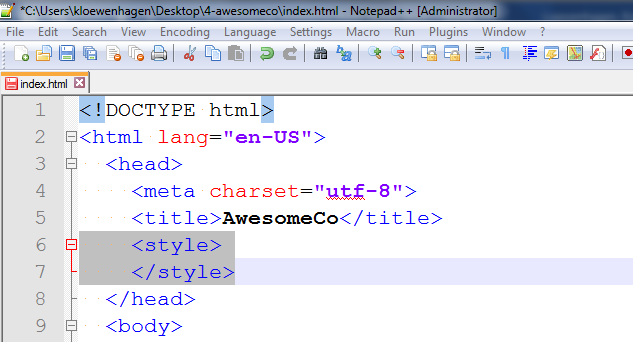
**Tutorial: AwesomeCo – Basic Styling with CSS**

Let's use basic CSS to modify the AwesomeCo web site that we did last time. We'll change some colors, some spacing, and we'll do a tiny bit of transformations with the navigation area of the page.

**Open and unzip the AwesomeCo template file found in the Lesson item above this tutorial.**

We're going to start by opening the index.html page of our site in Notepad++ and creating a **<style></style>** tags at the top of the page in the head section. We'll do all our style work here as we practice, and then we'll move it to an external file once we get the main page looking nice.



Let's change the color of the background and foreground of the page. To do that, we apply style to the body of the page. So, we can use an element selector to find the body, and then create a rule.

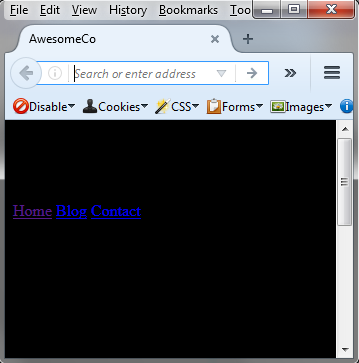
Add the body selector inside the style tag in the head. The background-color property sets the background color of the page. We can give it a word, like "black".

**body {**

**background-color: black;**

**}**

When we save the page and load in the browser, we see... absolutely nothing with the exception of the defaulted blue hyperlinks. We've changed the background to black, but the text color is black also, so we can't see anything!



So, let's change the foreground color:

**body {**

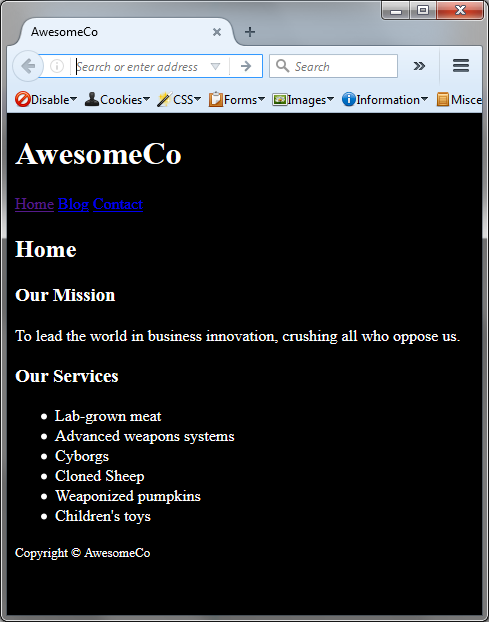
**background-color: black;**

**color: white;**

**}**



All right! This looks all "cool" and "hacker-like." Not very professional, and probably not very readable. We'll fix that in a minute. But first let's talk about colors.



**RGB**

There are quite a few words we can use, but if we want fine-grained control over colors we must use RGB hex codes.

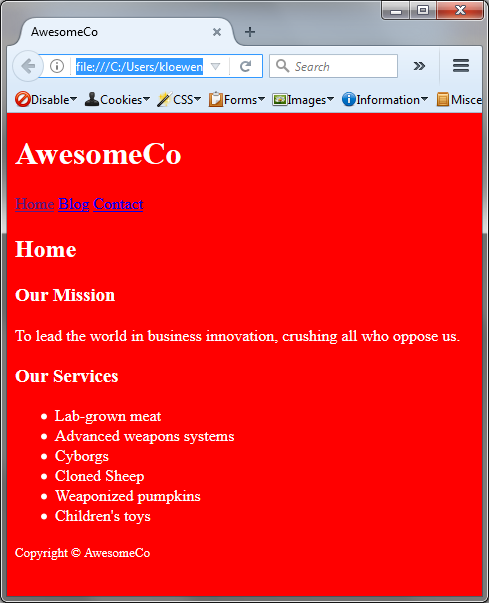
I'm going to change the background of the page to red using an RGB hex code:

**body {**

**background-color: #FF0000;**

**color: white;**

**}**



A hex code is composed of three sets of two hexadecimal numbers. The first two are the amount of red, the second two are the amount of green, and the third two are the amount of blue. The lower the number, the less color. 00 is all the way off, and FF is all the way on. FF is the number 255 in hexadecimal.

So, looking at our value here, you see FF, or all on, for red. Then the next two digits are both zero, and that's green, so green is all the way off. The last two are also zero, so those are off. Thus, we get red.

If we wanted a blue background, we just change the number to #0000FF.

If we want black, we do #000000 - all off.

If we want white? We do #FFFFFFF - all on.

Lower or upper case hexadecimal is the same. So, #FF0000 is the same as #FF0000.

If you use an online color tool, you can easily get the hexadecimal color values for the color you want. And you can use an eyedropper tool to pluck color out of pictures too.

Let's set the background color of the page to white. And let's set the font color to a very dark grey. We're going to avoid black, because black is unnatural - we never see true black anywhere, and it tends to overpower other colors. Using a dark grey can be more soothing to the eyes.

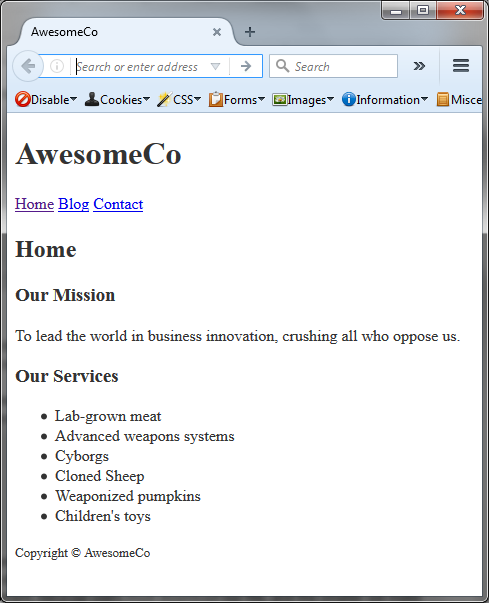
**body {**

**background-color: #FFF;**

**color: #333;**

**}**

Notice that I've only used three digits for my colors here. This is a shorthand notation. If the digits in each section of the number are the same, we can shorten it to three digits, where the first is red, the second is green, and the third is blue. This saves some characters in the file and is important, because users must download our file, and the smaller we make our files, the faster they'll download. That's important on slower mobile connections.



**Margins**

Next, let's look at how we can center some text. Add the following selector to the styles tag in the head. We'll center the text in the header:

**header {**

**text-align: center;**

**}**

We use the `text-align` property to do that. Be default, the value is left. What if we wanted to do this for the footer and nav elements too? Well we could duplicate the rule with a different selector, but CSS lets us use commas to apply a rule to multiple selectors:

So, if we want the footer and nav, we just add them to the selector:

**header, footer, nav {**

**text-align: center;**

**}**

Speaking of the footer, let's add some color to the footer. Let's also change its font size so it's a little smaller:

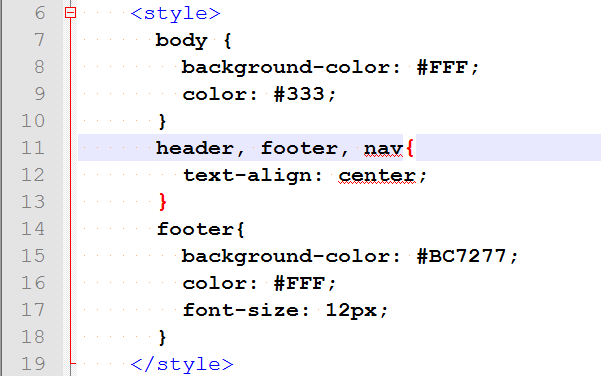
**footer {**

**background-color: #BC7277;**

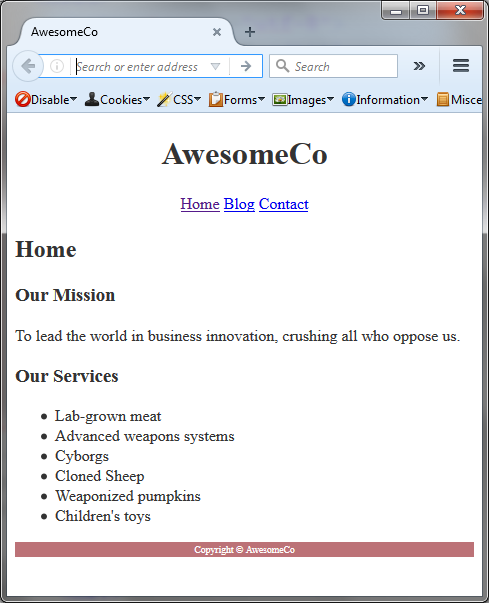
**color: #FFF;**

**font-size: 12px;**

**}**



Save the index.html file and view in browser:



Now let's look at fixing some of the spacing of elements. In the browser, if we right click on the h1 element, inspect it, and click on the h1 tag in the html; we'll see that there is some space above and below the element.

Now look at the p element: It also has space above and below, but it's a little smaller.

The space around an element is called its margin, and there are CSS properties that control this. Let's define a CSS rule that sets the margins for the h1, h2, h3, and paragraph elements:

We use the margin-bottom property and give it a value of space we want. Let's do 20 pixels and see how that looks:

**h1, h2, h4 p {**

**margin-bottom: 20px;**

**}**

Then for the other margin values, we'll use 0:

**h1, h2, h4 p {**

**margin-bottom: 20px;**

**margin-left: 0;**

**margin-right; 0;**

**margin-top: 0;**

**}**

We don't need the measurement type here. We can just say 0.

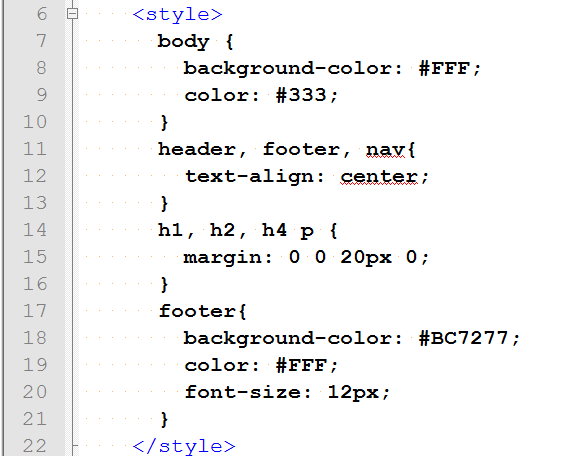
That's a lot of typing. There's a shorthand though.

**h1, h2, h4 p {**

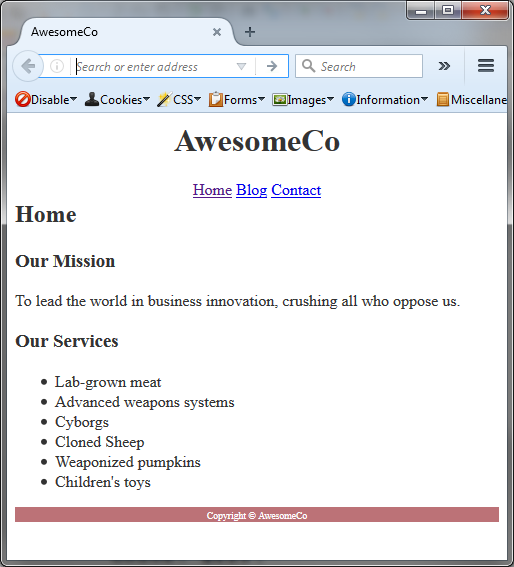
**margin: 0 0 20px 0;**

**}**

This does the exact same thing. And it's easy to remember too. Think of a clock. This syntax goes top, right, bottom, and left. Just like a clock. Top is 12, right is 3, bottom is 6, and left is 9.



Save the index.html file and view it in the browser:



Notice how the web page tightens in appearance.

**Navigation**

Let's do something cool with our navigation area now. First, let's space out the links. We've learned how to do that with the margin:

We don't want to do this to all of the links on our page, so we'll use what's called the "child" selector.

**nav > a {**

**}**

This says, "get me all the anchor elements inside of the nav area." But you must understand how CSS works: It's not going to find the `nav` area first. It first finds all the links on the page and THEN narrows it down to only the ones in the nav area.

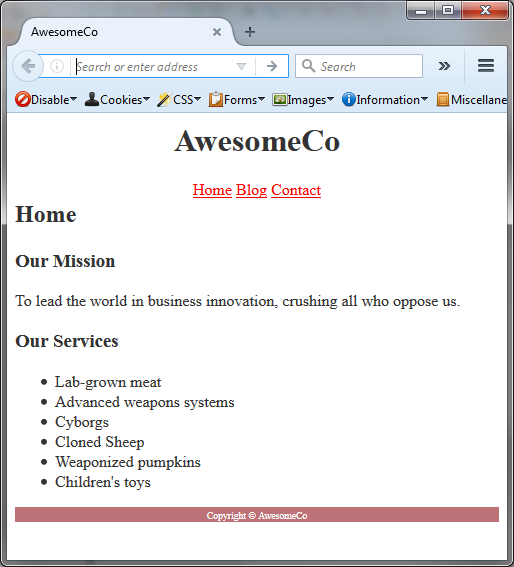
Let's change the color of the links.

**nav > a {**

**color: #F00;**

**}**

Add the above child selector to the style tag in the head. Save the index.html file and view it in the browser:



**Pseudoclasses**

The interesting thing about the link color we set is that it overrides the color for all the states of the link. When we click on a link, the browser usually changes that link color to let us know that we've already been there. That's important behavior and we want to preserve that.

I'll click on the link for the blog in my browser. When I get there, there's no styling on the page. We'll fix that later

when we share the styles across pages. But when I go back, I should see the "blog" link show up with a different color. I don't.

CSS provides "Pseudoclasses" which let us target certain interface components and behaviors.

For example, to change the color of visited links in the navbar, we do this:

**nav > a:visited {**

**color: #FF7373;**

**}**

When we view this in the browser, we'll now see our visited links in a different color. We can also change the color of links we hover over with the mouse!

**nav > a:hover {**

**color: #DDD**

**}**

Now, mobile devices can't handle hover. They have no way of knowing when we're hovering our finger over something. What we do is handle touch events. When the user touches a link, we can change the color. So, we can also change the color of links we focus on:

**nav > a:hover, nav > a:focus {**

**color: #DDD**

**}**

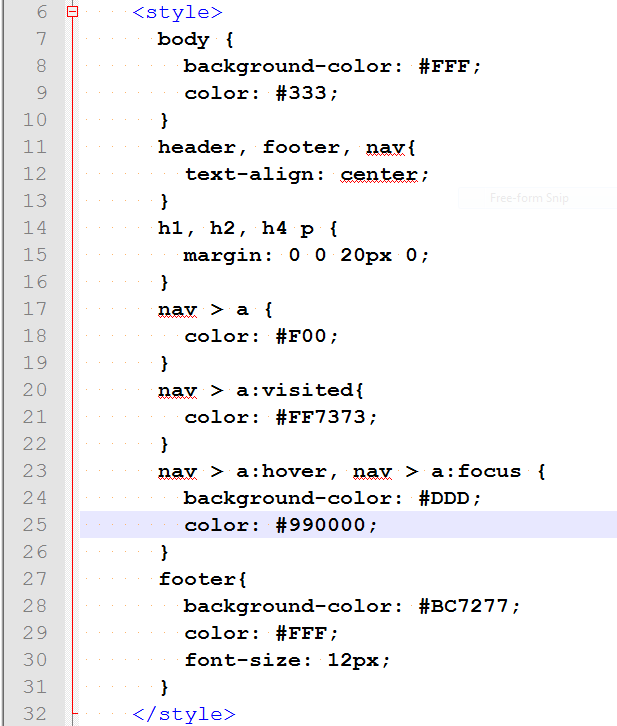
We can change the background color of this element too. Let's use the gray color for the background, and we'll use a nice dark red for the foreground.

**nav > a:hover, nav > a:focus {**

**background-color: #DDD;**

**color: #990000;**

**}**



After we save the file and run, when we hover, the background color changes to light gray too!

Our links are a little too close together and they touch the background color when we hover, so let's space them out a bit. We'll add space by using padding, which adds space between the text of the element and the element itself. To do this, add the following selector to nav > a.

**padding: 0 10px 0 10px;**

This uses the clock format, but if the top and bottom are the same, and the left and right are the same, we can shorten this further:

**padding: 0 10px;**

Modify the previous line to the line above. This is 0 on top and bottom, and 10px on the left and right.

While we're here, let's remove the underlines from the links. We do that with the `text-decoration` property:

**text-decoration: none;**

And while we're there, let's bump up the font size a little bit more:

**font-size: 1.5em;**

This uses the em unit of measure, which is a relative measurement. Pixels are fixed, but relative units of measure will change based on the overall font size we set. A value of 1em is equal to the current font size. Remember that the browser's default font size is 16 pixels. We're using 1.5em here, which will make the font size 24 pixels. But using relative units can be nice because then if we change the base font size, everything else can change with it.

So, the nav>a should look like:

**nav > a {**

**color: #F00;**

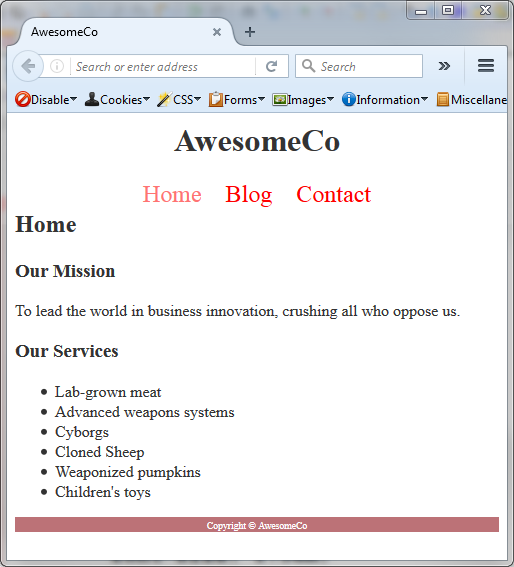
**font-size: 1.5em;**

**padding: 0 10px;**

**text-decoration: none;**

**}**

Let's save and reload and see the changes. It's looking better.



Let's add a little visual separation between the navigation area and the rest of the content. We'll apply a border to the bottom of the nav element.

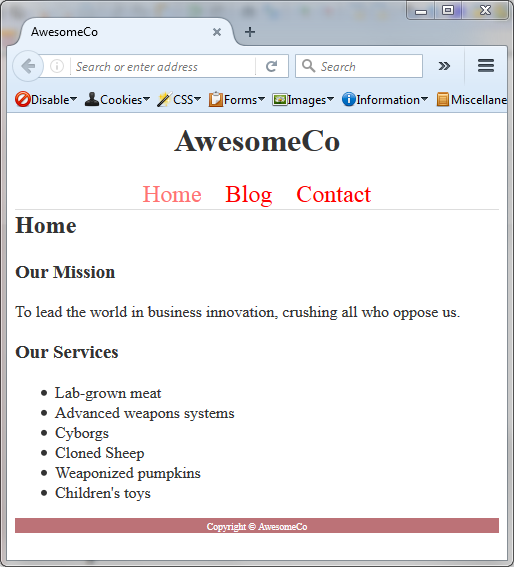
**nav {**

**border-bottom: 1px solid #DDD;**

**}**

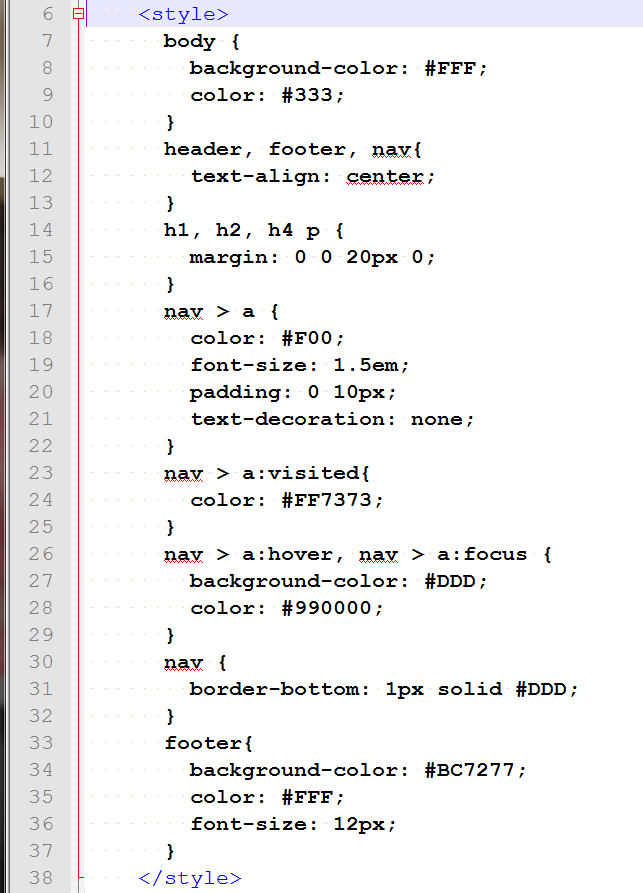
We only want to apply the border to the bottom, so we use the border-bottom rule. Then we specify the thickness of the border in pixels, then the type of border that we want - a solid line, and then we specify the color.

There we go! Our page is looking a little more colorful.



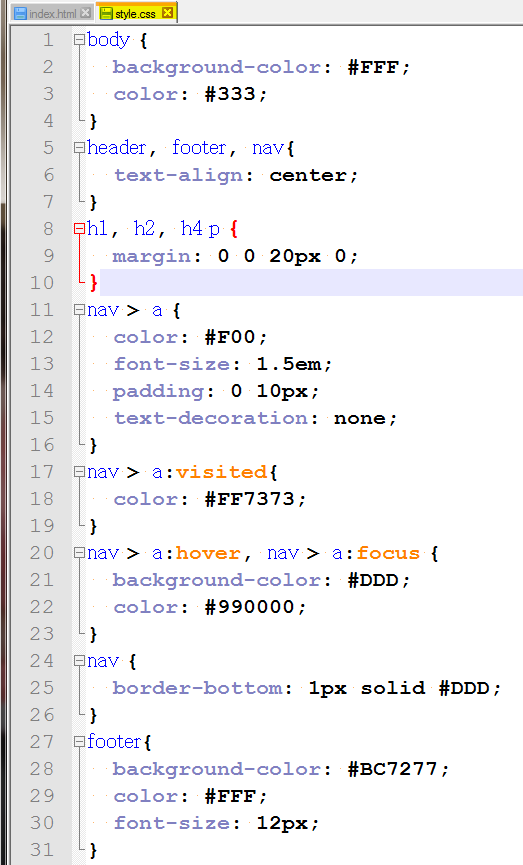
There's more we could do, but we'll stop here. Next week we'll learn how to make a little nicer navigation for our site.

The completed style tag in the index.html file:



**Sharing Styles**

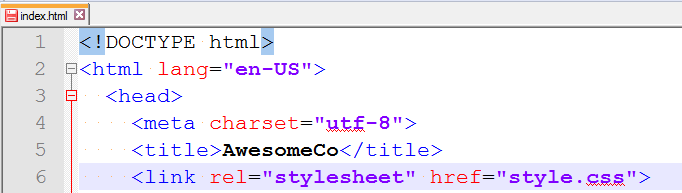
Our stylesheet is only available on this page because we placed it in the style tags at the top. Let's move this to an external stylesheet. Create a new file called `style.css` in the same folder as our pages. Then, we'll cut all the CSS code from our home page and paste it into that file. You may have to highlight all the cut selectors in the style.css file and press the SHIFT-TAB keys to move the indent to the left.



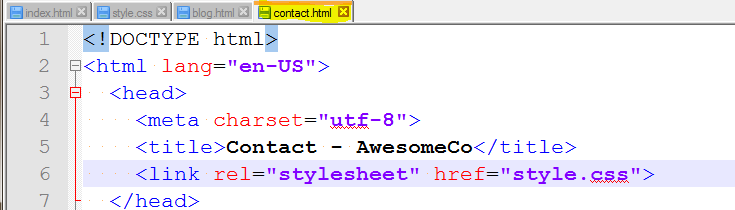
We'll remove the style tags from inside the HTML head tag and replace it with a link tag that points to the stylesheet:

**<link rel="stylesheet" href="style.css">**

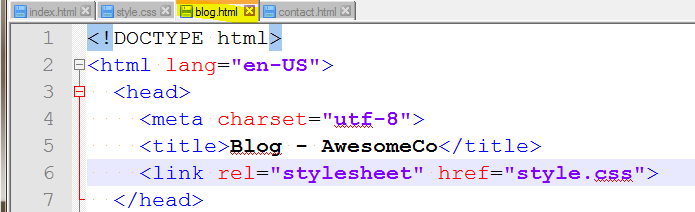
A link tag doesn't have a closing tag.



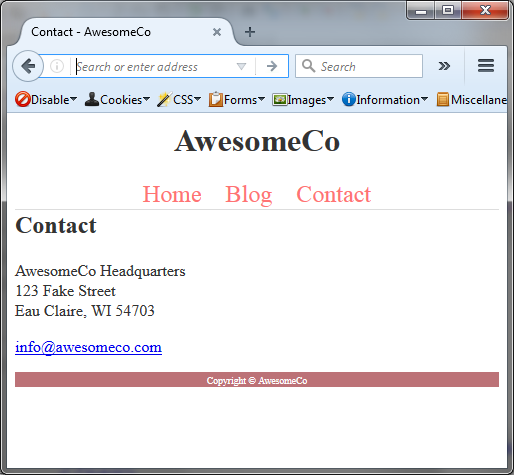
Then we can copy this line to our other two files, and now our style is applied to all the files.



And



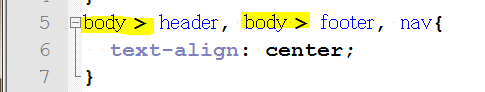
And that does it for this week. Save all files and browser all 3 pages.



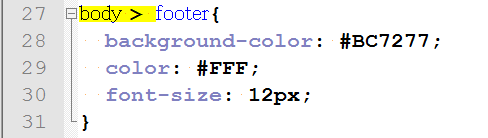
And



Do you notice anything particular on the Blog page? Yes, the article header and footer are picking up the page header and footer CSS. To fix that, go into the style.css file add body > to the following:



And



Let's reflect on what we learned. First, we learned about CSS colors and how the hex color system works. Then we looked at how margins work and how we can declare things. After that we looked at Pseudoclasses so we can make things change when we hover over them. Finally, we looked at moving the style sheet to a separate file so we can share the styles across our pages. Be sure to practice these techniques so you can begin applying them to your projects.

See styles.css file below:

