AQUENT
GYMNASIUM

RESPONSIVE WEB DESIGN:
BUILD A PORTFOLIO FOR ALL DEVICES

LESSON 2

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: SETTING THE STAGE WITH HTML5 AND CSS3

This is Responsive Web Design, an online course developed by Aquent. Responsive Web Design or Promote Yourself Responsibly: Build a Portfolio for All Devices.

My name is Jeremy Osborn. I'm the Academic Director of Gymnasium, and we're here to talk about lesson 2: Setting the stage with HTML5 and CSS3. Now, as always, there'll be an assignment and a brief quiz at the end of this lesson. Be sure to use the pause button at any time in order to catch up with what we're doing. And if you have questions, please hit the Forum. This is Responsive Web Design.

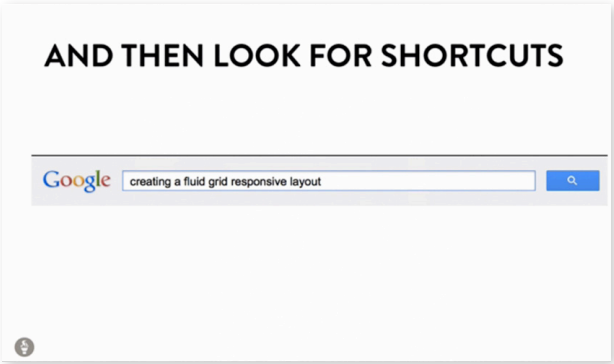
In this section, we're going to be talking about the challenges of responsive design. As we talked about in the last lesson, responsive design can be categorized as having three components: fluid grids, flexible media, and media queries.

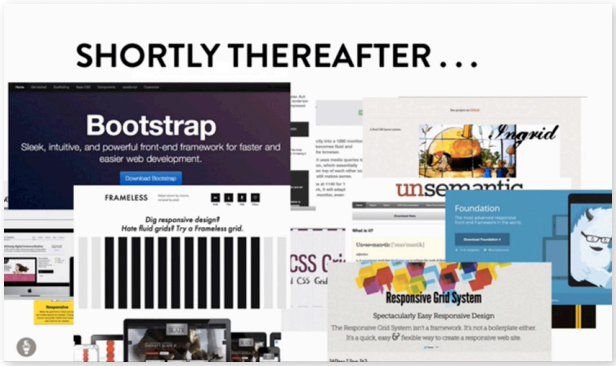
Now, it's worth pointing out that responsive web design is actually very challenging at first. It does get easier. I promise. But what I'd like to do is explain why it's so difficult.

So I'm going to put myself in your shoes. I'm not going to make any gender assumptions here. So I might be wearing these shoes as well, but I'm assuming that you're a front end web developer. And you're going to start at the beginning, and you're going to go online. And you've hopefully read this article, "Responsive Web Design" by Ethan Marcotte, published in 2010.

So you go ahead, and you read the article. But it leaves you wanting more. So you're going to look for shortcuts. You'll go to our friend Google, maybe type in a search phrase such as this. And here's the result. You're going to get one, two, three, four, five, six, seven-- you're going to get many results.

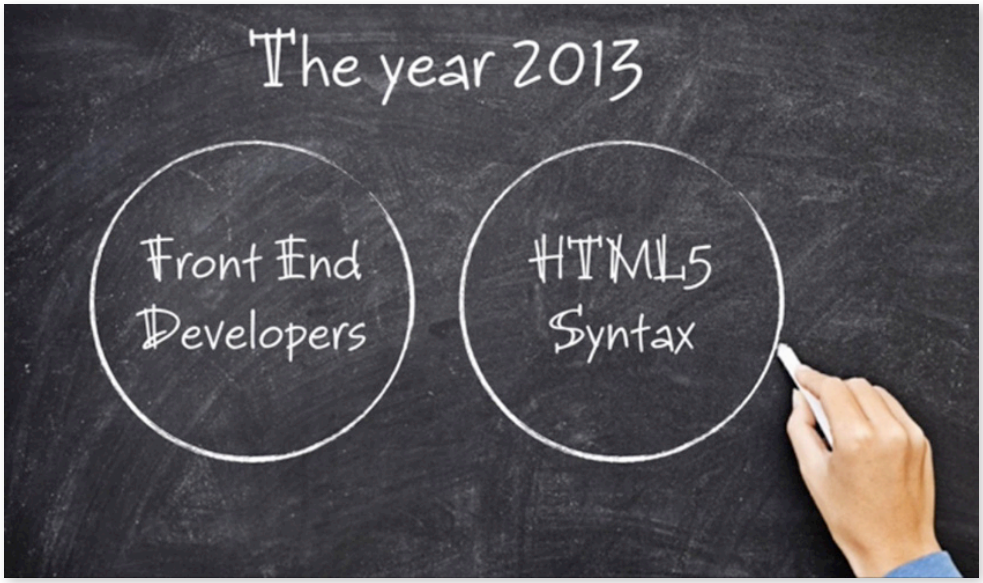
Now, these results may be a bit bewildering at first, because you'll find frameworks, templates, boilerplates, and even systems that claim not to be frameworks or boilerplates. They all promise the same thing, which is to make building responsive sites easier. So what's the deal here?





Don't get me wrong. I love shortcuts. And essentially, this is what they're promising you. The issue here is that frameworks and templates often assume a certain amount of knowledge on your part. My goal is not to make too many assumptions about you. And let me make that a little more clear.

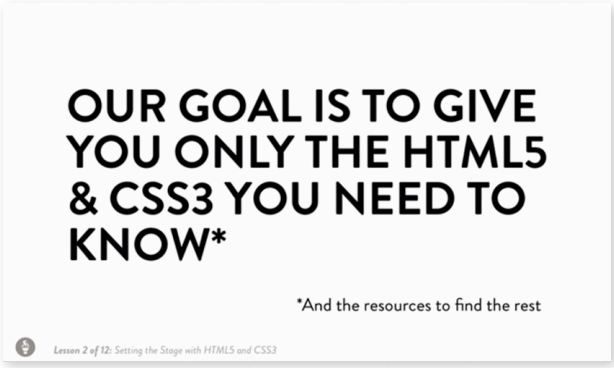
So here we are, the year 2013. And if you remember back to grade school perhaps, we have the Venn diagram. In the left circle, we have front end developers. In the right circle, we have all the people who know HTML5 syntax. And the question is, do these completely overlap? Well, chances are they don't. So you might know HTML5 syntax, and you might not. But we're going to make sure that everyone's on the same page by discussing some of the basics.



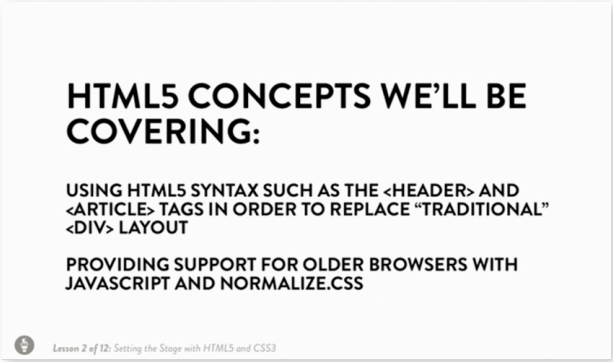
It's also worth pointing out that HTML5 syntax is not required for responsive design, technically. However, there's an increasing expectation that you, as a front-end developer, not only know HTML5, or at least the basics, but that you begin to use it depending on the project.

So turning back to the responsive frameworks I mentioned earlier, these can be great resources and timesavers once you know what you're doing. However, in this class, you're going to be building a responsive layout from scratch. Doing this will be of tremendous value to you if you need to upgrade to something more powerful. By the end of this course, you'll find a number of frameworks and other systems waiting for you at the finish line. And you'll be in a great position to evaluate whether you need one at all. And if you do need one, how to choose the best one for your purpose.

So our goal is to give you only the HTML5 and CSS3 you need to know and the resources to find the rest. Specifically, let's take a look at what we'll be covering throughout this lesson. You're going to be using HTML5 syntax, such as the header and article tags in order to replace traditional div layout. You're also going to be providing support for older browsers with JavaScript and something called normalize.css.



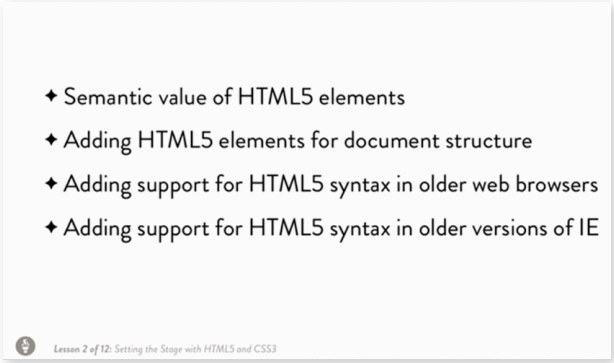
And when it comes to CSS3, we're actually not going to be diving too deeply into these concepts. We'll be looking at things like RGBA colors, text shadow, box shadow, and, of course, media queries, which were a huge component of CSS3. And conveniently enough, they also happen to be a huge component of responsive design. So we'll be spending a lot of time with media queries. Does that sounds good? Excellent.



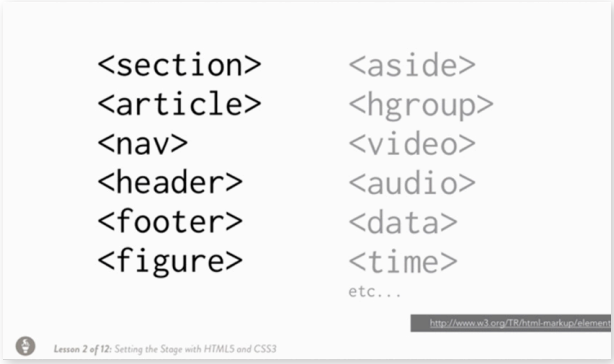
So if it doesn't sound good or if it feels like you know all this stuff, feel free to skip the two remaining chapters in this lesson and move on to lesson 3. I do have to tell you, you will be missing out on the pie if you skip ahead. So stay around for the pie. You might be learning something also. This is Responsive Web Design.

CHAPTER 2: WHAT YOU NEED TO KNOW ABOUT HTML5

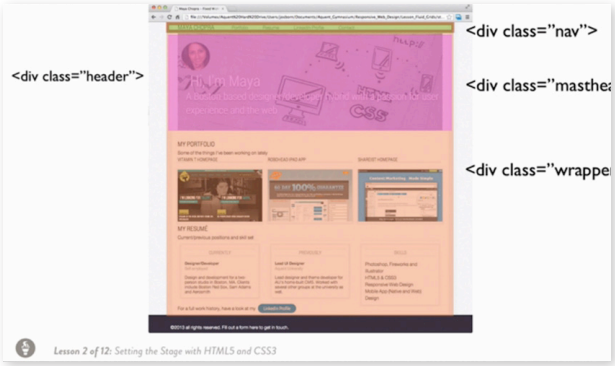
Now, in this section we begin to talk about what you need to know about HTML5. We're going to be covering a few different topics throughout this section, so let's just specifically discuss what we're doing. First, we're going to talk about the semantic value of HTML5 elements. Then, we'll talk about adding HTML5 elements for document structure, adding support for HTML5 syntax and older web browsers, and finally, adding support for HTML5 syntax in older versions of Internet Explorer.



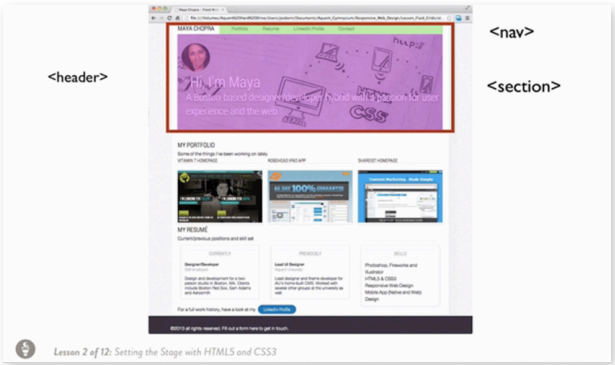
So to cut to the chase, here are the HTML5 elements you absolutely need to know, at least for this lesson are <section>, <article>, <nav>, <header>, <footer> and <figure>. On the right-hand side, in gray, are other HTML5 tags. You don't really need to worry about these, at least for this lesson. They do exist. You should look into them. They're great. However, why do we even need the half dozen that we're going to be using?



So to answer that question, let's take a look at how we might begin to mark up a mockup page that we're going to be converting to the HTML and CSS. And we would do this using the traditional pair of the html tag plus either a CSS class or a CSS ID. So as we begin to assign names to these styles we quickly realize that they are somewhat arbitrary. What I call "header" you might call "masthead". There is no true standardization. Using divs and CSS IDs and Classes in order to give meaning to a document is actually flawed, but it's the only tool we've had for many years.



So let's start again, and let's talk about how we would do this using HTML5. Now, we're not going to do the whole page, although you'll be doing that later. So we have nav,section and header. These are three of the HTML5 tags that we could use to define the structure of a page. HTML5 elements make it easier, ultimately, to style content for different devices and screens. So this is one of the major benefits when we're talking about responsive web design in HTML5. But there's other benefits as well.



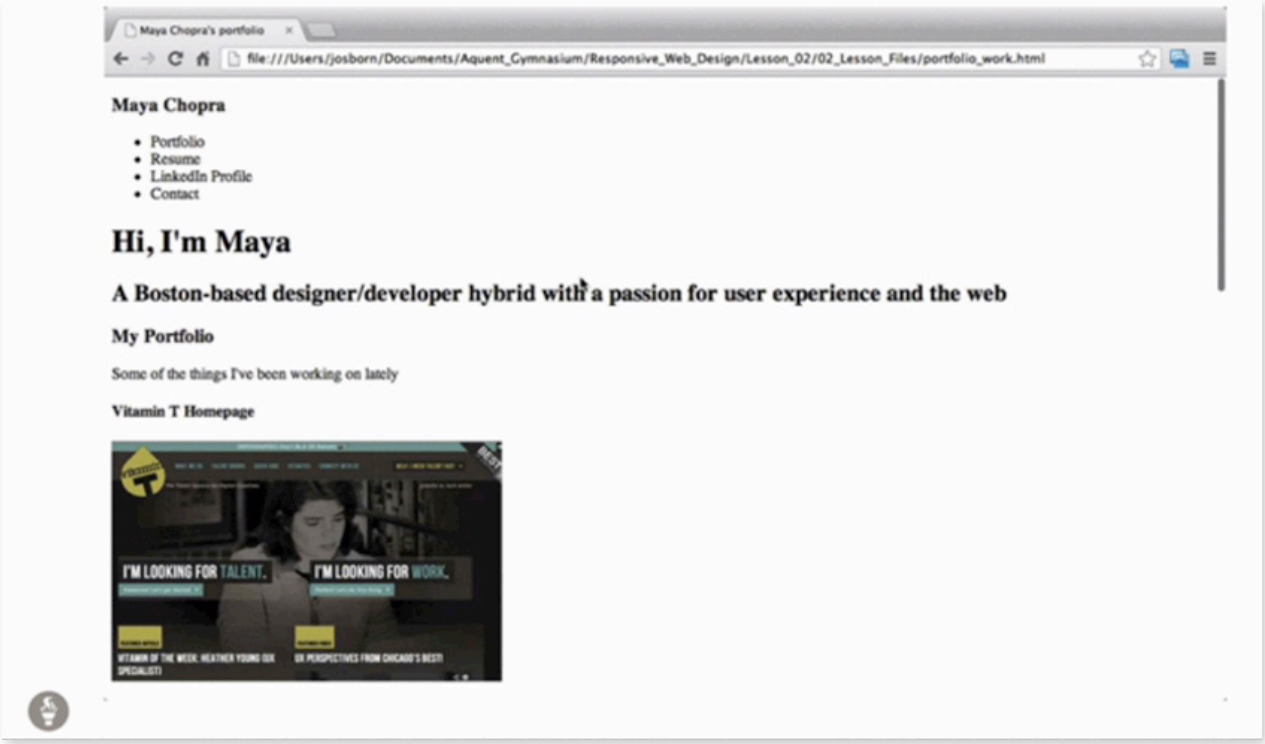
HTML5 elements have better semantic value. When the architects of HTML5 decided what names to assign to these new elements, they didn't just pluck them out of thin air. They pull professional designers and developers and ask them what class and id names they most often used in their layouts. They then took the most popular names and turned them into the new tags-- <header>,<footer> <nav> and so forth. That's essentially what semantic HTML is--using the best tag for the job.

Now another benefit is more organized code. If I'm a designer or developer, and I open up another designer's or developer's page, and they've used HTML5, it's much easier to read and understand. Another benefit is accessibility. The use of HTML5 syntax creates a meaningful document outline. Among other things, a document outline improves the accessibility of your page, especially with devices such as screen readers.

Lastly, I'll throw this out there, the use of more specific HTML5 elements can also improve search engine optimization. Search engines like being able to categorize the content on your page in order to give it weight or preference, and the specificity of HTML tags help make that happen. But enough theory, why don't I just go ahead and show you?

So we're going to be using this model as a mockup, and we're also going to be using this throughout the entire course. So this is time to fire up your text editor, and you're also going to want to download the lesson files from the classroom if you haven't already because you'll be using those now and moving forward. Again, you want to follow along. Be sure to use the pause button, and let's get started.

So the first thing we're going to do is open this document, portfolio_start.HTML, and we're going to save it as portfolio_work.HTML. And the reason we do this is to create a backup of the original file in case something goes wrong with this one. So let's take a look at this document in our browser. It's got a very basic unstyled property to it. So we have paragraphs, headings, some images, and now what I'd like to show you is how we're going to divide this using section tags in HTML5. So this first highlight here is the first section. The second highlight starts here, where it says My Portfolio and goes all the way down here to where it says My Resume. And then, the third and final section starts at My Resume and goes down to almost the bottom of the page.



OK. So the other thing I want to show you here is we're going to remove the comments out of the stylesheet, here on line six. Let's go ahead and save that. And this actually adds a standard reset to this page, where we're resetting the margins and columns. We'll talk about that a little bit later. Now it's one thing to see the page the way it currently looks, but you're about to add some tags to the lesson file. So I want to make sure you understand how those tags will relate to the final layout.

So in this mockup here of the three column layout, I'm going to show you the elements you'll be adding in order. The first step will be to add three section elements to the page. Then, within the first section, you'll add a unique <nav> element, as well as a unique <header> element. Next, you're going to group these two sections into an <article> element. Then, you'll return to the second section, give it its own unique <header> , and because there are some images in there, you'll add a <figure> element. And then last, you'll add two <footer> elements, one for the third section, and another for the entire page.

Now, I want to stress that you're not going to be creating this particular layout in this lesson. That's later on in lesson 6. In this exercise, you're just adding the skeleton of the site using HTML5. If you haven't had a whole lot of exposure to HTML5 syntax, some of these tags are probably unfamiliar to you, and I'll explain what each one does as it's introduced. And as I mentioned before, use the pause button if you have to. Take your time. This is the foundation of your responsive layout so it's important to get it right.

OK. So let's add some code. First step, make sure you're back in your text editor. Locate the tag and right below that on line 12, you'll add an opening tag. I've got some comments here that we can use, and these comments will help you find the area of the code where you can put the opening and closing tags. So again, we added the open tag. Let's go ahead and put the closing tag here. And now, let's add the second section, so we'll put that opening <section> tag, right above My Portfolio. Scroll down, and let's go ahead and add the closing <section> tag. And then finally, we'll add the third section right here above My Resume. And of course, we'll go down, and we'll add the closing <section> tag.

```
    <meta charset="UTF-8">
    <title>Maya Chopra's portfolio</title>
    <link rel="stylesheet" href="./css/style.css"/>

</head>

<body lang="en">
    <!--Add opening <section> tag below here-->
        <section>
        <h3>Maya Chopra</h3>
        <!--Add opening <nav> tag below here-->

            <ul>
                <li>Portfolio</li>
                <li>Resume</li>
                <li>LinkedIn Profile</li>
                <li>Contact</li>
            </ul>
```

Now that we've added these three sections, let's go back to the first one and begin to divide it up further. So at this point, we're going to add the <nav> tag to define an unordered list here. That list will end up being our navigation. We'll go down here, and we'll add a <header> tag. So a <header> tag is a new tag that allows us to define headings, in this case, heading one and heading two. But we can also put other content in there as well.

```
6       <link rel="stylesheet" href="./css/style.css"/>
7
8   </head>
9
10  <body lang="en">
11      <!--Add opening <section> tag below here-->
12          <section>
13          <h3>Maya Chopra</h3>
14          <!--Add opening <nav> tag below here-->
15              <nav>
16              <ul>
17                  <li>Portfolio</li>
18                  <li>Resume</li>
19                  <li>LinkedIn Profile</li>
20                  <li>Contact</li>
21              </ul>
22          </nav>
23          <!--Add closing <nav> tag above here-->
24          </section>
25      <!--Add closing <section> tag above here-->
```

Now, the <article> tag is something interesting. An <article> element is used to group content that, in theory, could stand alone. We'll talk a little bit more about versus later, but first let's just add the code. So there is our opening <article> tag. We're going to scroll all the way down, and we're going to put it right after the closing <section> tag here at the bottom.

Now, let's return, and we're going to go ahead and add a header to the first section, right around My Portfolio. And I'm just going to go ahead and indent this code here, just to tidy it up a little bit. And that looks better. So let's scroll down and find this image code for the Vitamin T home page. If we look at our mockup real quick, this is what you've done so far. Now, you're going to be adding a single <figure> tag for the first image only. I'm going to ask you to add the other two in a later assignment. So now, go ahead and add the open and closing <figure> tags around this image, and let me explain the role of this new element.

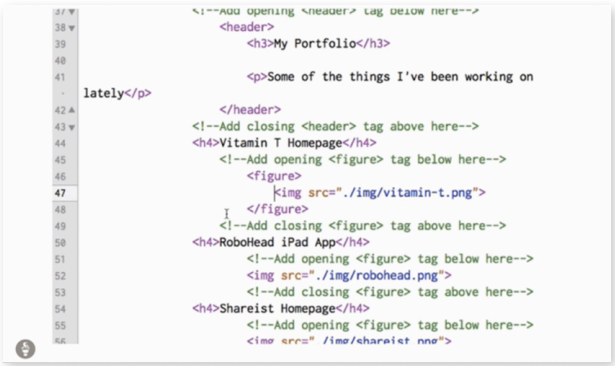
```
                <header>
                    <h3>My Portfolio</h3>

                    <p>Some of the things I've been working on
lately</p>
                </header>
            <h4>Vitamin T Homepage</h4>
                <!--Add opening <figure> tag below here-->

                <img src="./img/vitamin-t.png">

                <!--Add closing <figure> tag above here-->
                <!--Add opening <figure> tag below here-->
                <img src="./img/robohead.png">
                <!--Add closing <figure> tag above here-->
            <h4>Shareist Homepage</h4>
                <!--Add opening <figure> tag below here-->
```

The <figure> tag was invented as a way to identify objects on the page, such as illustrations, diagrams, photos, and so forth. In this case, it's being used to describe the image thumbnail that represents the designer's work. This dedicated element is not only semantically correct, but we can target it later on with CSS to add cool effects such as borders, drop shadows, and there's even an optional <figcaption> element that you can use to label images. But in our case, we're not going to be doing that. So let's go ahead and move on.



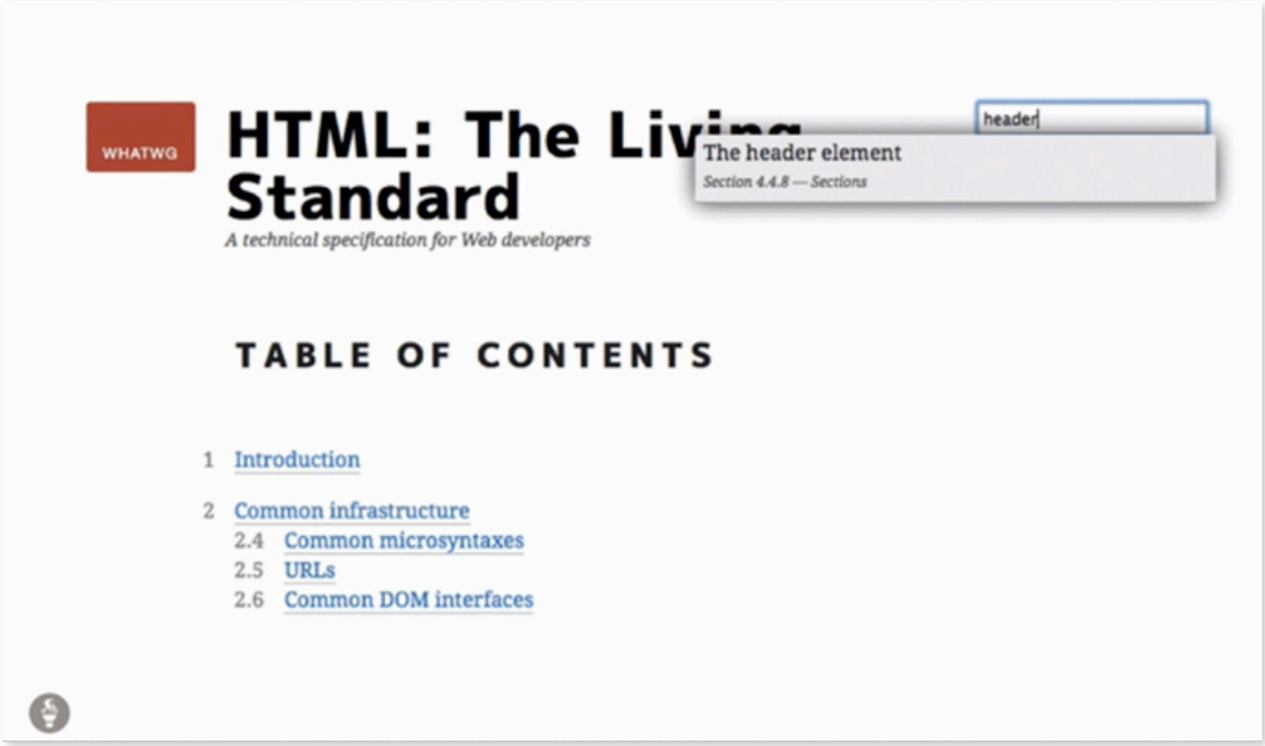
Now, let's go ahead and add two footers. There's one <footer>here, and we're going to add another <footer> right below it. Now, you might notice that we've added multiple footers as well as multiple headers. Are you allowed to do that? Absolutely. Let's go ahead and save this document. Let's take a look at it in the page. Reload it. Absolutely no visual change whatsoever! So again, the point of this is we're structuring this document so later on we can chunk it up for different devices and different screens. But right now, all the changes we've made are purely structural.

OK. So just as a recap as to what we did, I'm going to quickly walk through the order of tags that we added. So we added one section, two sections, three sections. Then, we added the <nav> . We added a <header> . We added an <article>, which was the parent of the sections. We added a <header> for the second , a <figure> element, a <footer>, and then, at last another <footer> . Now, we didn't add all the tags that we could have here, and that's quite on purpose. So you're going to be returning back to this code as an assignment, and you're going to be filling in the rest of the gaps.

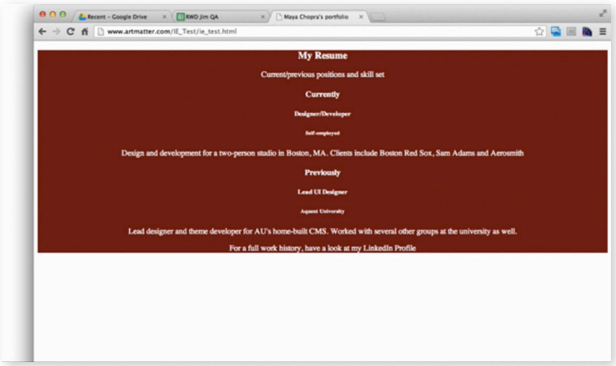
OK. Now, a quick overview of the difference between articles and sections-- when you hear <article>, think of an article of clothing like a shirt. So a shirt is a unique item. It has many parts like a collar, buttons, sleeves, but when put together, we recognize it as a shirt, no matter what. So the <article> element is like that. Any content within an <article> element should make sense on its own.

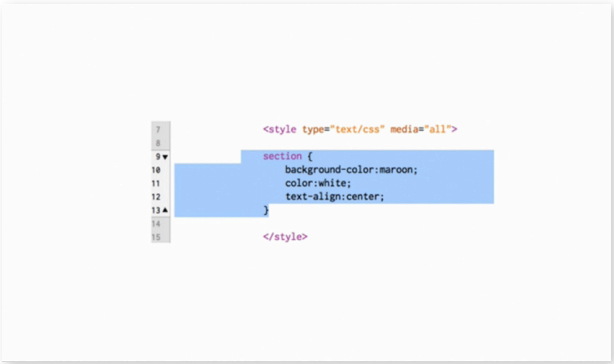
So a good example would be a blog post. A blog post typically has a beginning, middle, and end, and if you extract it from the blog and send it as an email, it would still make sense. Now, if we continue with the shirt analogy, a <section >would be like a sleeve. Remove a sleeve from a shirt and it no longer makes a whole lot of sense. In an HTML document we might use sections to define the different parts of a story or simply related parts of a page.

Now, if you want to learn more about the specifics of HTML5 syntax, there's no better resource than this one, developers.whatwg.org. So this is the official specification for HTML5. You can go in here, and you can type in a <header> element, for example, and you'll get some results that tell you exactly what it defines. And you'll even see some sample code here. So this is definitely your best resource to get the official scoop on HTML5 syntax.

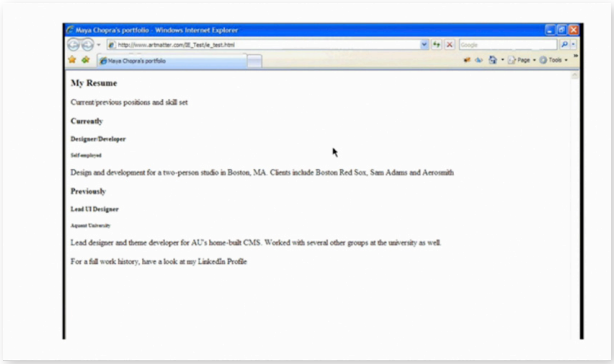


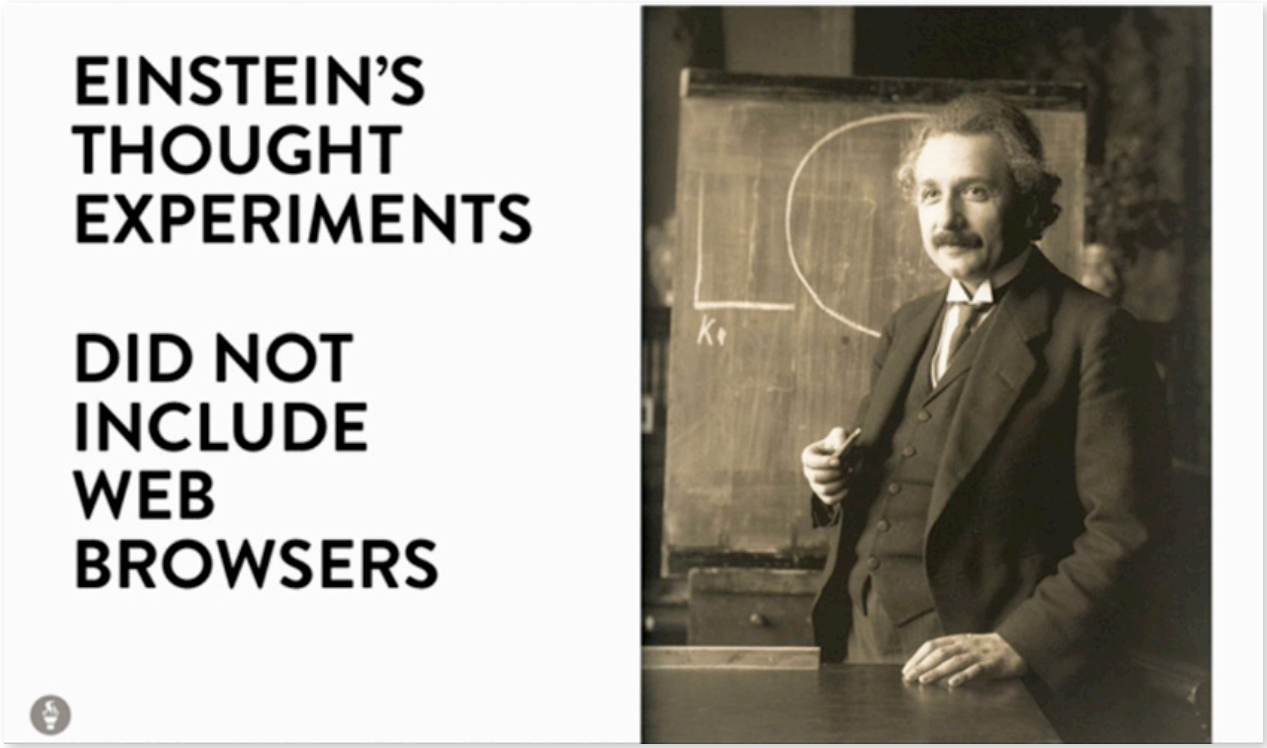
So I have some good news and bad news about these new HTML5 tags. I'll give you the bad news first. Most older browsers do not understand what to do with these new tags. So if you use them, you're most likely going to run into layout problems. Here's a quick example. I added a section tag to this page and then styled it with some simple rules-- background color maroon, text color white, and centered. But we're looking at this in a recent version of Google Chrome-- could be virtually any browser released in the last two years or so. If you're curious, the code looks like this.





And what I'm now going to do is open this exact same page in an older browser, in this case IE 7 or IE8, and what happens is we lose all of our styling, which is bad news. But the good news is that this is easily fixed-- not that big a deal. You can use HTML5 tags today with no problems. There's just a few things we have to do. I'm going to show you the answer in a moment, but there's a few details we need to look at first. And for that we'll turn to noted web developer; Albert Einstein.



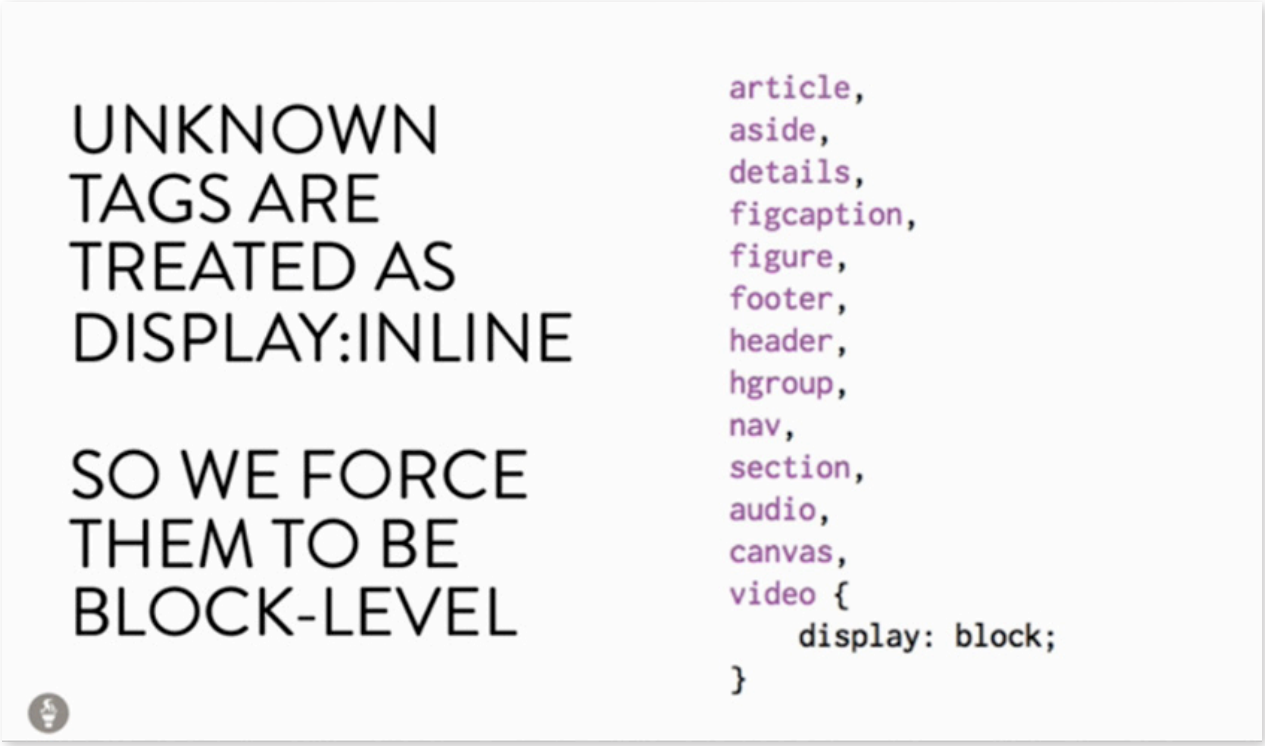


So we just added these HTML5 elements to our document, and they work fine in current browsers, but let's pause for a moment and do a quick thought experiment. Albert Einstein used to do these all the time. What happens if you're on a train, traveling at the speed of light, and you turn on a flashlight? Well, the purpose of a thought experiment is to think through the consequences of an action even though it might not be possible to test because you can still learn something of value.

We're going to do a thought experiment of our own, and you can actually test this one. So the question is how does a web browser style an element that it doesn't recognize? Let me make that a little more specific. Let's say we go in and we add a <cheeseburger>tag to this content. Well, most browsers will actually allow you to style that unknown element, even though we just made it up.

In fact, you can do this. You can test this on your own, and you could go ahead and add a style rule for `cheeseburger` and give it a color value, and it will work. So it's pretty cool. However, unknown tags in most browsers are treated as `display: inline`. So we have to force them to be block-level if we want them to work correctly. And this is what that code looks like. We have

a number of HTML5 tags, and we simply say, `display: block`. This allows us to use these in our code with no problems. However, there's a problem.



Internet Explorer and particularly versions previous to 9 do not allow unknown elements to be styled, at least without JavaScript. So what we have to do is add a shim to force older versions of IE to obey, which is what we want it to do. So let's take a look at those two fixes that we're going to use in our document. First of all, let's take a look at the external stylesheet that's been linked. If you open up that style.css document, you'll see the tags I just added, I mentioned earlier. So these are all the fixes for modern browsers, telling them to be `display: block`. We're actually getting this from normalize.css, something that I'll be talking about in just a moment.

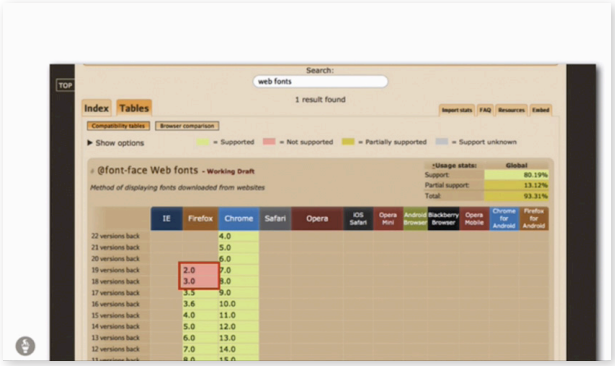
Now, we also want to take care of earlier versions of IE. So open up this document, HTML5_shim.txt. And what I'd like you to do is to copy all the code inside it, paste, and put it right here underneath the link tag. This is a conditional comment. And this means it only targets Internet Explorer and no other browsers. Specifically, the condition is "if you are a version lower than IE 9, then use this JavaScript code located here at this address-- html5shim.googlecode.com. "

So the job of this JavaScript is to force these versions of IE to render HTML5 elements, and then we can style them using traditional means. So here are the two resources I promised-- normalize.css is an updated version of a CSS reset stylesheet. Not only does it allow browsers to render all elements more consistently, but it includes those styles for HTML5 elements as we just saw. And, of course, here's the link for HTML5shim. You can find the specific URL in the classroom resources. This is Responsive Web Design.

CHAPTER 3: WHAT YOU NEED TO KNOW ABOUT CSS3

OK. In this section we are going to talk about what you need to know about CSS3 or at least for what we're doing here in responsive web design. The story of CSS3 is very similar to HTML5. In other words, we have a lot of great new features that we can use, but we have to be very aware of browser support.

So I'm going to give you one of my favorite references here. This is a website called caniuse.com. And I'm just going to take you through a quick tour of how this works. You can go ahead and find any CSS3 or HTML5 property here, and do a little search, for example, web fonts. If you do a search for web fonts, you'll see here the font-face Web fonts section come up. And what we're really interested in is a few things.



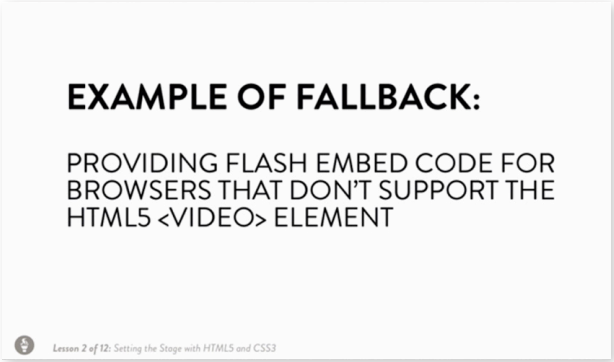
First of all, we want to see current support for font-face throughout all modern browsers. So we're seeing here that because they're green-- IE10, Firefox 19, Chrome 26, Safari 6, and so forth-- all of these support font-face. We can also see past versions of the support here. We're very interested in the total support. So here, 80% support for font-face is a pretty good number, and then the other 13% is actually partial. So that's also good.

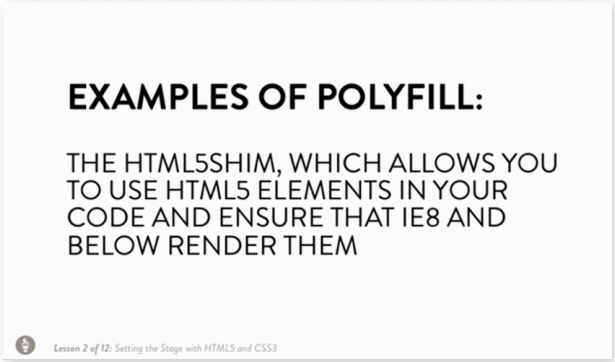
There's a couple little hidden tricks here that I want to show you. You do want to make sure that you're looking at all versions of browsers. The problem with browsers these days is they're being released so quickly that this graph is a little deceptive. So Chrome 24, for example, might only be two months old, who knows. So you want to go ahead and Show All Versions, and once you do that, you'll see a true glimpse of support. So for example, here we can see that Firefox 2.0 and 3.0 did not support web fonts. So if that's inside your profile or in your target audience, then you want to be sure that you come up with a solution for that or at least be aware of it.

Here's another source that I like, html5please.com. So this is a slightly different tack than Can I Use. So the way that this works is if you're looking for font-face support, you get a very simple graph. Go ahead and use it with a fallback. We have other options here. For example, if we wanted to use FileSystem in the FileWriter API, it would tell us to avoid it. Then, we've also got "Caution", as well. Now, if you click on any of the plus signs to the left, you'll get a little more detail about some of the fallbacks, for example.

Let's just talk about what that means. So a fallback is code or content that's to be used when the original resource can't be used, usually because it's an unsupported format. There's also another term that you'll find out there called polyfill. So polyfill is code that provides the feature that you expect the browser to provide natively. If that sounds a little confusing, let me give you an example. So an example of a fallback is providing Flash embed code for browsers that do not support HTML5 video. So that's a fallback. Browser doesn't support the video tag-- go ahead and fall back to traditional Flash code.

A polyfill is something like the one we just used, HTML5SHIM. A polyfill allows you to use HTML5 elements in your code and makes sure that IE 8 and below renders them. So you're actually providing support for missing features when you have a polyfill. OK. let's go ahead and choose one CSS3 feature that we're going to be using today. The winner of this is-- you probably have guessed this already-- that's right, CSS3 web fonts, otherwise known as @font-face.

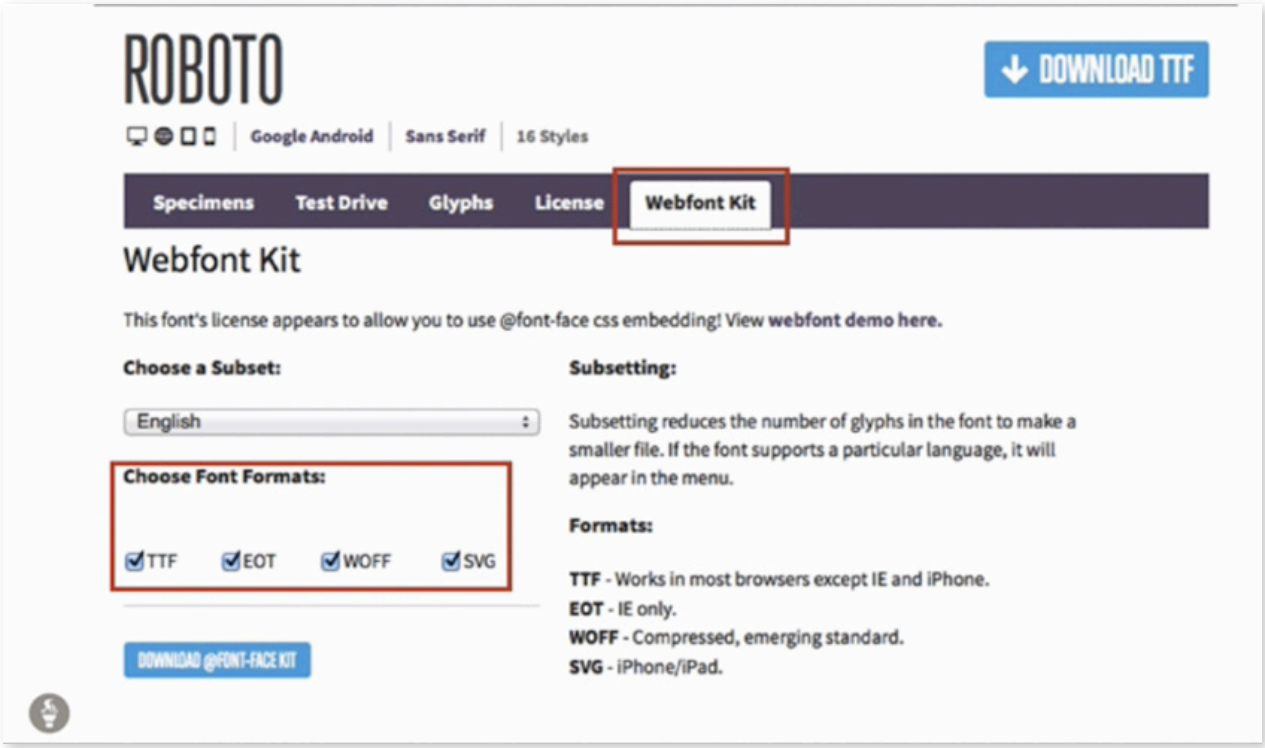


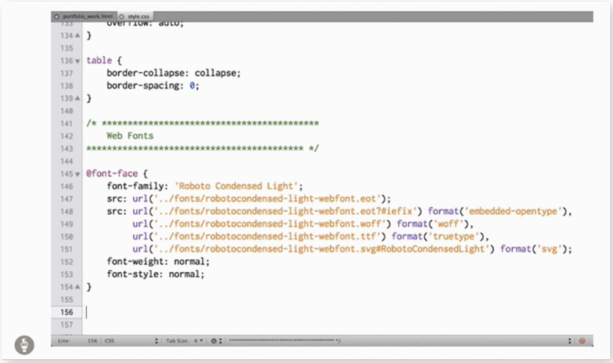


So here's a quick tour of how to embed web fonts. There's three steps. You have access to licensed web fonts. These can be local or remote. You add a @font-face reference to your stylesheet and you point to those font files. And then third, you add the CSS rules for the text you'd like to style. Now, the source of the fonts that I'm using here come from fontsquirrel.com. This is not the only source for web fonts, but this is a great source because these are 100% free for commercial use. Another good source is Google Web Fonts.

One thing we do want to pay attention to, though, is how Font Squirrel treats the different font formats. So you see here that there is such a thing as a Webfont Kit. What's a Webfont Kit? Well, it actually allows you to select up to four different font formats.

So this is the Achille's heel of web fonts. Web fonts are great. They allow us to embed fonts anywhere on our page and make it look exactly the way we want, even though the user doesn't have that font installed. The problem is we can't just use one font format. We have to use up to four if we want to make sure that these fonts are as widely compatible as possible. So it's a little bit of a hassle, as you'll see in a second, but in the end, it works, and a service like Font Squirrel makes this very easy.





OK. So let's see how this works. We're going to go ahead, and we're going to look at our portfolio work page. And I'm going to scroll down through the external stylesheet, and I'm simply going to remove the comments that I put in here. So this is a bit of a cheat. This is just to make sure that it works and to save you from all that typing. You'll see here that there are actually URL references to numerous font files.

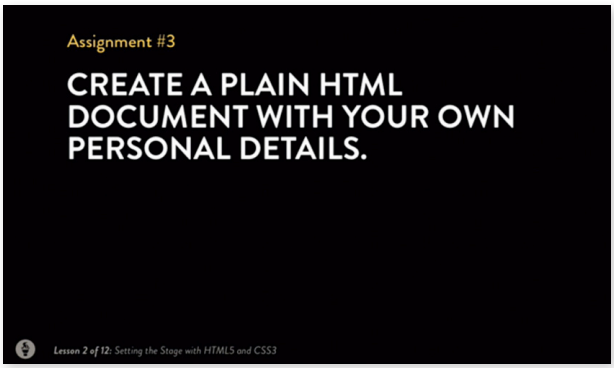
Now again, these are in your lesson folder, so this is why this is going to work. Now, we need to add the style, and I'm going to make a very simple style here for the body. And we'll go ahead and say font-family and then Roboto Condensed Light, that's the first choice. And we're also creating a font stack. So these are backups. Let's go ahead and reload our page, and there we are.

Now, there's obviously much more to web fonts than just this, but again, the goal here was to give you just the CSS3 that you needed to know to move on.

Any questions? Please hit the Forum. Here is some homework for you. First of all, you have a short quiz. That's assignment number one, so we always have a quiz at the end of each lesson.

Assignment number two is to finish adding some of the HTML5 tags in that portfolio_work.html document. So those astute observers in the audience may have noticed that I did not add all the HTML tags that I could have. So, for example, I only added one figure element, even though there are two other images that could have that figure element. So what I'd like you to do is to go back to that file and add those tags. It's fairly straightforward, particularly because I added some very useful comments. However, if you want to check your work, you can examine the portfolio_final.html document, and that's in the completed folder in your lesson folder.

Now, assignment number three is a little bit more challenging. What I'd like you to do-- because, of course, the objective of this entire course is to give you a responsive portfolio site or a response portfolio page -- is to create a plain HTML document, very similar to the one that we just worked on, and then put in your own personal information. Now, because this is the first assignment in this course that asks you to work on code, I wanted to take a few moments to mention how the Forum works and your role in it.



So we have a handful of teaching assistants who help out in the forums, and they check student work. However, there's only a few of them, and there's literally thousands of you, so they can't possibly check everyone's work. And Gymnasium isn't designed for that in any case. So one important request we have is that you go into the Forum, find someone else's assignment, and then comment on it. If it's code, take the time to evaluate it, and try to help if possible, or if it's critique or a written assignment, take the time to craft a thoughtful reply. So the end result is that everybody wins. Sound good? OK.

Now, for extra credit go ahead and visit fontsquirrel.com where, for free, you can download a new Webfont Kit of your own choice. Go ahead and try practicing embedding this new font onto your page, so you can get used to working with web fonts. And that's it. I'll see you in the Forum. As I mentioned before, we have TAs waiting there to answer your questions and of course your other class members. See you next session. This is Jeremy Osborn, signing off.