AQUENT

GYMNASIUM

RESPONSIVE WEB DESIGN:
BUILD A PORTFOLIO FOR ALL DEVICES

LESSON 4

ABOUT THIS DOCUMENT

This handout is an edited transcript of the Responsive Web Design lecture videos. There's nothing in this handout that isn't also in the videos, and vice versa. Some students work better with written material than by watching videos alone, so we're offering this handout to you as an optional, helpful resource.

Some elements of the instruction, like live coding, can't be recreated in a document like this one. We encourage you to use this handout alongside the videos, rather than as a replacement of them.

CHAPTER 1: CSS3 MEDIA QUERIES

This is Responsive Web Design, an online course developed by Aquent. Responsive Web Design, or Promote Yourself Responsibly: Build a Portfolio For All Devices. This is lesson 4, CSS3 Media Queries.

As always, at the end of this lesson, there will be an assignment and a brief quiz. At any given point, you may use the Pause button in order to catch what's going on, to follow up on code. As always, if you have questions, feel free to hit the Forum. There, there will be TAs ready to answer your questions as well as me, your instructor. Of course, you also have your other classmates.

Just a brief road map for this lesson. Chapter 1, well, we're currently in it. It's the intro.

Chapter 2, we'll be talking about browsers getting smarter with media queries. There will also be some presentation. There will be some coding.

And in chapter 3, we'll talk about column layout and media queries. In that chapter, there'll be presentations, some demos, and lots more coding. And then in chapter 4, we'll do the next level of media queries-- presentation & demos but no coding. This is Responsive Web Design.

CHAPTER 2: BROWSERS GET SMARTER WITH MEDIA QUERIES

Chapter 2, Browsers Get Smarter With Media Queries. So in this chapter, we'll be talking about the origin of media types in CSS2, to begin with. Then we'll begin to expand your vocabulary with CSS3 media queries. Finally, you'll be creating a print stylesheet using media queries. The objective of this chapter is to gain an understanding of the role and basic syntax of media queries.

So from the beginning, CSS was always looking beyond the screen. There are 10 media types that were defined in CSS2. So here are the 10 media types. And I'm not going to walk through all of them right now. We'll talk about the most important ones in just a moment.

The point I want to make here is that in 1998 is when these 10 media types were established. So they've been around for quite some time. And if we were to look at all of these, which are the ones that you are most likely to have worked with? And again, I'm assuming that you're a front-end developer or designer.

Chances are you've worked with the "all" media type. Perhaps the "print" media type, as well as the "screen" media type. So these other seven media types are really not so common.

However, there is one here that's very intriguing, "handheld". Yes, there was a "handheld" media type defined in CSS2. We're going to come back to that in a little bit.

But as I mentioned before, "all", "print", and "screen", these are the three most likely media types that you've worked with. Let's just define these very quickly in case you haven't.

So media="all". This is the default media value. Media="screen". This tells the document to render a stylesheet for a color computer screen. Media="print". Well, this is the media feature intended for page material or documents viewed on the screen in the Print Preview mode.

Now I've highlighted "print.css" here because the point is, we've got a linked stylesheet called print.css and this will only be used if a printer is detected.

Now, we don't have to have it formatted as a link stylesheet. We could also have an internal style and the syntax here is @media print and then everything in the curly braces is the CSS that you'd like to use.

In fact, the @media rule is an oldie but goodie. It's been around for a while. So if we were to take this code, for example, @media print and then body, color:black, background:#fff.

First of all, the syntax is interesting, right? We have a nested curly brace inside of the whole @media print section. And what would the stylesheet do? Well, essentially, when the document was printed, the text would be black and the background would be white. So this is an extremely simple print stylesheet.

We could make this a little more sophisticated. So we could set a rule for the heading 1, and we could say background:none. And that would override any background that we might have in heading, which could be very useful if you're printing.

TEN MEDIA TYPES DEFINED IN CSS 2.0

all	print
aural	projection
braille	screen
embossed	tty
handheld	tv

```
<LINK rel="stylesheet" type"text/css"
      href="styles.css" media="all">
```

Default value. Not necessary to define

```
<LINK rel="stylesheet" type"text/css"
      href="print.css" media="print">
```

Intended for paged material and for documents viewed on screen in print preview mode

```
@media print {
  body {
    color:black;
    background:#fff;
  }
}
```

And finally, we have some print-specific styles. So, "page-break-after: avoid;" is something that you would really only use in a print stylesheet. So in many ways, print styles are responsive. We don't necessarily think of them this way, but one way to frame it is if the web browser understands that a printer is there, it's going to respond by sending it the print stylesheet.

So in many ways, it is responsive in the broadest sense of the word. We're going to explore this a little bit later. But let's do more exploring. Let's look at some of those other media types that have been around for a while.

And in particular, let's look at handheld. Because for our purposes, this is really one of the most interesting. Again, the code looks something like this. You could have a stylesheet called handheld.css and the media="handheld". And this means that if the browser detects a handheld device, then send it this stylesheet.


```
<LINK rel="stylesheet" type"text/css"
      href="handheld.css" media="handheld">
```

Intended for handheld devices

Lesson 4 of 12: CSS3 Media Queries

In reality, this media type is actually very limited. The reason why is we can have that stylesheet, and it will send those styles to a number of common devices. However, over here on the right, we have our friend the iPhone. And guess what? The iPhone did not ever recognize or use handheld media types.

And the reason for this is somewhat complicated. We could get into this for another half-hour, but the point is it didn't recognize it. Instead, it's going to be using media queries.

Now, what this means is you don't necessarily have to ignore the handheld media type all together. There are some devices that support it. And if maximum mobile support is a goal, then you can't ignore it.

But we're not really going to be diving too much into the handheld media type in this lesson. The truth is because of the lack of support from the iPhone and other mobile devices, that handheld media type never really got a chance at the spotlight.


```
<LINK rel="stylesheet" type"text/css"
      href="handheld.css" media="handheld">
```

Lesson 4 of 12: CSS3 Media Queries

Because awaiting there, right outside the stage, we have our friend the World Renowned CSS3 Media Query. And the truth is, our friend the iPhone had a lot to do with the fact that the CSS3 media query is well-supported.

Media queries. They're all about conditions. What do I mean by that? Well, you use a media type and expressions to check for the conditions of particular media features. Might be a little more clear if we take a look at an example. So here is a media query being used for an external style, Media="screen and (color)". Here, the media type is screen. Color is the feature. And the operator, in this case, is and. So the way to translate this is, example.css will be used if this condition is true. A screen is detected and its color.

Now as I mentioned before, media queries can also be added to your stylesheet. So we don't have to point them to an external stylesheet. They can be nested inside of your internal styles. So that same media query could be expressed like this, "@media screen and (color){". And what's the style? Well, all of your heading 1's will be the color chartreuse. This will work.

To make all this easier, I have invented a device called the Gymnasi-Tron 3000. The way it works is you feed it a media query, and you get plain English in return. Let me show you how it works.

Here we'll go ahead and feed a media query into the Gymnasi-Tron @media screen and color. So this is the one we just took a look at. If a color screen is detected, then use this style.

So let's go ahead and try another one for the Gymnasi-Tron. In this case, we'll say @ media print and width 8.5 inches. So how does this translate? If the page is printed and the width of the page is exactly 8.5 inches, then use these styles.

So that media query could be useful in the following circumstance. Here, we have a web page I'd like to print. Because a letter size document is typically 8.5 inches, we'd be good. The styles would apply.

If for some reason, however, the page we were printing on happened to be more or less than 8.5 inches, that print stylesheet would not work. So would that ever happen? Absolutely. In the US, letter-sized paper is 8 and 1/2 inches wide. But standard A4 letter paper in Europe translates to about 8.27 inches or you could technically print on 11 by 14 or some other size.

So the question is, can media queries handle those types of variable situations? And luckily, the answer is yes. We have prefixes that we can add to the width feature to modify the conditions of the media query. So let's take a look at those.
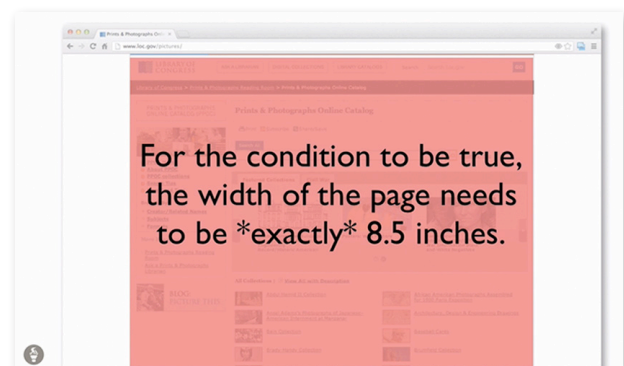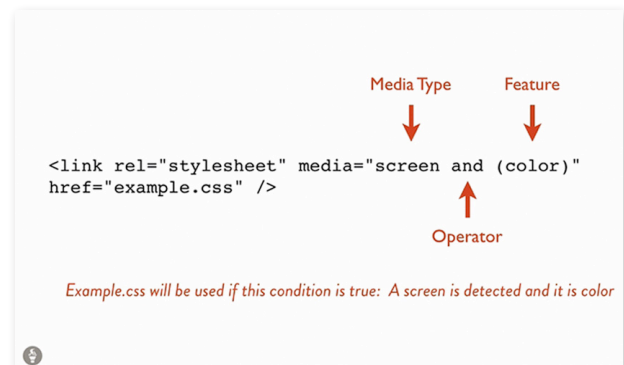
The min- and max- prefixes are going to be your best friends. "Min-" means greater or equal to. "Max-" means less than or equal to. So examples of the "min-" prefix would be min-width, min-resolution. Examples of the "max-" prefix would be max-width, max-resolution.

THE "MIN-" AND "MAX-" PREFIXES

- "min-" means *greater or equal to*
  - min-width
  - min-resolution
- "max-" means *less than or equal to*
  - max-width
  - max-resolution

Lesson 4 of 12: CSS3 Media Queries

So let's go back to our friend here and let's put this one in. @media print and min-width 8.5 inches. That translates to when the page is printed, and the width of the page is greater than or equal to 8.5 inches, then use these styles.

On the flip side, we have @media print and max-width 8.5 inches. When the page is printed, and the width of the page is less than 8.5 inches, then use these styles.

These are going to be much more useful because they express a wider range of conditions than just the explicit 8.5 inch. So let's put all this into practice. And we're going to make a print stylesheet in five minutes or less.

Before you start coding, here's a preview of what you'll be doing. So this is your target. When printed, you want to put all the information onto a single page, not three pages.

The way you're going to do this is to make two columns with the portfolio images on the left and the resume information on the right. And in order to make everything fit, you're going to need to scale down the current size of the images and make a few other adjustments. Sound good? OK. Let's start.

First of all, you're going to open your document, portfolio_start.html. Choose File, Save As, and let's rename this portfolio_work.html.

After you've done that, let's go ahead and preview it in our browser. And I do want to point out I'm previewing in Google Chrome at this point. And the reason I'm previewing this is because we're going to be seeing a print preview in a second. Not all browsers have the same sort of print preview. Chrome happens to have a very good one. You can test it in your browser of choice, but let's take a look at what Chrome does.

So we'll choose File, Print, and we'll get this preview. So what I want to point out here is this is based on the lesson file that we used at the end of last lesson. And we can see some interesting things are happening.


There's this big, wide swath of space below the heading, Hi, I'm Maya. And we can also see that the images are fairly big. They're a single column. They're spaced out. This is really not an ideal printed format.

So if someone's coming to your portfolio page and they're printing it out, this is what they would see. First of all, it's three sheets of paper. That's a lot. Let's go ahead and think about how we can begin to make a better printed stylesheet.
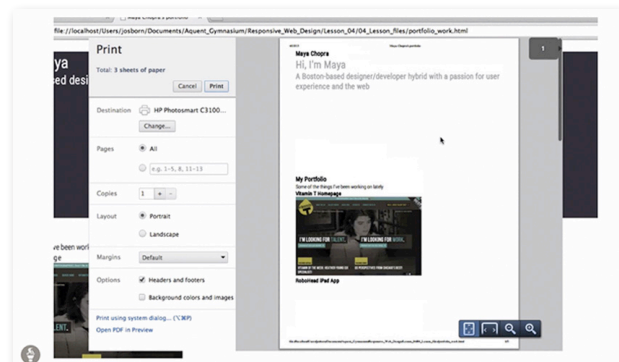
The first thing we need to do is add a little bit of structure to our page. So I'm going to go down here to the first section inside the portfolio section, and we're going to say id='portfolio'. And then down here, we'll say id="resume" for this resume section. So we have a portfolio id and a resume id.

Let's go ahead and go to this document, printstylesheet.text. So this is a plain text file that's inside your lesson folder. You're going to want to open that up and then copy the first line, which is the media query that I have already added for you. So @media print and max width 8.5 inches.
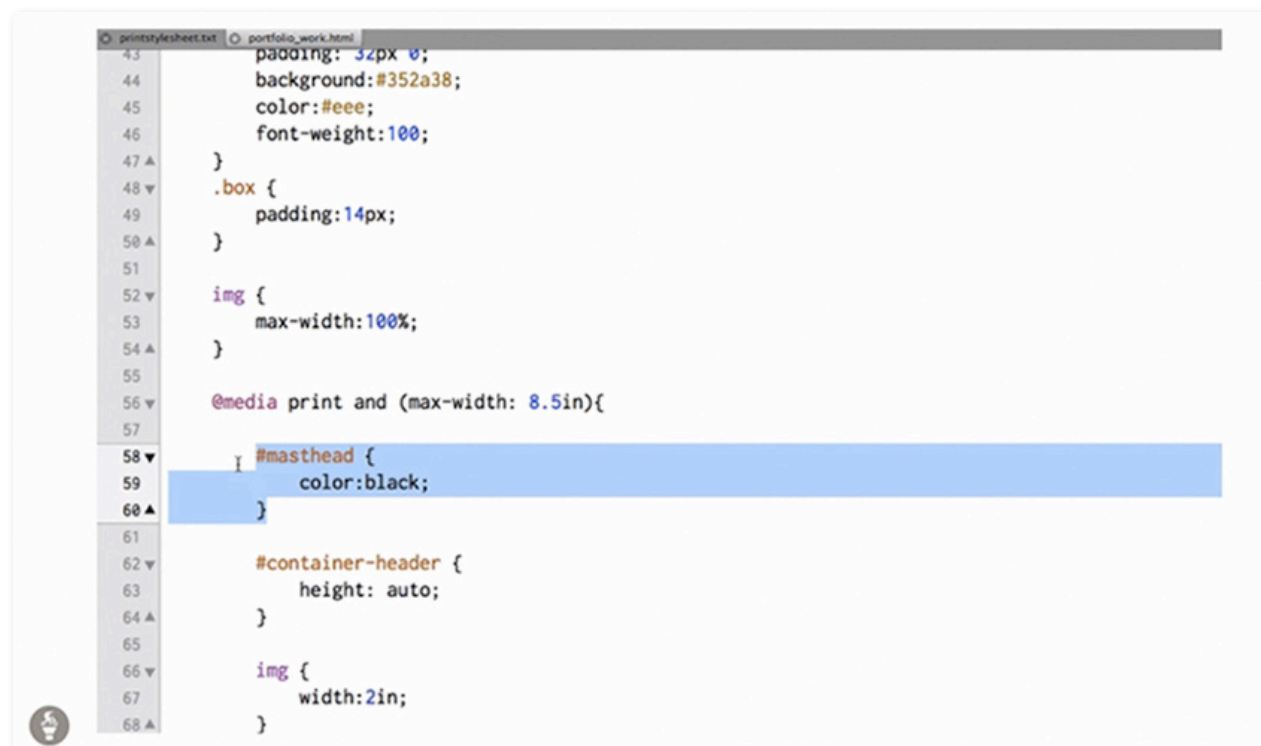
So this is the one that we saw before. Do you remember what it means? It means less than. So we're actually creating styles for anything less than 8.5 inches.

Let's go back to our document, paste this at the very bottom of our stylesheet. Make sure we put in that curly brace. And now we're going to steal the first half of these styles, so masthead, container header, and image. Let's copy those.

Let's go back to our stylesheet and paste them inside the media query. So we have masthead, container header, and image. Masthead, the color is black. Container header, the height is auto. And image, the width is two inches.

Now typically, we don't use inches on the web, but we can do it for a print stylesheet. Makes a lot of sense. Let's just go ahead and look at what these are overriding.

So the original image is a max width of 100. Now it's two inches. The container header had a height of 350. Now it's auto. And the masthead had a color of white, and we're going to make it black.

Let's go back to the browser. Reload. Take a look at this now. And we have collapsed that space below the heading. The heading is now black. And notice that those images are much smaller and everything re-flows. It's now a much tidier print

document. And it's only two pages at this point, not three.

However, we still have a little bit more work to do. So let's go back. And now let's make this two columns.

So I'm going to go to our text styles here. I'm going to copy the last three styles. And let's paste these. Make sure it's inside that media query, obviously.

And we have portfolio and resume. These are both basically doing the same thing. One is going to be floating left. The other is going to be floating right, using a width of 45%. That's going to give us two columns.

And then we have to clear the footer underneath it, or else that's going to create some visual problems. So these are the three styles. Let's go ahead and save it. Reload. Print. And there we have it.

We actually now have a pretty nice and easy to do two-column print stylesheet. And we've done this all with media queries.

Print stylesheets were responsive before responsive was cool. That's right. Print stylesheets have been around for over 10 years, and so we can take some of the concepts that we learned here and apply them to true responsive design when it comes to layout.

If you want to know more about print stylesheets, here is an advanced document that gives you some great tips and tricks. This link will be in the classroom as well.

Next, we apply media queries to layout. This has been Responsive Web Design.

CHAPTER 3: COLUMN LAYOUT & MEDIA QUERIES

In this chapter, we'll be taking a look at column layout and media queries. Specifically, we'll be covering the range of the @media feature, how to use media queries for layout, taking a look at more sophisticated media queries using logical operators, and finally, adding media query styles for a three column desktop layout.

In the last section, we took a look at a print stylesheet using media queries, and that created a very simple two column layout that will improve the user experience whenever they print out our portfolio page. Now, here's a little pop quiz. How would you create a stylesheet that targets only large screens, or screens of a certain dimension, or certain width? Let's go ahead and plug this into the Gymnasi-tron.

WHAT WE'LL BE COVERING

- ✦ Examining the range of @media feature
- ✦ Using media queries for layout
- ✦ More sophisticated media queries with logical operators
- ✦ Adding media query styles for a three column desktop layout

Lesson 4 of 12: CSS3 Media Queries

@media screen, and min-width, 1024 pixels. Take a second. Can you interpret what this will do? Remember that min-width essentially means greater than. Use these styles when there's a screen and the width is greater than or more than 1024 pixels. Now, the question is what other features can we target? Right now, these are the media features that are supported in CSS3 by most browsers, or at least most modern browsers.

MEDIA FEATURES IN CSS3

color	grid
color-index	height
aspect-ratio	monochrome
device-aspect-ratio	orientation
device-height	resolution
device-width	width

Lesson 4 of 12: CSS3 Media Queries

Of all of these, we have one that we've used so far. Width. But there's a lot of other interesting ones here. Color, device aspect ratio, grid, monochrome. We'll take a look at some of these, although not all of them. The one that strikes me as being very interesting is device-width. So device-width versus width. What does that mean? So to wrap your head around this, let's start with simply understanding width.
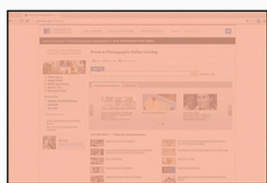
So width would refer to something like a web browser, and as you know, you can change the width of your web browser by simply dragging it to the right or left. In this case, I've chosen a random number 1221 pixels wide. That happens to be the width of the browser, or the document as I've currently set it. But it could be 1000, could be 900, could be whatever. Device width really refers to a device, something identified as hardware, typically a smartphone or tablet, but it could be anything, a screen in a car or a futuristic watch.

WIDTH VS. DEVICE-WIDTH

1221 pixels wide

Lesson 4 of 12: CSS3 Media Queries

DEVICE-WIDTH USAGE

1440

480

320

900

@media screen and (max-device-width: 480px)

Lesson 4 of 12: CSS3 Media Queries

Devices tend to have fixed resolutions, 320x480, or 640x960. Whatever it is, we have a specific device width that is built into the resolution of that screen. Now where it gets interesting is mobile browsers typically do not allow you to change the width of the browser window. So oftentimes, the width and the device width for smartphones and tablets-- well, I guess tablets are a little different, but for smartphones, well, they're generally the same.

Device-width and device-height can actually be very useful to target specific devices, because of that fixed resolution. As an example, let's take a look at a media query that targets 320x480, or more specifically, `@media screen and max device width 480 pixels`. So whatever stylesheet we create here will only target a device that has 480 pixels or less. And typically, desktop monitors won't even come close to that.

So you can be pretty safe in using this for a targeted style for a smartphone. Now, all of these media features look very interesting, but not all of them are supported, and some are used more often than others. I'm going to try to give you a subjective view of which ones those are. In the green box are the ones you're most likely to encounter. Width, height, orientation, device height, and device width.

On the right we have some of the lesser-used ones. Some of these are not even particularly well supported, such as grid. But the truth is, width is going to be the one you use most often, followed by height, and maybe orientation. Now, you can get even more specific with media queries by creating chains. Get that? Chains with the "and" operator. And we've already used one of those.

So again, let's zoom into our Gymnasi-tron and look at a chained media query. @media screen and max device-width 480 pixels, and orientation portrait. We translate that to use these styles when there is a screen, and the width of the device is less than 480 pixels, and the screen is in portrait mode. Now, device width and device height can be useful, but there's a little catch.

And one of the reasons that they are not as popular as they may have been a few years ago is we've got a lot more devices out there. So if you're trying to target specific devices with device-width or device-height, you can do it, but you might be excluding others that are slightly different sizes. So let's return to code. We want our layout to change once the screen width allows for at least two projects from the portfolio session to sit side by side.

So that's the visual appearance that we want. We just have to translate that to code. Specifically, you'll be defining some breakpoints, or at least one breakpoint. You're going to be adding some structure to your HTML document to define the portfolio and resume section elements. We're going to be using some CSS to make three columns. We also have to take care of layout issues that we might run into with a clear fix.

WHAT YOU'LL BE BUILDING

+ Define breakpoints
+ Add some structure to the HTML to define the portfolio and resume section elements
+ Add our CSS to make 3 columns
+ Provide fixes for any potential layout problems with a "clearfix"
+ Provide support for Media Queries for older versions of IE

Lesson 4 of 12: CSS3 Media Queries

And finally, we have to provide support for media queries for older versions of IE. Ready to get started? Make sure you've got your text editor up and running, and let's begin. Now, we can start with the same document that we worked with before. Let's go ahead and quickly take a look at this in our browser.

If we change the width of the browser window, we can see there's a little bit of responsiveness, and that's just the text that's re-flowing, and the max-width images that we have will tend to scale a little bit as they get smaller. In other words, for our smartphone layout. So again, we're going to be creating a three column layout here. We have some structure that we added already, but we have to add a little bit more.

We're going to be using the article tag to wrap around each instance inside my portfolio. So the first article section goes here, right before Vitamin T homepage. The second article will be wrapped around the Robohead iPad app section and the accompanying image. Later on we're going to have to go back and clean up some of this code, but for now the indentation is a little rough.

But let's go ahead and add the third article, and we're going to put that around the Shareist homepage section. Now that we added that structure, we have to put in some styles. So the styles here will be portfolio article float left, width 33%. So we want three columns, and we're going to be applying that here. And there we go.
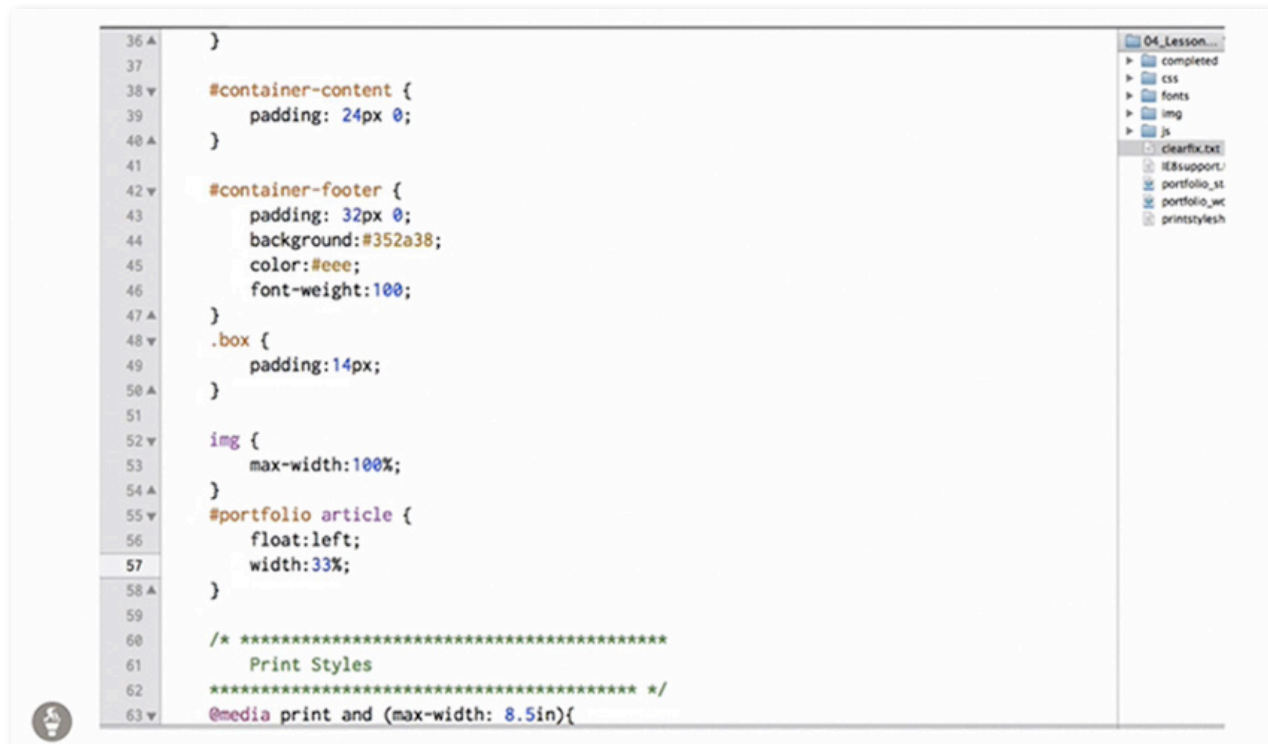
We now have a very nice three column layout, and that's going to scale all the way down to the size of our mobile page. So, that worked so far. But before we get all media queried, we have to do a couple of other things, some housekeeping here. This works great if we only have three objects for that three column layout.

So I'm going to go ahead and copy and paste the third element to make a fourth one, and the question is what does that do to the layout if we add a fourth item to my portfolio? So let's go ahead and reload our page, and the answer is it doesn't do very good. It's going to create some floating and clearing problems. So to fix that, we're going to use a concept that you may have used before called clearfix.
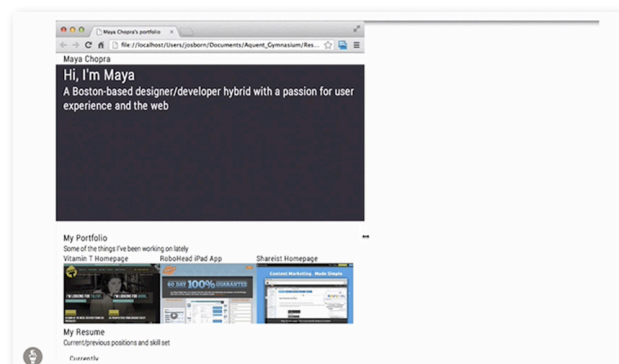
What this clearfix is going to do is make sure that whatever is below our floated elements does not jog up, like we saw there in the last example. So, open up your clearfix.txt file, and just copy all the code in it. And we'll go ahead and paste that into our styles. And we have to add a little more structure after we do that. But let's go ahead and paste that entire block of code, and just a note about clear fixes.

It's likely you've used them in other projects. They're pretty common in modern web development. Clearfixes have evolved over the years, and this one is called the "micro clearfix hack". In this particular instance, the class name "group" is used, but the concept is the same as other clear fixes. The styles for group and group after are designed to fake the web browser into thinking that there's content on the page, but there really is none.

By applying the hack to a group of elements, we ensure that there will not be other content sneaking up on the sides where we don't want it. In this case, we'll add the group class to our portfolio section. So class equals group. Let's go ahead and reload.

```css
img {
    max-width:100%;
}
#portfolio article {
    float:left;
    width:33%;
}

/* Float Clearing http://nicolasgallagher.com/micro-clearfix-hack */

.group:after {
    content: " "; /* 1 */
    display: table; /* 2 */
    clear: both;
}

.group {
    *zoom: 1; /* 3 */
}
```

And it takes care of that clearing problem. Now, that fourth item that I added was just for demonstration. Let's go ahead and remove that now.

But we can now rest assured that our code will work if we ever happen to add more. Let's go ahead and do the same thing. Let's add a class named "group" to the resume section. And just like we did with the portfolio section, we're going to wrap articles around the three boxes here. The first box is for "currently", and the second box will be for the "previously" sections. And the third one will be for the 'skills" section.

So really, we're just repeating the structure that we did earlier. We're adding a group, we're adding these three articles. The only thing left to do is just to make sure that we close up the loose ends and add that style to the one we had earlier. So, `portfolio article`, let's put in a comma, and we'll call this `resume article`. Exact same structure. Let's reload our page, and there we go.

We're getting a really nice layout. It's going to be three columns all the way from the desktop down to the mobile. And that's where we're going to have to change it, because clearly we need some media queries. We don't want our mobile layout to have three columns. We want it to use the original single column layout that we carefully created in the last lesson.

So to do that, we're going to steal the stylesheet that we used for the print styles, and then we're going to go ahead and paste it right up here. And then we're going to do a little tweaking. Instead of @media print, we're going to change that to screen. Instead of max width, we'll make it min-width. And instead of inches, we're actually going to change this to pixels. 720 pixels. So now we don't have the Gymnasi-tron at our side.

Can you interpret what this is going to do? Min-width really means greater than, so we're creating styles that will be applied if the screen is greater than 720 pixels. OK? Make sure we put in that

closing curly brace. Reload your page. It's still going to work here. As soon as we hit less than 720, it's going to flip back to our original styles. So this is pretty cool. Right there is our breakpoint, and we're going from three column to single column.

Because we started mobile first, we've got a nice single column layout. OK. Now unfortunately, we have to do a couple of other things to put in support for other browsers. If you open up this document, IE8support.txt, we have a bunch of code in here, and we're only going to be using half of this code. Option one is the code adding media query support for IE8 and under.

So IE8 doesn't even understand what media queries are. Doesn't support them. So what we have here is a JavaScript called respond.js, and this is essentially going to force IE8 to understand what media queries are. We can use this one, and in fact, we will use this one. If you do use it, you have to put your media queries in an external stylesheet. The second one, option two, adds media query support, but it does it using a conditional comment.
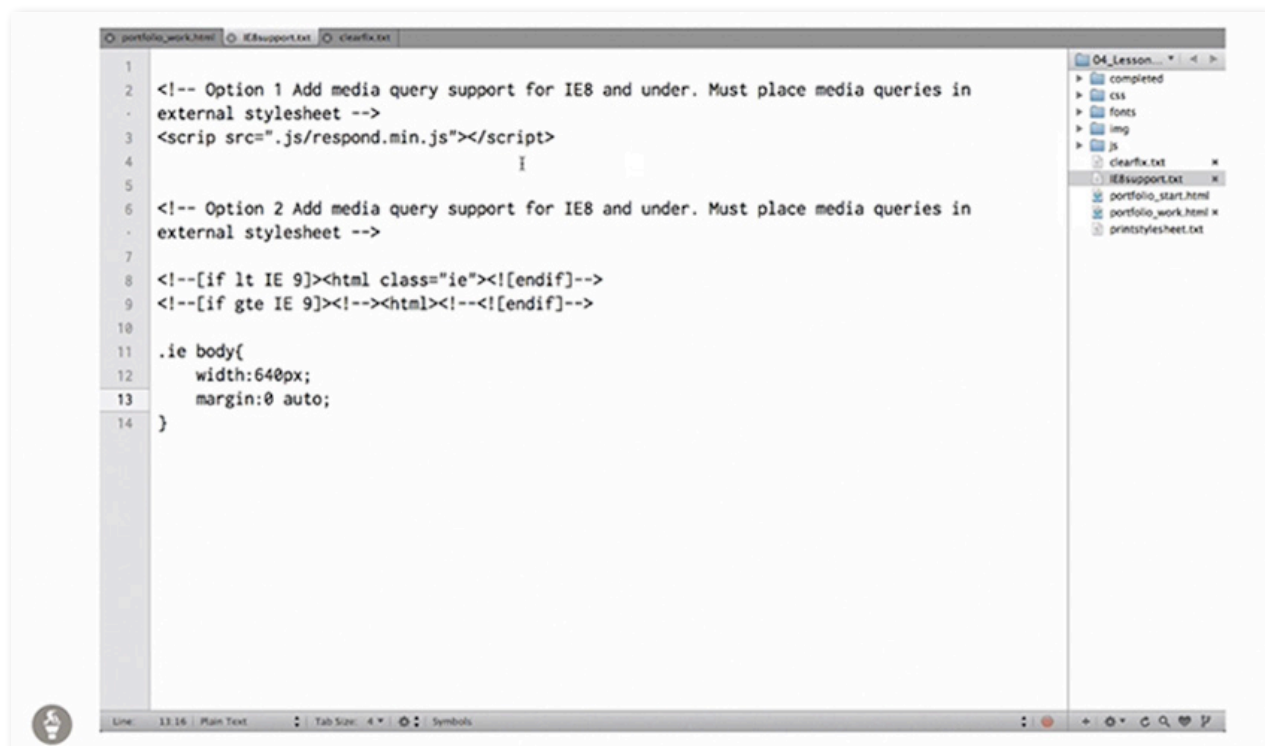
So again, some organizations like to put IE specific code inside of conditional comments, rather than using JavaScript. If that's your situation, then use the second option. We're going to use the first one, however. So let's copy all this code here, and we're going to scroll up a little bit. We'll have to put this in our head section. And let's put that right here between these other scripts.

And we'll paste there. Now, because this solution requires us to put our media queries in an external stylesheet, well, we've got to put our media queries in an external stylesheet. Let's go ahead and cut that entire media query there. Open up style.css. Paste it at the very bottom. For good measure, I'm going to put in some comments so we can label what's happening here. Media queries for screen widths wider than 720 pixels. Let's save that. Let's go ahead and reload, just to make sure that it's all nice and happy. And there it is.

Now, there's clearly a little bit more that we could do here, but this is a great foundation. Let's go ahead and talk about what we need to do. So one thing you may be noticing if you're doing this on your own, or you may have seen it in the video, is that the distance between the three images here gets a little squishy when we have this screen set to the widest length.

And that gutter, as we're calling it here, actually changes. It's currently a percentage. Later on, we're going to want to address that, so that we might have these static gutters, these gutters that but always stay the same width until we flip over into our single column. So that's something that we have to resolve, the squishy gutter columns.
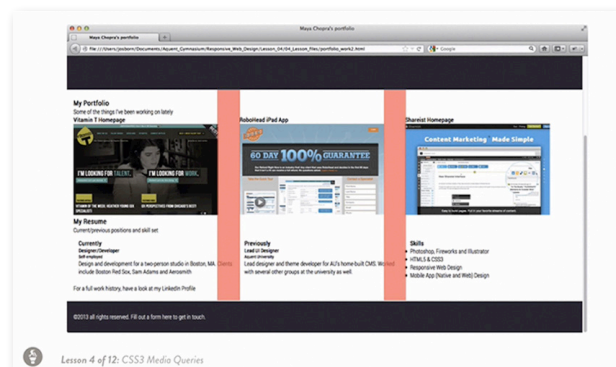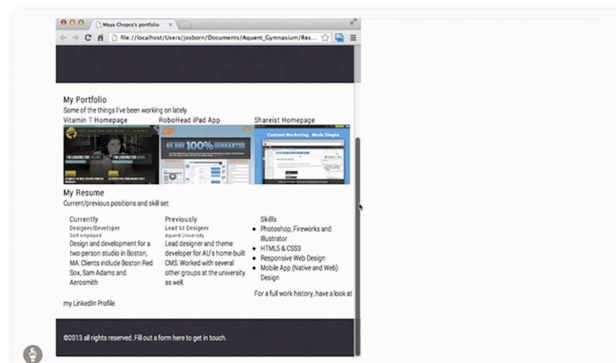
We also have to come back and address the spacing and the styles of the text. So margins, padding, font size, and so forth. We're going to be adding additional images, but for now we're done adding our media queries. This has been Responsive Web Design.

CHAPTER 4: MEDIA QUERIES: THE NEXT LEVEL

In this section, we take media queries to the next level. We're going to be using and talking about more sophisticated media queries. Now, we won't be applying these into the code. We'll just be taking a look at some examples.

To do this, we need additional operators. These are technically logical operators-- and, or, not, and only. These are operators that we can add to our media queries to make them more complex and to do specific things.

So we've already looked at the and operator, but let's just look at it again. Here is a style-- @media (min-width: 700 px) and (orientation: landscape). So use these styles if the viewport is 700 pixels wide or wider and the display is in landscape.

OK, let's take a look at another operator. In this one, we have a comma. So this is called a comma-separated list. So the comma is right there after (min-width: 700px), handheld and (orientation: landscape). That comma really is interpreted as or, so use these styles if the viewing device either has a minimum width of 700 pixels or is a handheld in landscape. So this, again, expands the range of scenarios that we can use for our styles.

And then we have this one-- @media not screen and (color). So not is the one we're interested in here. In this case, we translate it as, use these styles for anything that's not a color screen, ie. non-screens or monochrome screens.

Finally, we have this one which I'm going to be putting in an external style. media="only screen and (color)", and then we have a stylesheet. This one, the only operator means use the styles only for devices that are color screens. And if you're an older browser that doesn't support these media queries, move along. Nothing to see here.

What does that mean, really? It just means that older browsers will ignore this only keyword. And we can use these styles without worrying about having those styles apply.

So again, we briefly touched upon some of these media features. We're not diving deeply into what each of these mean. As I mentioned before, width, height, orientation, device-height, and device-width, these are the ones that you should become most familiar with. The others, again, are useful, but we won't be using them on a daily basis unless you've got a particular project.

But wait, there's actually more. There's always more. This is the web. Everything is always growing, evolving. There's always more.

In this case-- browser-specific media queries. Not only did we have that simple list of features, but many browsers also have their own media queries that will only work in their browser. To give you an example of how this manifests itself, let's take a look at one of these. @media screen and (max-device-width: 480 px) and (-webkit-min-dev ice-pixel-ratio: 1.5)

So this is why we need our Gymnasi-tron. This is what it translates to. Use these styles when there is a screen and the width of the device is less than 480 pixels and when the screen pixel ratio is 1.5.

Why do these exist? Why might you use these? What is a screen pixel ratio of 1.5 anyway?

That might be a device such as this one. So high-resolution screens are something that we might want to target. So the iPhone, for example, instead of 1.5, actually has a pixel ratio of 2. So the iPhone 4 and above, we could target that with something like (-webkit-min-dev ice-pixel-ratio: 2). So there are certain scenarios when you might want to target specific devices as we saw earlier.

For the next lesson, you're going to start learning how to optimize type on the web. We're going to be looking deeper into concepts such as line-height, line-length, font-size, and baseline grids.
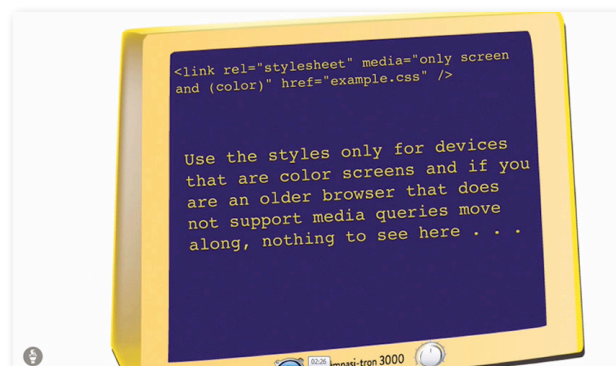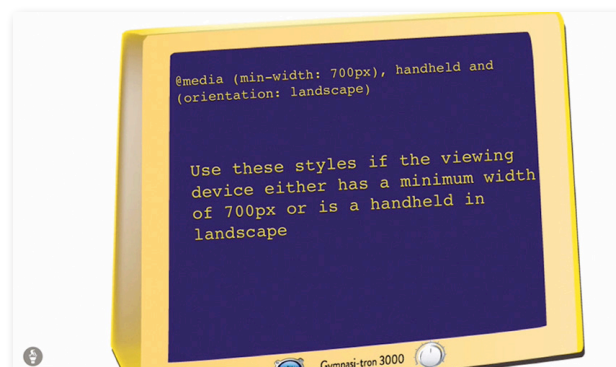
You do have some homework, however. So as always, assignment number 1 is a short quiz which you can take on our site. Assignment number 2-- I'd like you to create a print stylesheet of your portfolio in progress. You can follow the steps in the portfolio and embellish as needed depending on your content. For more advanced techniques, I encourage you to look at this article or others.

Assignment number 3-- just like we did in the second half of this video, add media queries to your portfolio in progress. Use similar breakpoints to the ones that we covered in the lesson. In fact, there was only one breakpoint. But in any case, I want you to add media queries to your portfolio page.

There's a little bit more information on the classroom about these assignments. But for now, I'll see you in the Forum, and I'll see you next session.

Assignment #2

CREATE A PRINT STYLESHEET OF YOUR PORTFOLIO-IN-PROGRESS

FOLLOW THE STEPS IN THE TUTORIAL AND EMBELLISH AS NEEDED.

FOR MORE ADVANCED TECHNIQUES SEE THIS ARTICLE FOR INSPIRATION:

HTTP://CODING.SMASHINGMAGAZINE.COM/2013/03/08/TIPS-TRICKS-PRINT-STYLE-SHEETS/

Lesson 4 of 12: CSS3 Media Queries

Assignment #3

ADD MEDIA QUERIES TO YOUR PORTFOLIO-IN-PROGRESS

USE SIMILAR BREAKPOINTS TO THE ONES COVERED IN THE LESSON.

Lesson 4 of 12: CSS3 Media Queries