.

The best place to put it is within your `<head>` tag in your HTML document. That way before the content even loads the style sheet has already been called and applied.

Your HTML `<head>` tag should look something like this to start:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
      <title>CSS Tutorial Index</title>
</head>
```

Now right before the closing `</head>` tag add the line:

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

Now your whole `<head>` tag should look like:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
      <title>CSS Tutorial Index</title>
      <link rel= "stylesheet" type= "text/css" href= "css/style.css"/>
</head>
```

So what is going on here? The style sheet is within the `<link>` tag, which is a self-closing tag, and there are three elements within the link tag. It tells you the relationship "rel" the link holds to the HTML—it is it's style sheet, what kind of information it is calling up "text/css," and where to find it (href, just like an image tag). Just like any images on your page you are telling the browser to go into the folder you just made called CSS and call the name of the file, where it will find your style sheet by the name you provided. You then close the tag. If you are attaching your style sheet to an HTML document that is in your HTML folder the href will be "`href= ../css/style.css`" with the "`../`" telling the browser to jump out of the HTML folder, go into the CSS folder, and then grab your style sheet. Now your HTML document will open with the style sheet applied.

Step 2: Creating styles knowing the Syntax

CSS like HTML has a particular syntax that the computer reads as style. It can be validated just like a HTML text; for a good CSS code validator try: *http://validator.w3.org/#validate_by_input.* The syntax is rather simple. It is:

```
element name{
                style-type: the style you want;
          }
```

For example if you want the default body of the website to be all sans-serif, burnt orange, with a light blue background color, and 11 pt text the code would look like this:

```
body
      {
            font-family: sans-serif;
            color: #CC3300;
            font-size: 11pt;
            background-color: #99CCFF;
      }
```

Body is the tag HTML name so the browser knows where to apply the style (anything that falls in between the body opening and closing tag, curly brackets set apart the instructions for what style will be presented, each line ends in a semicolon, and lines are organized in a way that you can follow. For example: all the font declarations are grouped together. As a general tip: try and keep the order of color, size, and family of all the fonts declared in the same order no matter the element they are styling; it will make it easier for you to navigate and adjust your own code.

CSS has a hierarchy; it goes from broad to narrow. This means if you set a style in the *<body>* tag all elements in the body will follow suit. As you declare styles for elements housed within the body the more specific tags and elements will trump the default you set with the body. The browser also reads code from top to bottom. It is best to organize your CSS from broad to specific within certain elements. Let's say your website has a particular style scheme for your paragraphs, but you want to some of them to have blue text. Those paragraphs are declared in a CLASS in your HTML. We'll talk about how to do the actual declarations and styles in a minute, for now what is important is the order of the two tags. You want the generic *<p>* tag listed first, and then following that and special tags that go along with your paragraphs to follow before you move on to the next element. Your CSS page will then have your entire paragraph styling lumped together starting broad with a generic paragraph, to paragraphs with a specified CLASS, to paragraphs with a particular ID.

Now it is time to jump into the particulars about how to tell the browsers to differentiate between generic, classed, and ided.

A class is declared in CSS by a *period* before the element's class name for example:

```
.bluepara
        {
                color: blue;
        }
```

Any paragraphs in your HTML that have the class of bluepara will now have the text color changed to blue.

An id is declared in CSS by a # before the id name

```
#head3
        {
                color: green;
        }
```

This would change the element that had the ID as head3 to have a green font color.

Step 3 Lets get rolling

Now that your website is communicating with the style sheet, and you know how to format your code, we are ready to really start styling some of this elements.

Look through your page, what kinds of things do you need to define? Which headings, styles of lists, and specific classes do you need to be aware of? With those things in mind you can get started adding some pizzazz to your website.

General Document Changes

Remember how I said, "less is more" earlier? This is where that really comes into play. It will overwhelm most users if every paragraph changes fonts, sizes, and background color. A lot of times picking out what you want the general overall look of your website you can define that off the bat within the `<body>` tag and the web, unless changed somewhere lower in your code, will load everything to this standard. This is nice for keeping most things uniform.

So what can you do within the body of your webpage? It is good to set a background-color, a font-family, font color, and perhaps the size and position of your elements on the page (position goes beyond the scope of this tutorial).

Before I go into specific tags within CSS I think we need to have a conversation about color. Other elements have depth to them, but color gives you a lot of options, and if you are not familiar with web or print design it can be a little frustrating. Computers read color in 3 ways an all can be used in CSS. First we have plain English. CSS allows you to name the standard crayons from crayola and it will know what color to put.

Ex:

```
p{color: red;}
```

This will give the text of your paragraphs red. This is limiting and some of the colors are not the best for viewing on a computer screen. There are more than eight colors and you will probably want some of the variations.

The computer also understands hexadecimal unless you plan on using it a lot there is no reason to go into any depth, but instead supply a source that gives you a chart of how to declare what colors http://html-color-codes.com/.

The declaration looks like:

```
p
      {
            color: #CC3300;
      }
```

This will give you paragraphs with the font color set to a bold burnt orange.

The third way to declare color is using RGB or red green blue. This declares on a scale from 0-255 how much red, blue, and green you want mixed together to make the perfect shade. To get the same burnt orange as above the code would be:

```
p
   {
         color: rgb (204,51,0);
   }
```

A good reference chart is http://html-color-codes.com/rgb.html. Use what color scheme makes most sense to you.

Okay, now that we have some color figured out, let's get rolling on what tags are actually used in CSS.

Tags to remember

For Background changes

```
Body
      {
            background-color: black;
```

changes the background color with a flat color

```
            background-image: url: paper.gif;
```

an image will be placed on your webpage as a background behind your text and other elements

```
            background-repeat url: imageyouwantrepeated.png;
```

this allows for the image to be repeat either horizontally or vertically which is particularly effective for texture like images

```
            background-position: right top;
```

sets the starting position of an image

```
      }
```

For Text changes

```
Body
        {
                color: #CC3300;
```

this will change the color of the font, it can be declared using RGB "rgb(number1,number2,number3) hexadecimal "#number of color you want", or plain English

```
        line-height: 200%;
```

sets the line height, if you want to add leading, this is how you can do that, particularly useful for websites text heavy selections geared to children. It can be set as percents, pixels or numbers, in terms of percents 100% is a single spaced word document, 200% is a double spaced word document.

```
text-align:center;
```

defines the horizontal alignment of the text, with values of left, center, right

```
        text-transform: lowercase
```

can change the capitalization of text with values of uppercase, capitalize, and lowercase

```
        }
```

Fonts

Fonts are a little funny. Because each user may have different defaults set on their browser, or different fonts loaded onto their computer you need to make sure you are meeting the needs of all the viewers of your webpage. Luckily CSS gives you options with that. There are three font families: serif, sans-serif, and monotype; serif fonts have feet (Times, Times New Roman, Garamond etc), sans-serif fonts do not have the lines at the end of the letters (Arial and Verdana), and monospace fonts have characters that have a uniform width (Courier, and Courier New). With all of this in mind you can pick any font you want for your webpage, but there is no guarantee that your user will see your page with that font. CSS lets you set multiple kinds of fonts for each element to allow for various computer defaults. When you made the font-family declaration you can create a list of fonts you want used from left to right in priority.

As serifed example:

```
Body
        {
                font-family: Garamond, Times, "Times New Roman", serif;
        }
```

This tells the browsers "please use `Garamond`, if that isn't available use `Times`, if that is not available use `Times New Roman`, and if `that` isn't available use `a serifed font` that is available.

Other tags with font include:

```
Body
    {
        font-size: 12px;
```

this specifies the size of the font used, you can define it using ems, px [pixels], and using ems with a percent, if you use px you will be in the realm of Microsoft word sizing so it is often considered easiest

```
        font-style: italic;
```

this allows you to change the text from being italic or not

```
        font-weight: normal;
```

this allows you to change the text from being bold or not

```
    }
```

Conclusion

This is enough to get you started, but it is by no means the whole power that CSS contains. After you master these basics your website will look inviting and professional, however some of the powerhouse in CSS is its positioning abilities and the variety of ways it can manage images. These are more complicated concepts that would be the next steps in moving forward with your website design.

Places to go for further reading:

The W3Schools are great for explanations, cheat sheets, and have great examples for you to try, and you can find them at: *http://www.w3schools.com/css/default.asp*

To get some ideas for different ways that CSS can be used you can go to *csszengarden.com*

Color charts can be found at: *http://html-color-codes.com/rgb.html* *http://html-color-codes.com/*

Also, *HTML, XMHTML, and CSS All-in-One Desk Reference For dummies by Andy Harris and Chris McCulloh (2008)* has some great reference tools, explanations, and lists of tags in case you have forgotten.

Once you have written your CSS code, don't forget to validate it to make sure you aren't missing any semicolons, and that it is running up to web standards. A great place to do that is *http://validator.w3.org/#validate_by_input*.

Referenced

"CSS Tutorial." W3Schools. Refenes Data, 2011. Web. 24 Oct 2011. <http://www.w3schools.com/css/default.asp>.

Gibbons, Matt. "CSS Lectures." CHum250 Class. Brigham Young University CHum Department. Provo. October 2010. In Person.

Harris, Andy, and Chris McCulloh. HTML, XHTML, and CSS All-in-One Desk Reference for Dummies. Hoboken, NJ: Wiley Publishing, Inc, 2008. Book II Chpt 1.

"Web Color." Visibone. Visibone, 2011. Web. 25 Oct 2011. <http://html-color-codes.com/>.