**Tutorial: AwesomeCo - Mobile First Design**

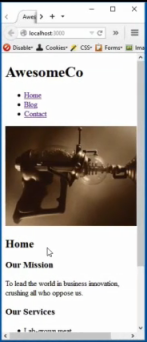
So far, we've developed sites for a single screen size. But now it's time to develop a site for mobile phones, tablets, and large screens. And the only good way to do that is to start with the small screen design first, and work outward.

If we don't do that, then we're trying to think about ways to hide things, instead of thinking about things to add.

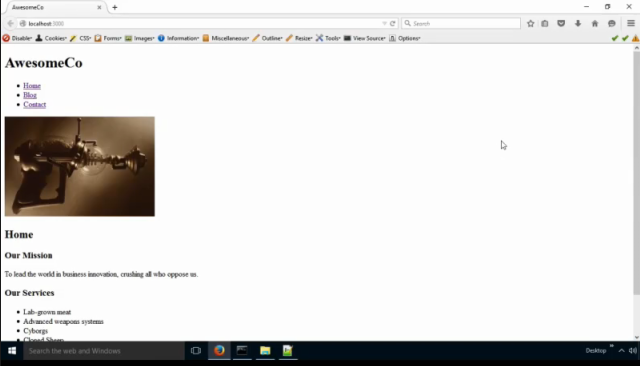
We'll start with the same AwesomeCo site we've been working with. The only change you'll see this time is that there's an image on the home page. The rest of the markup is exactly the same as last time, with a wrapper div and a main section, and navigation in an unordered list.

We're going to use an approach called "responsive design", where we change the layout by responding to the way the user is using or holding their device. We'll have one page and one stylesheet, but we'll use special CSS rules called "media queries" to make the site look different on different screen sizes.

But before we begin, we need to understand that the browser already does this for us. If we make our browser window very narrow, you'll see that our page would look quite good on a mobile device. There's very little that we have to do.



It's when the screen gets wider and wider that we run into issues of layout and readability. This is another reason we start with the small screen.



**If you have not done so already, download the AwesomeCo starter files for this tutorial. Open the files in a text editor.**

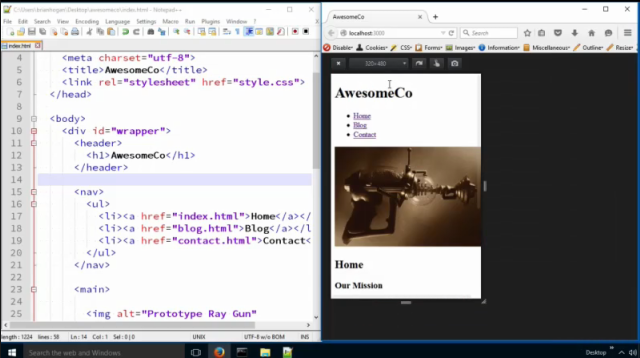
So, let's use the Responsive mode in Firefox to display our site.

**Mobile Devices**

**Right click on the web page – “Inspect Element” and select the responsive button. Close the console window.**



**At the top of the responsive design window, select 320x480 which is where we want to start.**



Now, let’s **open our style.css stylesheet in a text editor**. Let's first change the margins on all the headings. There's too much wasted space.

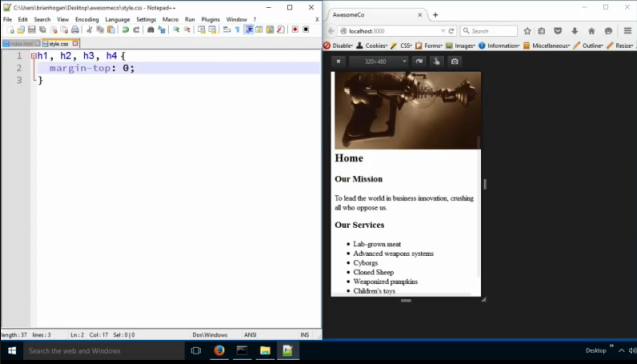
**In the empty style.css file, type in the following CSS:**

**h1, h2, h3, h4 {**

**margin-top: 0;**

**}**

**Save the style.css file and refresh the index page.** Notice the text move up abit on the screen which reduced white space.



Next, let's get rid of the text logo and use our own graphical logo. We have a smaller version we can use for smaller screens. It's smaller in file size.

We'll use the image replacement trick we learned about before.

**In the style.css file, type in the following CSS:**

**header > h1 {**

**background-color: #fff;**

**background-image: url(awesomeco\_logo\_mobile.png);**

**background-repeat: no-repeat;**

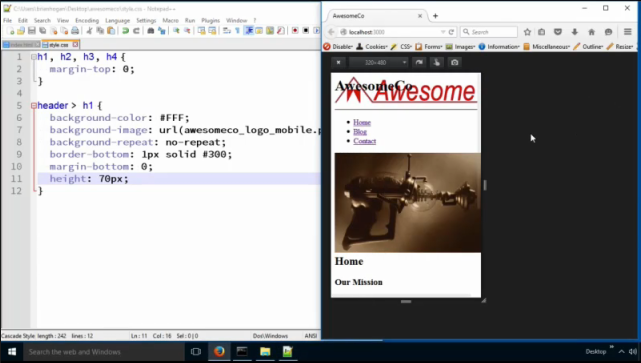
**border-bottom: 1px solid #300;**

**margin-bottom: 0;**

**height: 70px;**

**}**

**Save the style.css file and refresh the index page.**



Notice that the h1 is overlapping the logo image; let’s fix this. We’ll move the header off screen.

**In the style.css file, type in the following bolded CSS:**

header > h1 {

background-color: #fff;

background-image: url(awesomeco\_logo\_mobile.png);

background-repeat: no-repeat;

border-bottom: 1px solid #300;

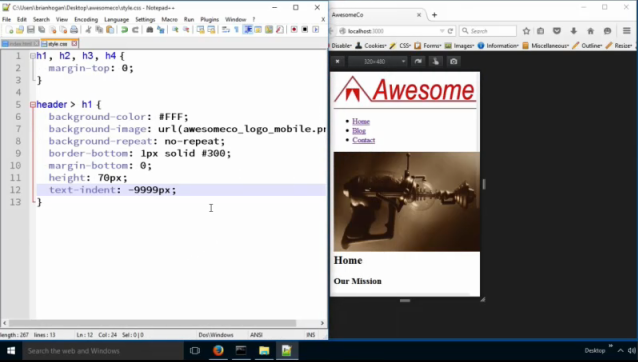
height: 70px;

**text-indent: -9999px;**

}

**Note:** -9999px means that the h1 is located 9999px off the left side of the web page in never-never land; not visible to the human eye but still visible to the search engine bots to read the h1 tag.

**Save the style.css file and refresh the index page.**



Now, the image doesn't quite fit. We can center it and make it fill the space using some additional background properties.

**In the style.css file, type in the following bolded CSS:**

header > h1 {

background-color: #FFF;

background-image: url(awesomeco\_logo\_mobile.png);

background-repeat: no-repeat;

**background-position: center;**

**background-size: contain;**

border-bottom: 1px solid #300;

height: 70px;

margin: 0;

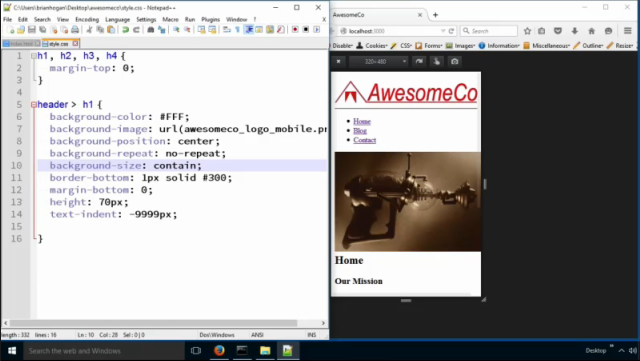
text-indent: -9999px;

}

`background-position: center` centers the image in the box.

`background-size: contain` tells the browser to contain the image inside of the box, stretching it to fit if it overflows.

**Save the style.css file and refresh the index page.**



Now, let's center the headings. It'll make them stand out a little more.

**In the style.css file, type in the following bolded CSS:**

h1, h2, h3, h4 {

margin-top: 0;

**text-align: center;**

}

**Note:** You can save and refresh if you want to see the headings centered.

Now let's build out some navigation by turning the links into buttons. First, we take off the bullets and margin and padding. We will be careful to only target the underordered list.

**In the style.css file, type in the following bolded CSS:**

**nav > ul {**

**list-style: none;**

**margin: 0;**

**padding: 0;**

**}**

And then we turn the individual list items into buttons.

**In the style.css file, type in the following bolded CSS:**

**nav > ul > li {**

**border: 1px solid #333;**

**border-radius: 5px;**

**margin: 10px auto;**

**text-align: center;**

**width: 90%;**

**}**

Then we fill in the button color by styling the anchor elements.

**In the style.css file, type in the following bolded CSS:**

**nav a {**

**background-color: #600;**

**color: #FFF;**

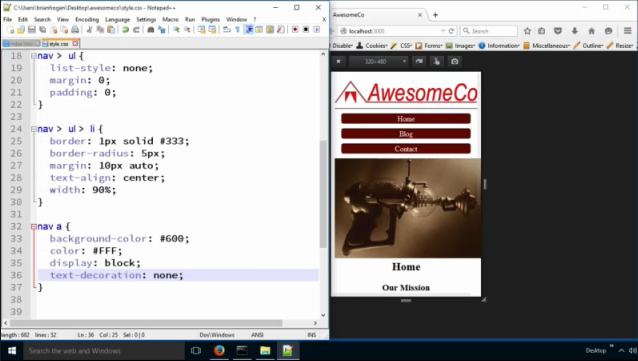
**display: block;**

**text-decoration: none;**

**}**

**Note:** text-decoration: none removes the underline.

**Save the style.css file and refresh the index page.**



Now, let's add a background color to the wrapper.

**In the style.css file, type in the following bolded CSS:**

**#wrapper {**

**background-color: #BC7277;**

**}**

Now our content is a little hard to read, so we'll go ahead and style up the `main` region of the page with a white background.

**In the style.css file, type in the following bolded CSS:**

**main {**

**background-color: #FFF;**

**border-radius: 5px;**

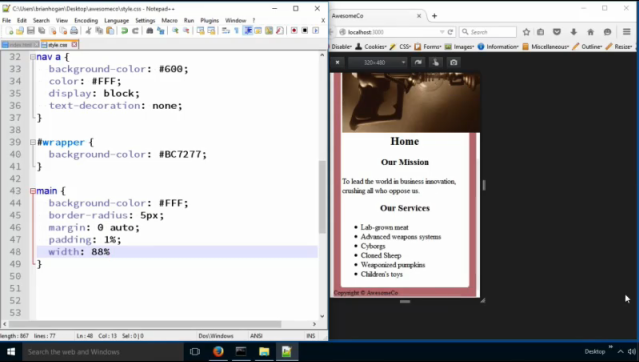
**margin: 0 auto;**

**padding: 1%;**

**width: 88%;**

**}**

**Save the style.css file and refresh the index page.**



We add some padding around this to keep the text from touching the edges. Also, note the math here. The width of the buttons was 90%, but the width here is only 88%. Remember that because we have 1% padding, we have to account for that. We have 88% for the width, plus 1% padding on the left, and on the right. In the end, the actual computed width is 90%, the same as the buttons.

This is the number one mistake people make when it comes to doing layouts in CSS - simple math errors.

So now we have this nice little border around our main content, and our footer is colored in too. Of course, we should style up the footer by centering the text and changing its color to white.

**In the style.css file, type in the following bolded CSS:**

**footer {**

**color: #FFF;**

**text-align: center;**

**}**

**Note:** Save and Refresh the index.html page to see the results.

Now, our image is a little big. Maybe we should just not show the images on these small screens. They don't really add much. Let's hide them.

**In the style.css file, type in the following bolded CSS:**

**main > img {**

**display: none;**

**}**

Sadly, this will still download the image, since it's referenced in the actual HTML code. But it does save us some space on the page. We could use CSS to load this image in, but that's not always an option, and this image is really part of the page's content, not just a decorative thing like a logo.

Another thing we can do is shrink the image down a bit. Let's do this instead. **Remove display: none; in the main > img selector.**

**In the style.css file, type in the following bolded CSS:**

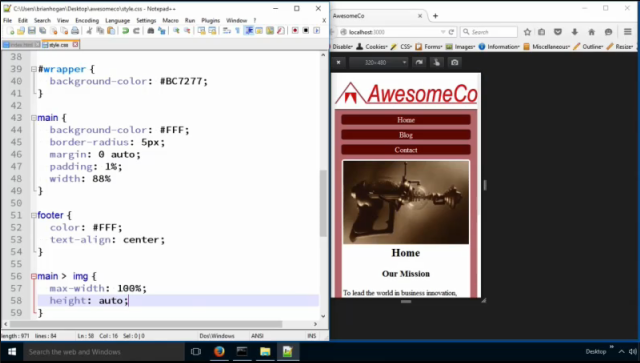
main > img {

**max-width: 100%;**

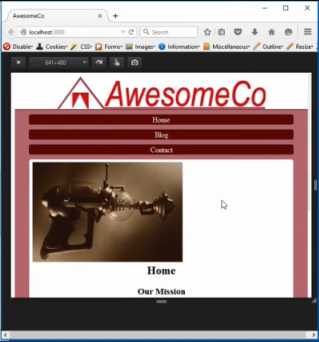
**height: auto;**

}

**Save the style.css file and refresh the index page.**



So, let's scale this up a bit. How does it look? Well, because we planned for the small screen first, it looks pretty great.



**Tablets**

Tablets have more room to display content, and we should take advantage of that. Let's use a media query to support screens that are tablet size or higher.

**In the style.css file, type in the following bolded CSS:**

**@media only screen and (min-width: 768px) {**

**}**

On a tablet, we can use our larger image of the logo. We only need to redefine some of the properties. The new logo is taller, so we'll change the height of the header area, and we'll load in the new image. We'll also set the `background-size` property to `auto`, which will turn off the stretching we were doing on the smaller screens.

**In the style.css file, type in the following bolded CSS in the tablet media query:**

**header > h1 {**

**background-image: url(awesomeco\_logo.png);**

**background-size: auto;**

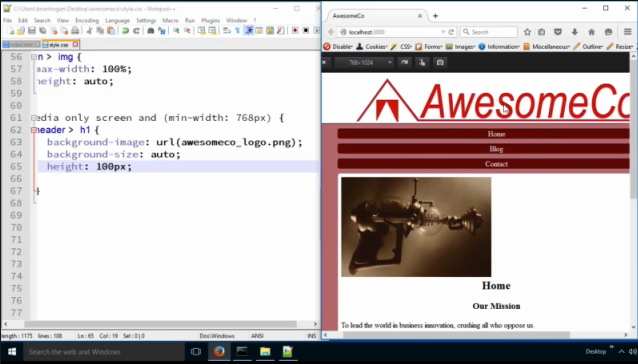
**height: 100px;**

**}**

**Note:** awesomeco\_logo.png is the larger image.

**In Firefox, adjust responsive design view port to 768x1024.**

**Save the style.css file and refresh the index page.**



We certainly have enough room to show the image and have the text flow around the image. We'll make the image float left.

**In the style.css file, type in the following bolded CSS in the tablet media query:**

**main > img {**

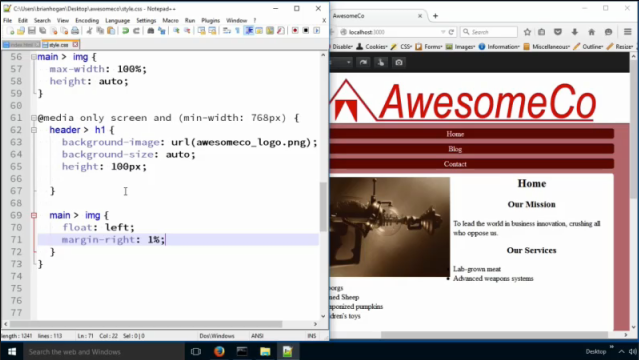
**float: left;**

**margin-right: 1%;**

**}**

The image is a little too close to the text so we'll add a right margin to it, to give it room.

**Save the style.css file and refresh the index page.**



It's important to note that we only have one image here, so this works fine. But if we had multiple images and we only wanted to target one image, we would need to go alter the markup by adding in a class or an ID on the image. But for this example, we only have one image so we can use this selector and move forward.

The bullets end up hiding under the picture because the picture is floated and out of the normal flow. We need to establish a new block formatting context for the unordered list. We do that with several techniques, but the most universal is to define `overflow:hidden` on the undordered list.

**In the style.css file, type in the following bolded CSS in the tablet media query:**

**main ul {**

**overflow: hidden;**

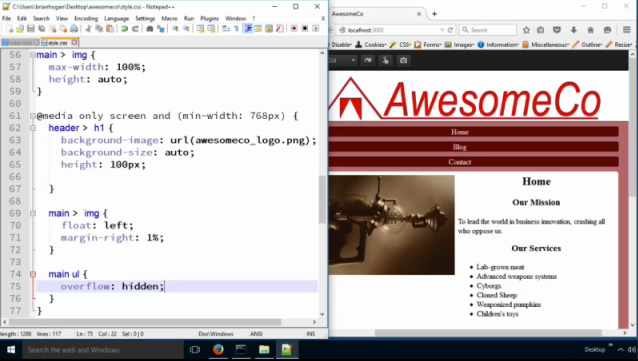
**}**

This establishes what we call a "block formatting context" in the unordered list, so that all its child elements are locked into the location of the unordered list. So, it'll start where the top of the list starts.

This concept of the block formatting content is in the W3C specification. Using 'overflow: hidden" here causes the browser to make the element narrow and affixes it next to the floated element, instead of allowing it to wrap around

the floated element.

**Save the style.css file and refresh the index page.**



Now that looks better.

Our navigation buttons are pretty wide. We have a lot of room now; let's make our navigation into a horizontal navigation area.

We'll set each list item to be `display: inline-block` which gives the elements the ability to have widths set, but they won't take up their own line. Then we can apply a width of 20%.

**In the style.css file, type in the following bolded CSS in the tablet media query:**

**nav > ul > li {**

**display: inline-block;**

**width: 20%;**

**}**

Finally, we'll center the contents of the navigation area.

**In the style.css file, type in the following bolded CSS in the tablet media query:**

**nav {**

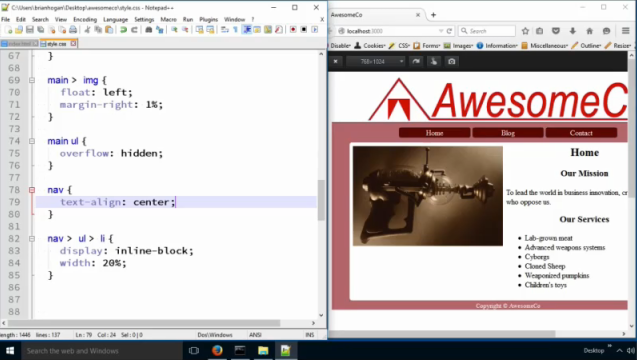
**text-align: center;**

**}**

We can't rely on the auto margins to center these elements horizontally - we have to center the contents of the navigation to get the effect we want.

And that does it for our tablet layout.

**Save the style.css file and refresh the index page.**



That does it for our tablet layout.

If we think about how we’re building this, we’re adding things on to our base style, designed for mobile. As we get larger, we may find ourselves setting something for small screens and then having to undo that for large screens. For example, we've centered the headings on our page. And now we want to uncenter the text.

In this case, it makes more sense to alter our CSS so we only center the text on \*small\* screens. We can use a max-width media query to do this. We can say if the screen’s maximum width is 767 pixels or \*smaller\* then we’ll hide the images and center the headers.

So, we'll add this new media query.

**In the style.css file, type in the following bolded CSS:**

**@media only screen and (max-width: 767px) {**

**h1, h2, h3, h4{**

**text-align: center;**

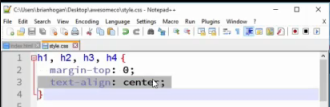
**}**

**}**

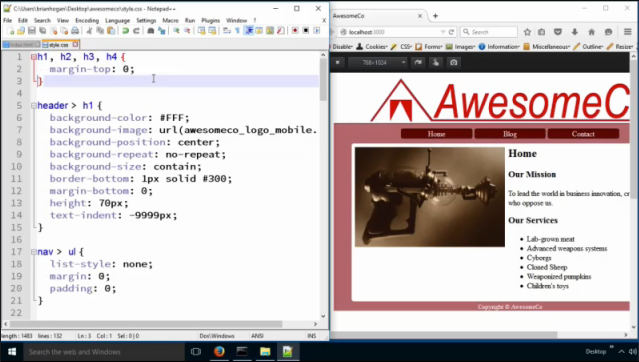
and then we'll remove the rule that centers the text from the top of our stylesheet.

**Delete the following CSS line at the very top in the h1, h2, h3, h4 selector:**

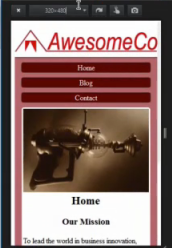
~~text-align: center;~~



**Save the style.css file and refresh the index page.**



And now this rule is all constrained to the smaller screen size. The tablet display headings are not centered and the mobile display is centered. Brilliant!



**Desktop / Large Screen**

So now, let's expand out to the big screen. Anything over 960 pixels wide can be "the desktop size." So, we need a media query for that.

**In Firefox, change Responsive Design to change to 980x1280.**

**In the style.css file, type in the following bolded CSS:**

**@media only screen and (min-width: 960px) {**

**}**

So far, we haven't needed to constrain our page; we've let it expand as wide as we want. But if we keep on doing that, the columns of text are going to get too wide for people to read comfortably. Let's constrain the wrapper so that it takes up 80% of the screen. We'll also center it.

**In the style.css file, type in the following bolded CSS in the desktop media query:**

**#wrapper {**

**margin: 0 auto;**

**width: 80%;**

**}**

Also, let's un-center the logo. We may want to put a search box in the upper right corner on the large screen, so let's make room for that.

**In the style.css file, type in the following bolded CSS in the desktop media query:**

**header > h1 {**

**background-position: left;**

**}**

Now, let's put the navigation on the side and the main content next to it like we did previously. First, we float the navigation left.

**In the style.css file, type in the following bolded CSS in the desktop media query:**

**nav {**

**float: left;**

**width: 15%;**

**}**

And then we nudge the main section over. We'll also take off the border radius on the main region.

**In the style.css file, type in the following bolded CSS in the desktop media query:**

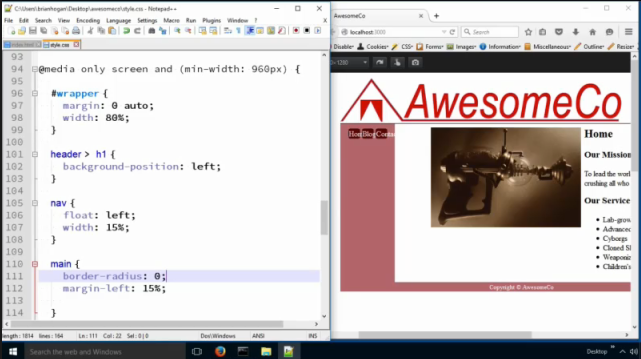
**main {**

**margin-left: 15%;**

**border-radius: 0;**

**}**

**Save the style.css file and refresh the index page.**

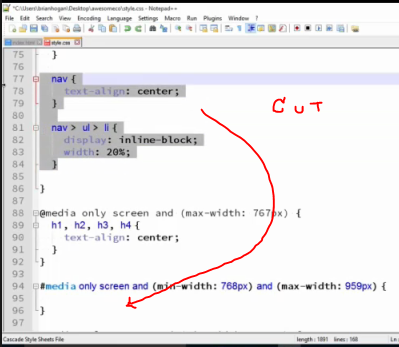


But as you can see, our navigation is kind of messed up. We're inheriting the styles of the horizontal navigation. We could write more CSS to "undo" that navigation, but if we think about it some, we'll realize we're only using that navigation when the screen is between 960px and 768px.

So, let's wrap the two rules for the horizontal nav in their own media query. Put that media query above the last one we defined.

**In the style.css file, cut in the following bolded CSS:**

**Note: Cut the nav>ul>li and nav from the CSS above into the below stated media query.**



**@media only screen and (min-width: 768) and (max-width: 959px) {**

**nav > ul > li {**

**display: inline-block;**

**width: 20%;**

**}**

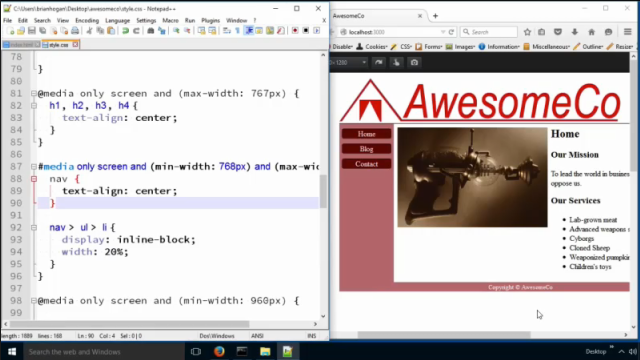
**nav {**

**text-align: center**

**}**

**}**

**Save the style.css file and refresh the index page.**



There! That takes care of that!

Now just a moment to reflect; we're just using what we're learning about media queries and adding in rules based on when things look silly. That's how you have to go about this. There's not any "rules" you can follow because every site is different. You need to understand how media queries work so you can make adjustments. Different screen sizes exist; you can't just make up breakpoints for everyone. Instead, try mobile first and keep expanding.

**Eye Candy**

Our original desktop design had a gradient background, so let's add that in here. We'll do that in the Desktop / Large Screen media query:

Now let's add in the gradient background.

**In the style.css file, add in the following bolded CSS to the desktop media query:**

**body {**

**background-color: #DDD;**

**background-repeat: no-repeat;**

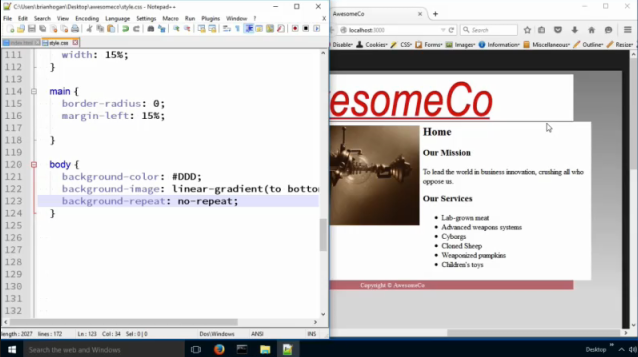
**background-image: linear-gradient(to bottom, #888, #DDD);**

**}**

**Note:** #DDD is a light gray color.

**Save the style.css file and refresh the index page.**

When we add the gradient in, we can see a new problem: Our main content region sticks out pretty far. Can you guess why?



It's because earlier, we defined the main content region to be 88% of the width of its container. However, when we did that we didn't have a 15% margin on the left side. Now we've added one that pushes the content over. So now the

math is off.

We need to adjust the width of the main section by subtracting the amount that's overflowed. So, we have to take 100%, subtract 15% left margin, which gives us 85%. But then we have to account for the 1% left padding, and the 1% right padding.

**In the style.css file, add in the following bolded CSS to the desktop media query:**

main {

margin-left: 15%;

border-radius: 0;

**width: 83%;**

}

Our new width ends up being 83%.

For fun, let's add a drop shadow to our wrapper, but only on large screens. And let's also round the corners of the container by adding a border-radius.

**In the style.css file, add in the following bolded CSS to the desktop media query:**

#wrapper {

**border-radius: 5px;**

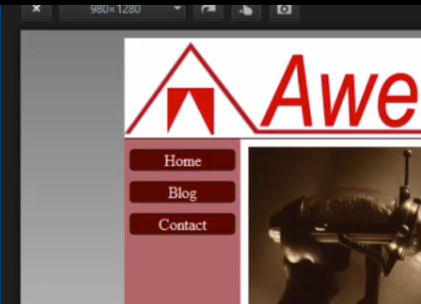
**box-shadow: 5px 5px 5px #555;**

margin: 0 auto;

width: 80%;

}

**Save the style.css file and refresh the index page.**



The bottom seems great, but notice the corner up by the logo: It's not rounding. That's because the background image is sitting over the edge of the logo. We can fix that by telling the wrapper that it has to hide any content that goes outside of the wrapper.

**In the style.css file, add in the following bolded CSS to the desktop media query:**

#wrapper {

….

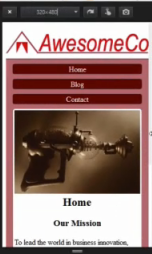
**overflow: hidden;**

}

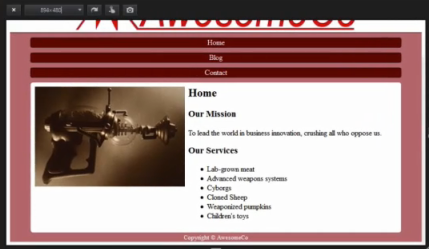
And that does it for this mobile-first design.

**Save the style.css file and refresh the index page.**

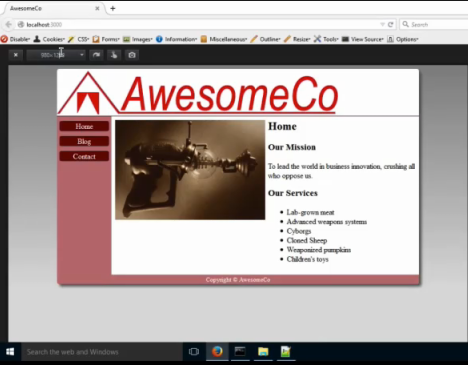
Mobile View:



Tablet View:



Desktop View:



We just added some style, changed the screen size, and looked at what changed, tweaking it as we go. That's what you'll have to do on your own projects. Don't spend time trying to remember exactly how I did this process, because in order for this to work for you, your page markup and content would have to be exactly like this. Instead, focus on the process we used here. We started with a small screen and made it work. And then we made the screen bigger. And then we tweaked what looked funny using media queries. Along the way we made some choices that were the best choices we could make given the circumstances. For example, on the really-really small screens we decided to use CSS to either hide that image or scale that image. We also had to occasionally move rules around to avoid creating too many rules that overrode other rules.

We were fortunate here to be able to start with a clean slate and think mobile first. However, we're not always that lucky. Sometimes we're just given an existing stylesheet and we'll have to make it work on mobile devices too. You'll do it the same way: you'll use media queries and overwrite existing rules. The only difference is that you may have to override a lot more than you would if you went mobile first.

**Final Touches**

There's one more bit of code we have to add to our pages to make this work on mobile devices. We have to add a special meta tag that tells mobile browsers that our site supports a mobile view.

**In each of the HTML files, add in the following meta tag to the head:**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**Save all your .html pages.**



Mobile browsers read this and immediately zoom in and make our mobile version fit, instead of making our visitors pinch and zoom.

Once we add that tag, we can call this mobile-first design finished.

Below is the completed style.css file:

h1, h2, h3, h4 {

margin-top: 0;

}

header > h1 {

background-color: #fff;

background-image: url(../images/awesomeco\_logo\_mobile.png);

background-repeat: no-repeat;

background-position: center;

background-size: contain;

border-bottom: 1px solid #300;

height: 70px;

margin-bottom: 0;

text-indent: -9999px;

}

footer {

color: #FFF;

text-align: center;

}

nav > ul {

list-style: none;

margin: 0;

padding: 0;

}

nav > ul > li {

border: 1px solid #333;

border-radius: 5px;

margin: 10px auto;

text-align: center;

width: 90%;

}

nav > ul > li > a {

background-color: #600;

color: #fff;

display: block;

text-decoration: none;

}

#wrapper {

background-color: #BC7277;

}

main {

background-color: #fff;

border-radius: 5px;

margin: 0 auto;

padding: 1%;

width: 88%;

}

@media only screen and (max-width: 767px) {

main > img {

/\* display: none; \*/

max-width: 100%;

height: auto;

}

h1, h2, h3, h4 {

text-align: center;

}

}

@media only screen and (min-width: 768px) {

header > h1 {

background-image: url(../images/awesomeco\_logo.png);

background-size: auto;

height: 100px;

}

main > img {

float: left;

margin-right: 1%;

}

main ul {

overflow: hidden;

}

}

@media only screen and (min-width: 768px) and (max-width: 959px) {

nav {

text-align: center;

}

nav > ul > li {

display: inline-block;

width: 20%;

}

}

@media only screen and (min-width: 960px) {

nav {

float: left;

width: 15%;

}

#wrapper {

border-radius: 5px;

box-shadow: 5px 5px 5px #555;

margin: 0px auto;

overflow: hidden;

width: 80%;

}

header > h1 {

background-position: left;

}

main {

margin-left: 15%;

border-radius: 0;

width: 83%;

}

body {

background-color: #ddd;

background-repeat: no-repeat;

background-image: linear-gradient(to bottom, #888, #DDD);

}

}