AQUENT
GYMNASIUM

RESPONSIVE WEB DESIGN:
BUILD A PORTFOLIO FOR ALL DEVICES

LESSON 8

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.

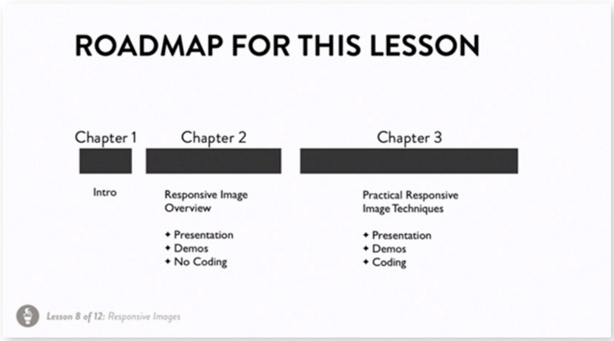
CHAPTER 1: RESPONSIVE IMAGES OVERVIEW

Welcome to Responsive Web Design, a Gymnasium course brought to up to you by Aquent and Vitamin T. Responsive Web Design-- or Promote Yourself Responsively: Build a Portfolio for All Devices. This is lesson 8, Responsive Images.

GYMNASIUM

8. RESPONSIVE IMAGES

As always, there'll be an assignment and a brief quiz at the end of this lesson. Be sure use the pause button at any given point in order to look at the code or just stop and reflect. If you have any questions, be sure to hit the Forum. There, there will be instructors, TAs, and your other classmates who will be there to assist.



ROADMAP FOR THIS LESSON

Chapter 1	Chapter 2	Chapter 3
Intro	Responsive Image Overview • Presentation • Demos • No Coding	Practical Responsive Image Techniques • Presentation • Demos • Coding

Lesson 8 of 12: Responsive Images

Let's talk a little bit about the roadmap for this lesson. We're currently in chapter 1, the intro. Chapter 2 will be a responsive image overview. In that overview, there will be a presentation, some demos but very little coding. Chapter 3-- practical responsive image techniques. There, there'll be some presentations and demos and a significant amount of coding.

This is Responsive Web Design.

CHAPTER 2: AN INTRODUCTION TO RESPONSIVE IMAGES

This section is a responsive image overview. Our objective here is to present some strategies for the challenges that responsive images will introduce to your projects. And they will introduce some challenges. Responsive images are a particularly thorny issue in the world of responsive web design, and if you're not careful, you might get hurt.

To make this a little more clear, let's take a look at this quote by Tim Kadlec. "The most common offender for poor responsive performance is downloading unnecessarily large images, or worse yet, multiple sizes of the same image." So what's that all about? I thought this was about responsive images, not performance.

Tim Kadlec

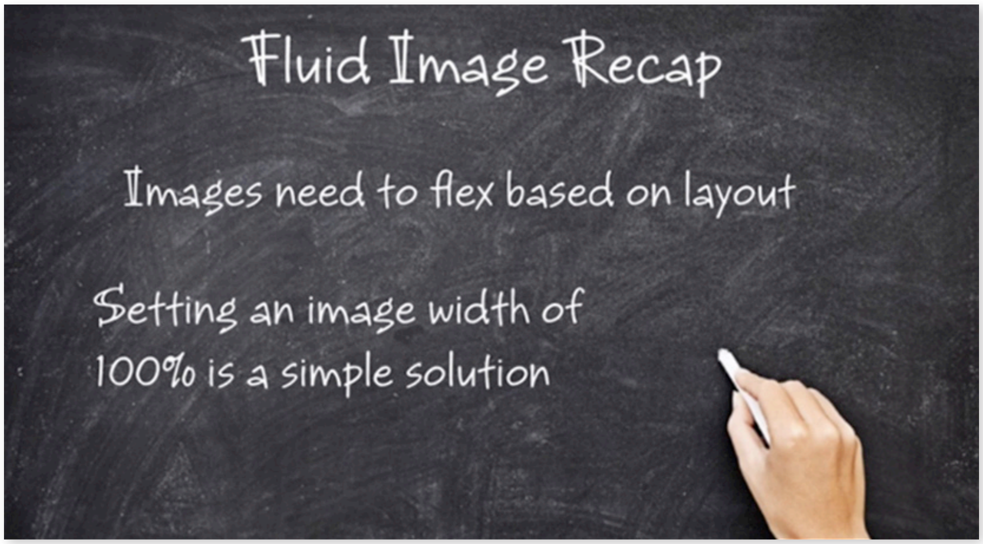
"THE MOST COMMON OFFENDER FOR POOR RESPONSIVE PERFORMANCE IS DOWNLOADING UNNECESSARILY LARGE IMAGES, OR WORSE YET, MULTIPLE SIZES OF THE SAME IMAGE."

HTTP://24WAYS.ORG/2012/RESPONSIVE-RESPONSIVE-DESIGN/

Lesson 8 of 12: Responsive Images

Well, in fact, this is part of the challenge. Your images need to communicate effectively in all scenarios but not at the cost of poor performance. Let's just do a little recap to put this into context. We know that images have to flex based on layout. This is what fluid images are, and this is one of the characteristics of responsive design.

We've also seen in previous lessons how you can set an image width of 100% as a simple solution. Now, more specifically, using a property of max-width of 100% is what we do. The max-width property sets the maximum width of an element, and with a value of 100% , as long as that image is not wider than its container, it scales great in most browsers.



Fluid Image Recap

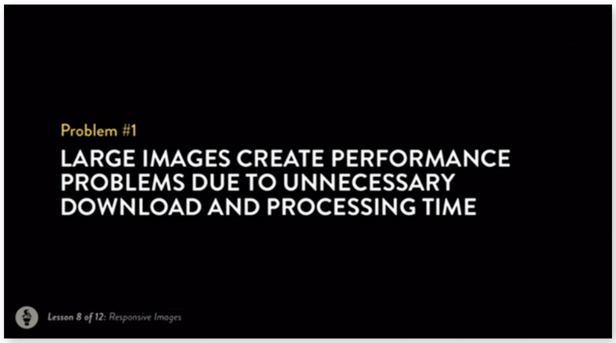
Images need to flex based on layout

Setting an image width of 100% is a simple solution

Just to see how this works, let's just take a look at a very simple example. Here we have a large image, and it's currently seen in a large screen. I'm going to go ahead and resize this to about 320 pixels, or smartphone view.

And we can see that the fluid image works exactly the way we want it to. So it starts off at a large size, and it scales down to a small size. Perfect, right?

So the problem here is that large images can create some problems when it comes to performance. What happens is, if you're on a smartphone, there's no reason to download that large image. It's just going to be overkill, and it's going to slow down the performance and begin to annoy your users. So, to get a little more objective with this, let's just take a look at some of the different scenarios here.

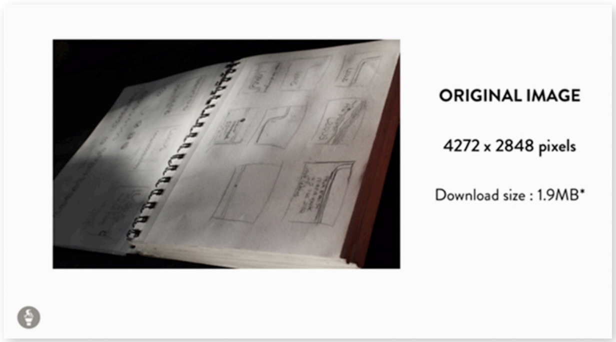


Problem #1

LARGE IMAGES CREATE PERFORMANCE PROBLEMS DUE TO UNNECESSARY DOWNLOAD AND PROCESSING TIME

Lesson 8 of 12: Responsive Images

So this is the original image as we might get it from a digital camera. I should make some qualifications here that this is 1.9 megabytes, but this is a JPEG at 80% quality. Of course, you've got different compression and different file formats, but for now, let's just assume this is the baseline. So our original image has 4272 pixels-- way too big to be on virtually any screen out there. So we might resize it to this-- 1440 by 960.



ORIGINAL IMAGE

4272 x 2848 pixels

Download size : 1.9MB*

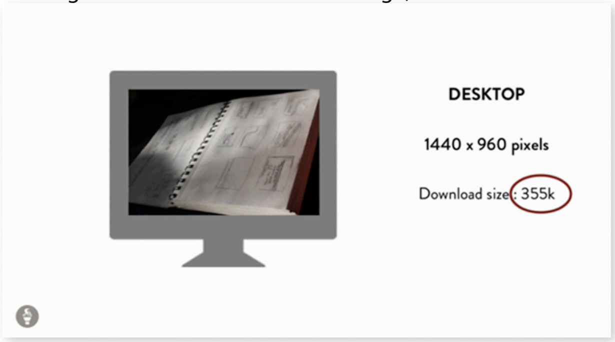
This is a relatively common desktop size, although we can get larger. Even here, a full screen image is going to be 355kb, so that's pretty big. We generally want to stay around 100kb, maybe 200kb, depending on your target audience.

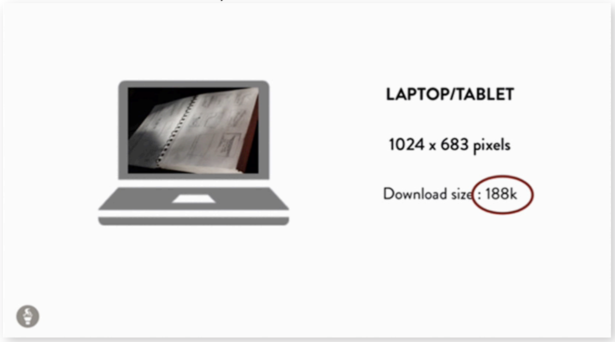
So let's keep on going down. A laptop or a tablet size-- so 1024 by 768. That download size is 188kb. Still pretty big.

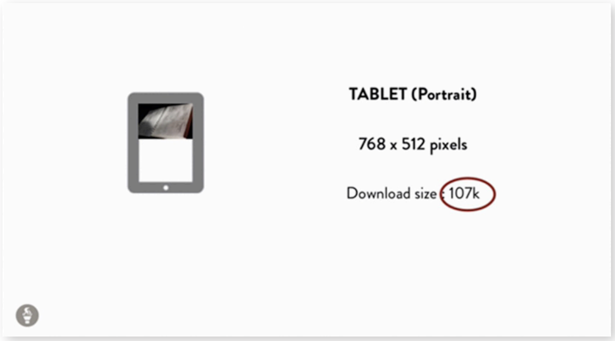
Tablet size in the portrait view-- 768 by 512. Download size 107kb-- we're getting better. And smartphone-- 320 by 213. Download size 20kb-- acceptable under almost any circumstance.

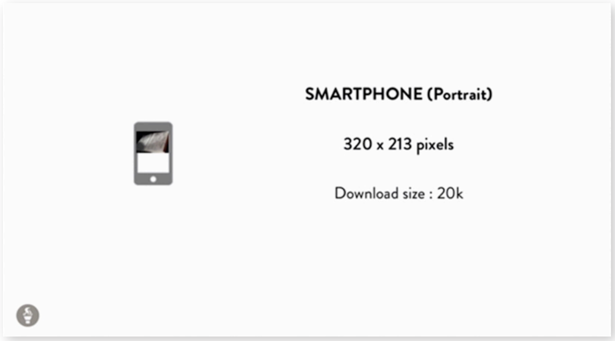
So we're going to come back to that issue of download time and performance, but let's just look at a related issue. Problem number two is different screen sizes require different editorial content, and what do I mean by editorial content? Again, let's take a look at this desktop, 1440 by 960.

We've got five runners here in our image, and this works well as a wide screen. However, it's not going to work so well on a smartphone. I've scaled it up a little bit just for this presentation, but at its actual size, everything is going to get lost here. If we were able to crop that image so that you have the winner of the race and maybe one other racer, this is going to be more effective.

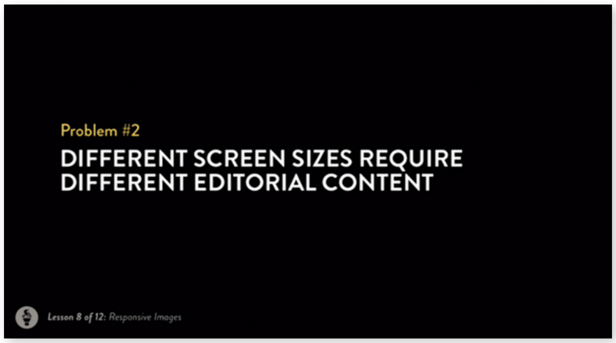








It still gets the point across, and we potentially have a savings of file size also.


Problem #2

DIFFERENT SCREEN SIZES REQUIRE DIFFERENT EDITORIAL CONTENT

Lesson 8 of 12: Responsive Images

So, one of the solutions is to use media queries and CSS background images. Let me talk about how this solution works. If we use a media query such as this, which is media screen only, max-width 320 pixels, this is targeting all screens smaller than the width 320. So we can use this image, race-small.png, and this will be loaded for the smartphone.

We could also create a more targeted media query-- so between 321 and 768, and we could send a larger image, race-medium.png. And here we have three racers in the image. And then, last but not least, everything above 769, for example, we could use a race-large.png with all five racers. So we can have an image that's suited for the screen in both file size as well as content.

High-five, right? We haven't seen you guys for a while. That must mean something bad is coming up, because what if I told you this only worked for CSS background images? You might not be so happy.

So, just as a brief refresher, let's go all the way back to CSS in HTML 101. And remember that HTML equals content. So, on our portfolio page, our thumbnail images here, this is our content. CSS equals decoration-- decoration or style, and that's really what the background image is.

Partial solution

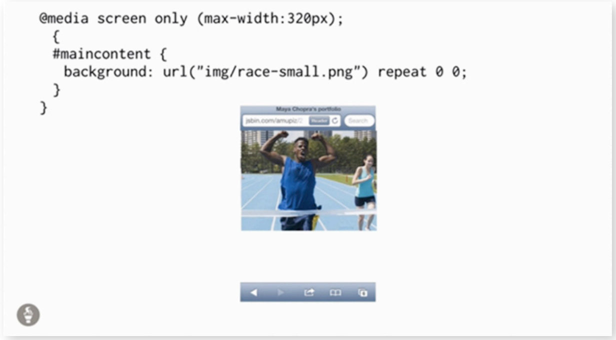
USE MEDIA QUERIES AND CSS BACKGROUND IMAGES

Lesson 8 of 12: Responsive Images

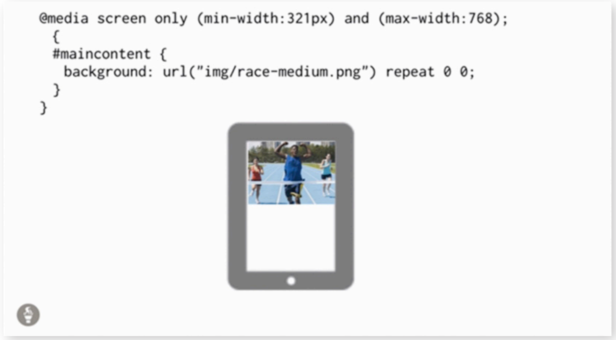
If we were to remove the background image, would we lose significant content on this page? No, it just wouldn't look as great. However, if we were to lose those thumbnail images, that would be a serious blow to the image for losing content.

So, the issue is we've established that we have to use in-line or HTML images for important content, and the image element is only allowed a single source. We cannot send multiple sources of images using HTML.

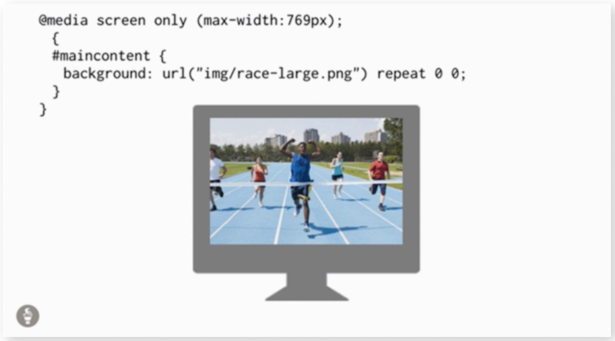
Now, I should point out that a lot of work has been done by other folks. This is perhaps some of the most thorough and significant work on responsive images at this point by Jason Grigsby. That's the URL down below, that blog.cloudfour.c om/responsive-imgs/. So I'm going to summarize some work that Jason has done as well as other folks out there.



```
@media screen only (max-width:320px);
 {
 #maincontent {
  background: url("img/race-small.png") repeat 0 0;
 }
}
```

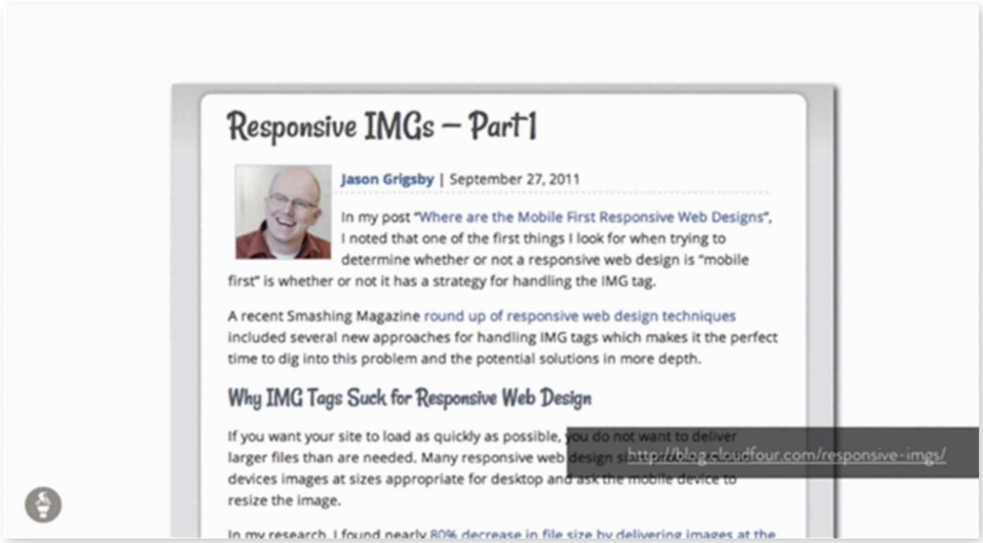


```
@media screen only (min-width:321px) and (max-width:768);
 {
 #maincontent {
  background: url("img/race-medium.png") repeat 0 0;
 }
}
```



```
@media screen only (max-width:769px);
 {
 #maincontent {
  background: url("img/race-large.png") repeat 0 0;
 }
}
```

The end result that Jason came to, as well as a lot of other folks, is we need a new way to deal with images. The good news is people have figured this out. So Responsive Images Community Group at responsiveimages.org, this is a group of developers and designers and a bunch of other folks who are trying to figure out the path for expanding images or making images work better, especially responsively.



Responsive IMGs – Part 1

Jason Grigsby | September 27, 2011

In my post "Where are the Mobile First Responsive Web Designs", I noted that one of the first things I look for when trying to determine whether or not a responsive web design is "mobile first" is whether or not it has a strategy for handling the IMG tag.

A recent Smashing Magazine round up of responsive web design techniques included several new approaches for handling IMG tags which makes it the perfect time to dig into this problem and the potential solutions in more depth.

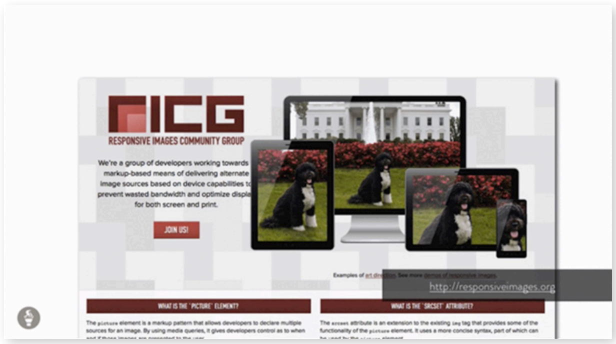
Why IMG Tags Suck for Responsive Web Design

If you want your site to load as quickly as possible, you do not want to deliver larger files than are needed. Many responsive web design s devices images at sizes appropriate for desktop and ask the mobile device to resize the image.

In my research, I found nearly 80% decrease in file size by delivering images at the

http://blog.cloudfour.com/responsive-imgs/

So let's talk about this for a little bit. There's a page here that I'm going to ask you to read later, usecases.responsiveimages.org, and this goes through a pretty thorough exploration of what we're trying to achieve with these responsive images.



RICG
RESPONSIVE IMAGES COMMUNITY GROUP

We're a group of developers working towards markup-based means of delivering alternate image sources based on device capabilities to prevent wasted bandwidth and optimize displa for both screen and print.

JOIN US!

Examples of art direction. See more demos of responsive images.

http://responsiveimages.org

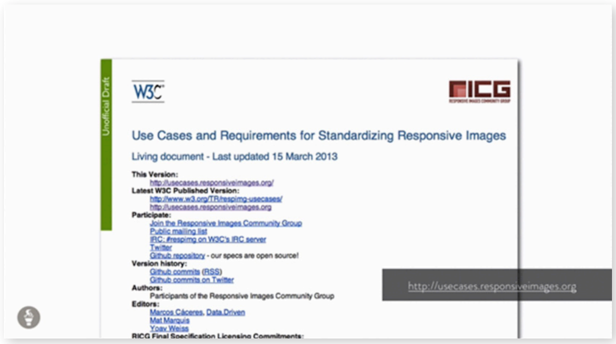
WHAT IS THE "PICTURE" ELEMENT?

The picture element is a markup pattern that allows developers to declare multiple sources for an image. By using media queries, it gives developers control as to when

WHAT IS THE "SRCSET" ATTRIBUTE?

The srcset attribute is an extension to the existing img tag that provides some of the functionality of the picture element. It uses a more concise syntax, part of which can

So there are currently two proposals on the table-- the picture element and the srcset attribute. Let's just take a quick look at both of these. Now, before we start discussing this, we don't want to fall too far down this rabbit hole. In fact, there's a lot of detail that we could begin getting into here. I'm going to avoid it and talk about why we're doing this.



Unofficial Draft

W3C

RICG

Use Cases and Requirements for Standardizing Responsive Images

Living document - Last updated 15 March 2013

This Version:
http://usecases.responsiveimages.org/
Latest W3C Published Version:
http://www.w3.org/TR/respimg-usecases/
http://usecases.responsiveimages.org
Participate:
Join the Responsive Images Community Group
Public mailing list
IRC: #respimg on W3C's IRC server
Twitter
Github repository - our specs are open source!
Version history:
Github commits (RSS)
Github commits on Twitter
Authors:
Participants of the Responsive Images Community Group
Editors:
Marcos Cáceres, Data.Driven
Mat Marquis
Yoav Weiss
RICG Final Specification Licensing Commitments:

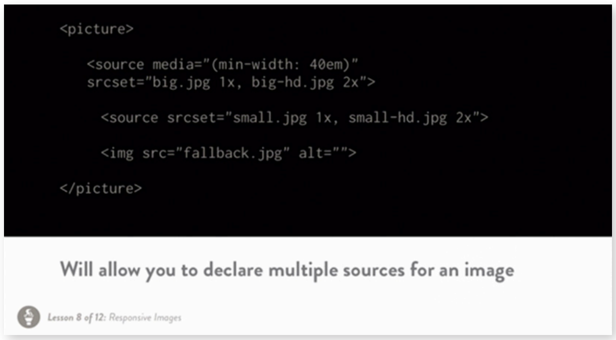
http://usecases.responsiveimages.org

To make a long story short, here, we have to think about the different ways that responsive images can be used. So we're creating a little tiny portfolio site here. It's about a page long. There's some complicated things going on in terms of the responsiveness and the typography, but this is like making a little house or maybe even smaller-- maybe a custom room.

However, if you're creating a large-scale site with multiple HTML pages, content management systems, images, and so forth, this is more like building a skyscraper, and so your tactics for responsive images are going to have to scale accordingly if you've got a large project. So, let's take a look at the picture element.

The picture element essentially allows you to declare multiple sources for an image. So we have the picture element, and we have a source which is referencing a media query of min-width of 40ems, and we've got a big JPEG and big HD JPEG that we would be targeting.

Now again, I'm not going to get too far into the details right now. Let's take a look at the second one. So this is the srcset attribute. Unlike the picture element, which is a brand new element, the srcset attribute piggybacks onto the image tag. So here, again, we have srcset equals small.jpg 640 wide, small-hd.jpg 640 wide 2 times, and med.jpg, so forth.



```
<picture>
   <source media="(min-width: 40em)"
   srcset="big.jpg 1x, big-hd.jpg 2x">
     <source srcset="small.jpg 1x, small-hd.jpg 2x">
     <img src="fallback.jpg" alt="">
</picture>
```

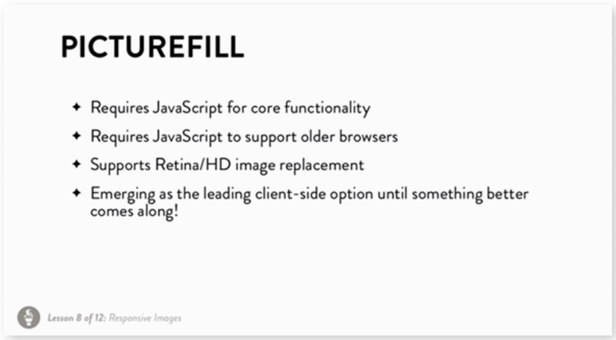
Will allow you to declare multiple sources for an image

Lesson 8 of 12: Responsive Images

But at this point, speaking of Alice in Wonderland, this is a little bit of a pipe dream. Both of these proposals aren't going to be real for quite some time. However, we're not just exploring this for the fun of it. There are some solutions out there that reference these.

One of the leading solutions of this nature is picturefill. So the way that picturefill works might look a little familiar. Instead of this picture element, we use a div tag. But we also get to use the HTML5 data source attributes.

So essentially the way this works is the data source points to a specific image, such as small.jpg or medium.jpg. Notice that the date-media attribute points to a media query. So essentially we can target different screens and send images to that size screen, which is exactly what we want.



PICTUREFILL

- Requires JavaScript for core functionality
- Requires JavaScript to support older browsers
- Supports Retina/HD image replacement
- Emerging as the leading client-side option until something better comes along!

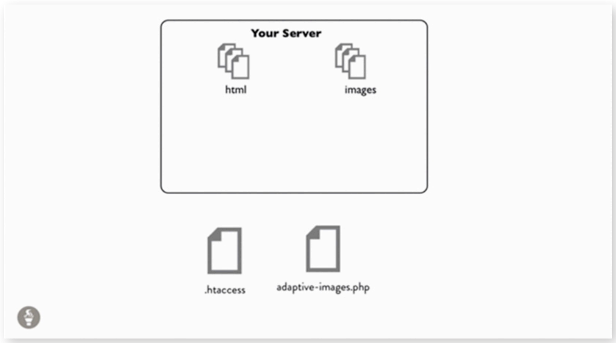
Lesson 8 of 12: Responsive Images

Now, of course, the magic here is not just in the HTML, but it's in the JavaScript that creates this. So there is JavaScript that's required for this. And not only that, but we need additional JavaScript to make sure that older browsers also support it. However, one of the benefits is we are allowed to use Retina or high definition images, and they will be replaced accordingly, and this seems to be emerging as one of the leading client-side options, at least until something better comes along.

So you'll be using picturefill in the coding exercise in chapter three. By the way, you can download all this code for free on GitHub. Scott Jehl is the one who created this, and this is the URL.

Let's take a look at another solution that's out there. And this is a server-side solution. And again, there are many others out there. This is just one of them, adaptive-images.com.

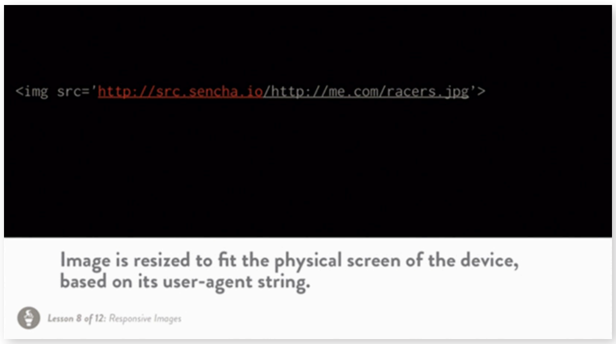
Here's the way adaptive images works. You've got your server here with some HTML and images. And what we need to do is take these files, htaccess and adaptive-images.php, and these need to be placed onto a server. Now once that happens, we need to do a little more work. We have to add some JavaScript into the head of the site, and we need to add the media query values that we want to target into the PHP file.



Now, this is simplifying it somewhat, but essentially the way this works is when there is a request for a specific image, it gets resized depending on the size of the screen. So you only need to provide one, single, high resolution image, and the adaptive images will scale everything as needed. If you want to learn more about how this works, you'll need to go to adaptive-images.com, there are more details on the site.

Now, some of the downside here has to do with the fact that it's server-side, and it may not fit in your solution. So you may not use PHP, or you may have some other server setup, but let's take a look at one other option. This is a third-party solution, and again, just like the previous two examples, there are multiple other ways for this to happen, but this seems to be one of the leading examples, sencha.io Src.

So again, to make a long story short, the way this works is we add a prefix to the URL for our images. So here, http://src.sencha.io. When you do this, your image gets resized to fit the physical size of the device, and this is based on the user agent string. So, if an iPhone goes to your site, it gets resized for that screen.



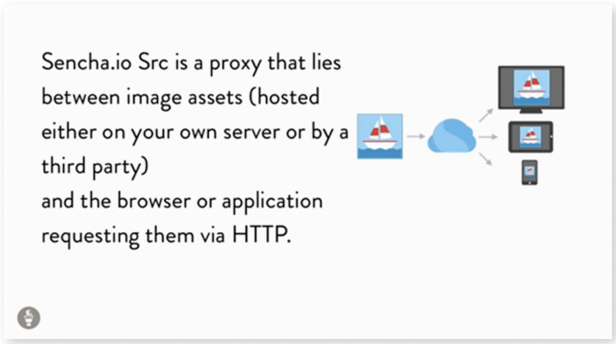
```
<img src='http://src.sencha.io/http://me.com/racers.jpg'>
```

Image is resized to fit the physical screen of the device, based on its user-agent string.

Lesson 8 of 12: Responsive Images

Essentially, the way it works is Sencha is a proxy, so it sits between your images and the browser or application that's requesting them, and it does all that resizing offline or on this third-party service.

Now of course, there are pros and cons here, just like the other two examples. If you need to learn a little bit more about this, you can go to sencha.com, and the URL is down here, and we'll put it in the classroom as well.



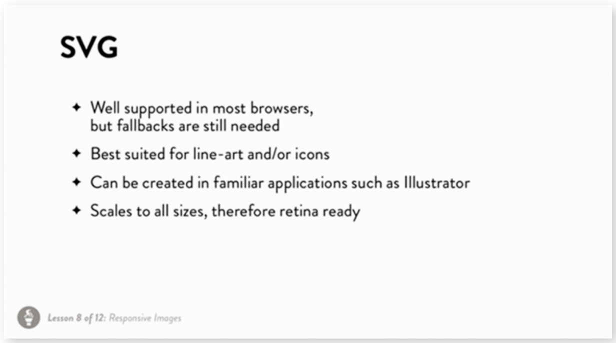
Sencha.io Src is a proxy that lies between image assets (hosted either on your own server or by a third party) and the browser or application requesting them via HTTP.

Now, with all this talk about pixels, it's easy to forget that there are some other solutions. So we're going to leave pixel world for a second and go into vector world, and specifically SVG, Scalable Vector Graphics. So SVG is pretty well-supported in most browsers. It's a format that's been around for some time. We still need fallbacks, however, because it's not completely supported.

Graphics created in SVG are typically best for line art or illustration or icons. You can actually make these in programs such as Illustrator. One of the most significant benefits is the fact that SVG is resolution independent. That means that we can use them, and they will scale natively to all sizes and look great in devices such as the Retina iPad or any HD device. This is great.

Let's just take a look at how you could begin to use this. This is a small project put out by the folks at ZURB, and what they've done here is convert some familiar social icons, such as Twitter and Facebook, to SVG.

So you can go ahead, and you can add these to your page, and they will be extremely small in terms of file size. Additionally, they will always scale as needed.



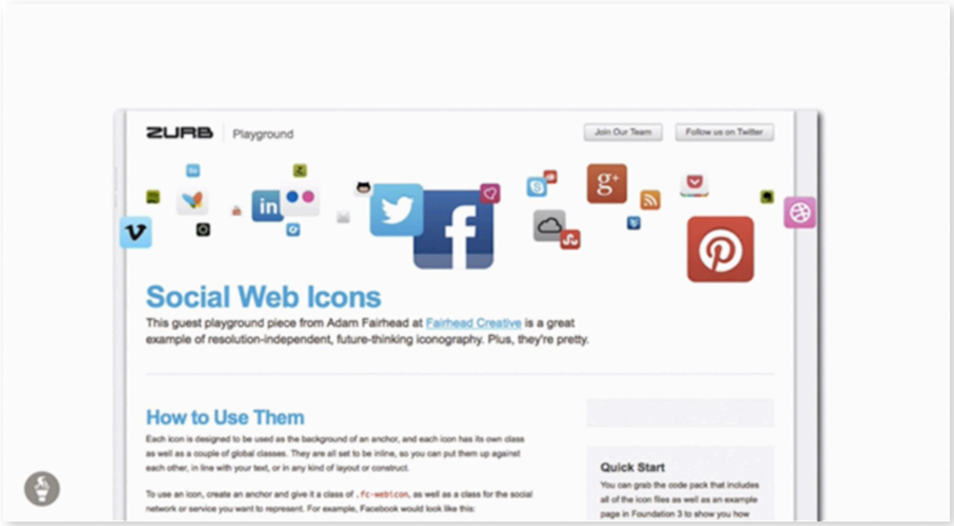
SVG

- Well supported in most browsers, but fallbacks are still needed
- Best suited for line-art and/or icons
- Can be created in familiar applications such as Illustrator
- Scales to all sizes, therefore retina ready

Lesson 8 of 12: Responsive Images

Let me just show you a quick example here. So, this is the Dribbble icon in SVG, and you can see that as I start off, it's fairly small, and then, I began to resize the browser, and it's going to maintain its crisp, clean, sharp edges like all good vector graphics do.

And that's exactly what we're seeing here. Total file size for this document-- 1 kb, 1. So it's going to work really well even at the largest size.



Social Web Icons

This guest playground piece from Adam Fairhead at Fairhead Creative is a great example of resolution-independent, future-thinking iconography. Plus, they're pretty.

How to Use Them

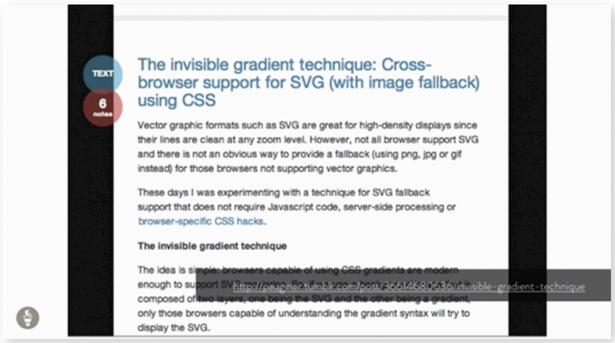
Each icon is designed to be used as the background of an anchor, and each icon has its own class as well as a couple of global classes. They are all set to be inline, so you can put them up against each other, in line with your text, or in any kind of layout or construct.

To use an icon, create an anchor and give it a class of `.fc-webicon`, as well as a class for the social network or service you want to represent. For example, Facebook would look like this:

Quick Start

You can grab the code pack that includes all of the icon files as well as an example page in Foundation 3 to show you how

If you do start playing around with SVG-- and I encourage you to-- you do want to make sure that you're optimizing these graphics. Even though they're lightweight, usually, you do want to make sure they're as small as humanly possible. And this site, the SVG Optimizer, will help you do that automatically. Additionally, we do have to worry about cross-browser support, so here is another technique called the invisible gradient technique, and this provides some cross-browser support for older browsers.



The invisible gradient technique: Cross-browser support for SVG (with image fallback) using CSS

Vector graphic formats such as SVG are great for high-density displays since their lines are clean at any zoom level. However, not all browser support SVG and there is not an obvious way to provide a fallback (using png, jpg or gif instead) for those browsers not supporting vector graphics.

These days I was experimenting with a technique for SVG fallback support that does not require Javascript code, server-side processing or browser-specific CSS hacks.

The invisible gradient technique

The idea is simple: browsers capable of using CSS gradients are modern enough to support SVG [illegible] composed of two layers, one being the SVG and the other being a gradient, only those browsers capable of understanding the gradient syntax will try to display the SVG.

http://pauginer.tumblr.com/post/36614680636/invisible-gradient-technique

Now, last but not least, it's easy to forget, with all these fancy new technologies coming down the road, that we have to go back to the old school, and we've got to optimize our graphics. We've got to make sure they're as small as possible.

So here is one particular solution that I recommend called ImageOptim, and the way that this works is you simply drag and drop your images into the application, and it will automatically optimize them. We'll take a look at how this works in the next section.

Additionally, there are some other options such as tinypng.com. This reduces or shrinks your PNG files, and, in a similar fashion, tinyJPG.com which does the same thing.

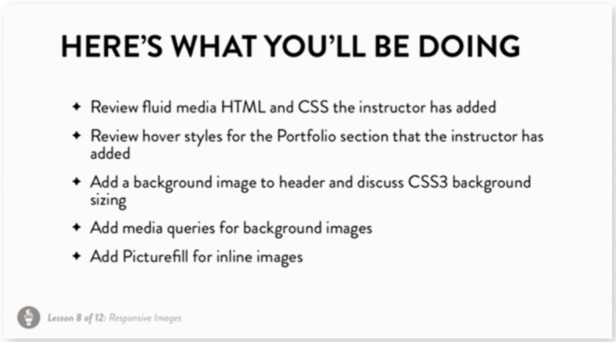
Additionally, there's one last thing to talk about, and this is a new image format. So WebP, this is a format that originally started at Google. It's open source. So, the deal here is that with WebP, if we can get much smaller images to begin with, well, maybe we don't need multiple images. Maybe one is just fine if it's small enough.

However, WebP is still a very new format. It's not supported in as many browsers as you might like. We can begin to use it now, but we definitely still need fallbacks with this one.

So coming up next, we're going to move away from theory and get back into practice. We're going to add a responsive background image as well as a responsive HTML image and explore some other image techniques. This is responsive web design.

CHAPTER 3: APPLYING WHAT YOU'VE LEARNED: RESPONSIVE IMAGES

OK, in this section you'll primarily be doing some coding, and specifically, we're going to be reviewing some HTML and CSS that I have added to the page. So, we need to talk about what that's done. And we're going to be reviewing some hover styles that have been added to the portfolio section, so you can see how that was done. And then you'll be adding a background image to your header, and we'll be talking about CSS3 background sizing, then adding media queries for our background images as well as Picturefill for our in-line images, both of which we talked about in the previous section.



HERE'S WHAT YOU'LL BE DOING

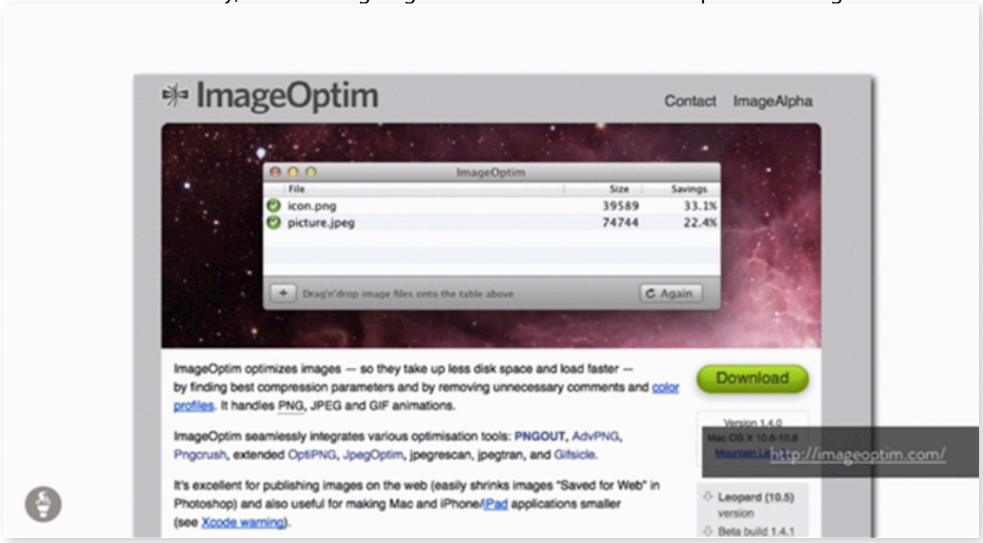
- Review fluid media HTML and CSS the instructor has added
- Review hover styles for the Portfolio section that the instructor has added
- Add a background image to header and discuss CSS3 background sizing
- Add media queries for background images
- Add Picturefill for inline images

Lesson 8 of 12: Responsive Images

But before we do that, let's just remind ourselves that image optimization is a good thing. I talked about this process in the previous section. Let's just take a look at how this works real quickly. You don't actually have to do this, all your images are optimized, but I'll show you how it works. So, essentially you've got a series of images here, and we're going to drag and drop them into the ImageOptim application, and that's it. So, this is going to automatically optimize these, not resize them, just optimize them, and you can see that you've got a savings of about 74kb out of 672. That's about 11%. So again, pretty significant. Definitely optimize your images and floss your teeth.

OK, now, while you've been busy, I've been a little busy myself. So, what I want to do is show you some of the styles that I've added to our portfolio page. Because I want to save some time, I'm going to focus just on responsive images for this section and some of the other work I've done with images. I'm assuming that you know how to do [this], or if not, you can pick it up pretty quickly.

So, let me explain what I mean here. Let's open up our document portfolio_start.html. Let's go ahead and save it as portfolio_work.html, and this creates a backup. After you do that, let's go ahead and open up the file. It's easier to show you what's happening here in the browser versus the code. So essentially, what we're going to see here is that our three portfolio images have a



File	Size	Savings
icon.png	39589	33.1%
picture.jpeg	74744	22.4%

ImageOptim optimizes images — so they take up less disk space and load faster — by finding best compression parameters and by removing unnecessary comments and color profiles. It handles PNG, JPEG and GIF animations.

ImageOptim seamlessly integrates various optimisation tools: PNGOUT, AdvPNG, Pngcrush, extended OptiPNG, JpegOptim, jpegrescan, jpegtran, and Gifsicle.

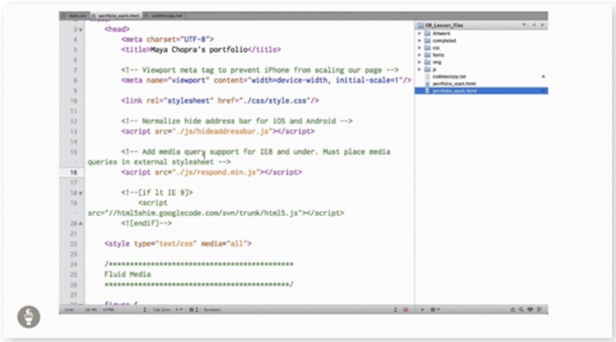
It's excellent for publishing images on the web (easily shrinks images "Saved for Web" in Photoshop) and also useful for making Mac and iPhone/iPad applications smaller (see Xcode warning).

http://imageoptim.com/

hover effect, so when we place our cursor over them, they're going to jump up slightly to show that they're active links.

Now, right now they don't actually go anywhere, but what we could do is put the address of the site that we're working on here, and let's just see how this works in the code. If we go back to our code and we scroll down into our internal stylesheet, what I've done is I've taken all of the internal styles from the last lesson, and those are inside style.css, but these are new styles.

Now, the most important of these is this whole block here that starts at fluid media and goes down. So, these figure styles here in the image tag below, essentially, what these create are an image foundation. Images with the CSS are going to display at their native dimension as long as there's enough room in the HTML container to do so. So when the browser window narrows or expands the images scale to fit.



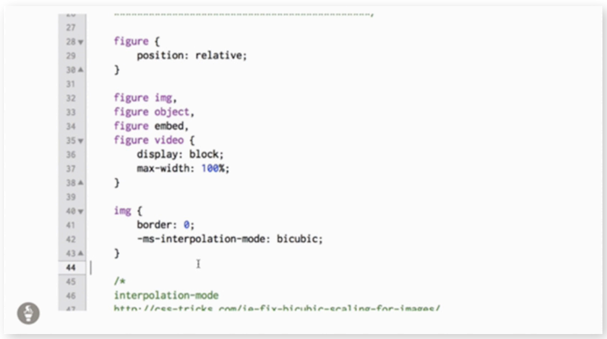
```
    <head>
        <meta charset="UTF-8">
        <title>Maya Chopra's portfolio</title>

        <!-- Viewport meta tag to prevent iPhone from scaling our page -->
        <meta name="viewport" content="width=device-width, initial-scale=1"/>

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

        <!-- Normalize hide address bar for iOS and Android -->
        <script src="./js/hideaddressbar.js"></script>

        <!-- Add media query support for IE8 and under. Must place media
queries in external stylesheet -->
        <script src="./js/respond.min.js"></script>

        <!--[if lt IE 9]>
            <script
src="//html5shim.googlecode.com/svn/trunk/html5.js"></script>
        <![endif]-->

    <style type="text/css" media="all">

    /*********************************************
    Fluid Media
    *********************************************/
```

This is actually not crazy new. This is very similar to what we did before with our `img {max-width}` style. The only difference is we're making this one a little more robust and so that we can put other content in here as well. So, you'll notice that there is an embed and a video tag and an object tag. This would allow us to put in media such as Flash or video, and they will scale effectively.

Additionally, you can see this style down here, the MS interpolation mode, this is essentially a fix for Internet Explorer which does not scale images natively very well, so we have to put in this little style to fix that. If you want to learn a little bit more about what bicubic scaling is then you could take a look at this URL here at css-tricks.com.

Now this code here is going to work well with this code, the portfolio figure hover, box hover. So, this targets those portfolio images that we have, and it's using that link style in order to create this hover effect. So, right there, that image offset is what's actually creating that jump when we hover over the images. Inside the HTML, the only thing we did here was we added this 'a href' link for these three figures. So here, here, and here. Additionally, if we go into style.css, the only thing that I did here was I added the id "#portfolio" to this "figure, .box" style, and this is going to target and make sure that only the portfolio figures have this effect, not every single figure on our page.

OK, that's it for the overview. Let's talk a little bit about what we need to do. So this screenshot here is our target, so currently our page doesn't look like this. We need to add this background picture of the notebook, and that's going to be a CSS background picture. Additionally, we're going to be adding an inline image of our fictional student, Maya Chopra, and that's going to go right here above this heading. So essentially, we need to get to this layout. Let's take a look at how we do this.



```
    figure {
        position: relative;
    }

    figure img,
    figure object,
    figure embed,
    figure video {
        display: block;
        max-width: 100%;
    }

    img {
        border: 0;
        -ms-interpolation-mode: bicubic;
    }

    /*
    interpolation-mode
    http://css-tricks.com/ie-fix-bicubic-scaling-for-images/
```

The first thing you want to focus on is this div with the id, div id="container-header", so this is our main focus. If we switch back to our style.css, and we find this rule for container header, we can see that there's a background color, and what we're going to do is we're going to go to this text file here where we can copy some code, and we've done this in previous lessons too. This just gives us some shortcuts so you don't have to type. And what I want to do is I want you to copy this line, background, and notice that it's got a URL to an image, photo-wrong.jpg. So, I want you to replace the background property in the external stylesheet with this text, and that looks like this.

Now again, I'm labeling this photo-wrong.jpg because I want to demonstrate something to you. So, we're going to be swapping this out a little bit later, but this image is not going to fit appropriately within our header. So, the size of this is inadequate for the size of the header. Let me just show you what I mean-- that. OK, we don't want our background image to be cropped like that. We want it to stretch and to scale the whole size of the header. Now, of course we could do that just by swapping out a larger image, but this is responsive design, and we're never quite sure how large that image is supposed to be, right?

So having said that, let's take a look at a nice solution using CSS3. So, what I want you to do, here, is to go back to the code_to_copy.txt file, and steal these three pieces of code-- so background-size for WebKit, Mozilla, and of course the regular background size-- so these three lines here, copy them. Jump back to style.css and paste that code right here after with 100% inside container header. Go ahead and save your file and let's just take a look at what this does, and then I'll talk about how it works. Bingo, there we go.

Let's take a look at the difference between the last one and this one. So, that same image is now scaling to the width and height of that entire section, and how did it do that? Well, let's just take a break from the code, and I'll go through this quickly.

CSS3 background-size-- this is a CSS3 property. It's got pretty decent support. Here's how it works. Background-size allows you to define the width and height of an image. So in one example, 50% is the value for width, and if you don't have a second value the height is simply auto. However, if you add a second value such as 25% here, that becomes height, so what this would do would be to scale an image 50% width and 25% height.

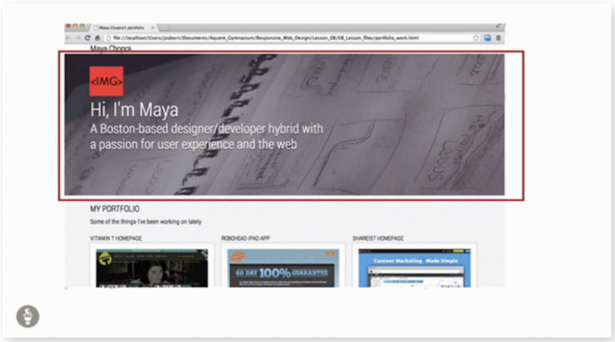
However, we're using the keyword here. So what the keyword cover does is it scales the image as small as possible while making sure that its dimensions are greater than or equal to the dimensions of the background area. That's exactly what we saw in our previous example. Now, there's another keyword as well that can come in useful. That's 'contain', and essentially, this does the opposite. It scales that image as large as possible while making sure the dimensions are greater than or equal to the dimensions of the background area.

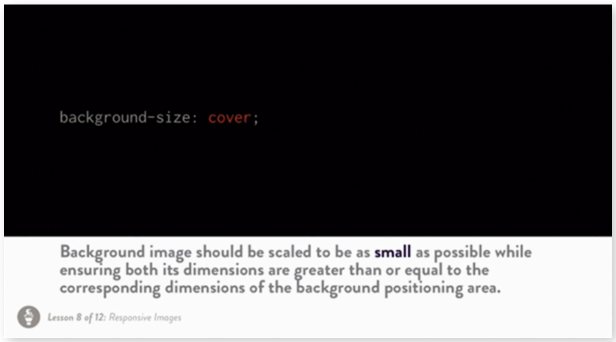
Now, I like to think of this background-size as an insurance policy,because it gives you a sense of security that your images will never be accidently cropped, so we want to do is the following. First of all, we're going to swap our higher resolution image for this one. So I just use that first image to show you how background-sizing works, so we're going to swap that out. Additionally, we need to add some space here where it says, "Hi, I'm Maya." We're going to be putting an inline image above this heading, and we need to make space for it.

So, first things first-- let's just go ahead and swap this out-- so bg-photo-l, for large, and now, while we're here, let's go ahead, and remove the width and height and replace it with padding-- so padding 10% and 0. So, the top and bottom padding will be 10%, and left and right will be 0, and let's go ahead, and see how this works. Save your file. And we'll see it in the browser, and that's it. So, the background image doesn't really change too much, but what we've done is we've added some padding on the top and bottom. And as we scale, we can see how this affects it.

But we need to stop here for a second, and think about this. So, that image that we just put in here, what size was it? It's about 350kb. So here in the smartphone view we really don't want to be loading that 350kb image. So, we talked a little bit about using media queries to target different screen sizes for different images. That's exactly what we're going to do. We're going to go ahead, and we're going to make sure that this image only gets sent to the appropriate screen size.

So, you're going to want to follow along for this one. It's not that tricky, but there are few things that I'm doing here. First of all, go to the code_to_copy.txt, and I want you to find the beginning of this media query, here, on line 10-- @ media screen and (min-width: 45em). So, go ahead, and copy that first line of code. Go back to your style.css, and put that right here above #container-header. Since we're using a media query, we need to make sure that we put in the curly brace-- {, so let's do that. And now, what I want you to do is I want you to copy that entire block, including that media query, and cut it. So, take it completely out of the external stylesheet and jump into the internal stylesheet.


Maya Chopra's portfolio
file://localhost/Users/josborn/Documents/Aquent_Gymnasium/Responsive_Web_Design/Lesson_08/08_Lesson_files/portfolio_work.html
Maya Chopra
<IMG>
Hi, I'm Maya
A Boston-based designer/developer hybrid with
a passion for user experience and the web
MY PORTFOLIO
Some of the things I've been working on lately
VITAMIN T HOMEPAGE
ROBOHEAD IPAD APP
SHAREIST HOMEPAGE



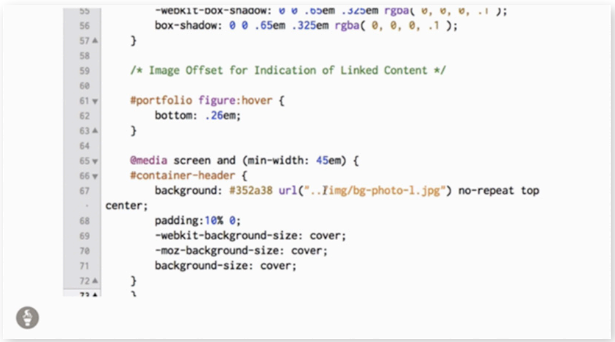
```
background-size: cover;
```

Background image should be scaled to be as **small** as possible while ensuring both its dimensions are greater than or equal to the corresponding dimensions of the background positioning area.

Lesson 8 of 12: Responsive Images

Now, the only reason I'm doing this is just so that we can see what new styles we've added. Typically, we would keep this in the external stylesheet, but for education purposes, I want to put it here, so we can see what's happening. The only downside of this is that this will break the link because it was in a different directory, so let's go ahead, and remove the dot dot slash (../) here to make sure that it's properly loaded. So go ahead, and make sure you Save All here, so we can save all the changes.

And let's think about what this is going to do. It's easier to see in the browser, so let's load this into a new tab, and we've got our image. It's loading as necessary on the wide screen. However, as we begin to re-size, when we hit that breakpoint, right there-- no background image, and this makes sense because we targeted this background image for screen sizes greater than, in this case, 720 pixels. Anything less than that is not going to have a background image because, well, we're telling it not to, so what we need to do is fill in that gap. We need to put a different image for everything less than 45em, and the good news is we've got that right here.



```
        -webkit-box-shadow: 0 0 .65em .325em rgba( 0, 0, 0, .1 );
        box-shadow: 0 0 .65em .325em rgba( 0, 0, 0, .1 );
    }

    /* Image Offset for Indication of Linked Content */

    #portfolio figure:hover {
        bottom: .26em;
    }

    @media screen and (min-width: 45em) {
    #container-header {
        background: #352a38 url("..Iimg/bg-photo-1.jpg") no-repeat top
center;
        padding:10% 0;
        -webkit-background-size: cover;
        -moz-background-size: cover;
        background-size: cover;
    }
    }
```

So, let's go ahead and use this entire block of code here in our code_to_copy.txt. So in this case, you want to copy everything-- so the whole media query, and let's go ahead and paste it back into our internal stylesheet. Again, right below the other media query, and what's the only difference between these two? One is min-width, the first one, and the second one is max-width. 45-em equals about 720 pixels, so what we're saying is anything less than 720 load this image, photo-s.jpg. That's the photo small jpg. It's a much smaller file size and, as you're about to see, it's got a different editorial content as well.

So, let's take a look at how this works in the browser. We can see it take place here. So, here's our original, right? At a certain point, boom, we jump and there's the change, right there. So, if we scale all the way down to the smartphone view, we can see what this would look like on that smartphone. And again, we've got a benefit here in that there is a reduced file size, and we can also change the editorial content again. Maybe we just want a hint of that notebook showing in the right corner. Pretty cool right? Now of course, unfortunately, we can only do this with background images. We can't do these with the image tag, so we're going to have to look at another solution for that.

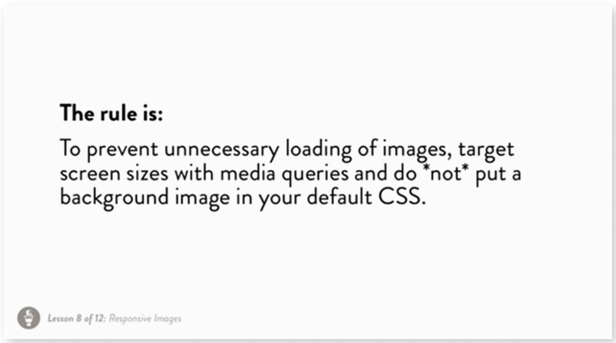
So, the rule is: to prevent unnecessary loading or processing of images, you need to target the specific screen sizes of media queries just like we did here. Be sure not to put a large, default background image into your CSS because this is going to create double loading of images, and that actually defeats the whole purpose. If you want to learn a little bit more about the specifics there, here is a URL, cloudfour.com-- examples, media queries, image test, and this goes through all the details about what exactly happens behind the scenes when images are downloaded.

OK, let's turn our attention to the inline image. What we want to do is we want to take an image of Maya, and we want to put it right here above that header, and so, we're going to turn to 'picturefill' to do this. Let's go into our code_to_copy.txt file. Scroll down and let's just take the styles first. So, we're going to take these styles here for 'maya' and 'maya img'. Copy them. Put those in the internal stylesheet. We could take a quick look at what these are doing. Essentially, this has a width of 25% for this whole div and the image itself has a width of 100.

OK, so that's the first step. We put in the styles. Now, let's go back to our text file, and let's take this whole chunk here for picturefill. First, we're going to take these references to the JavaScript that is required, so there's two pieces of JavaScript required, picturefill.js and matchmedia.js. We're going to put these here in our head section, and let's do it right here after the last JavaScript. Obviously, keep in mind that you have these files in your lesson folder because I put them there for you.

And now the big daddy... we're going to go down here, and we're going to take all this code starting with the starting figure, id="maya" tag, and going to the closing /figure tag, and we're going to put that right here below the header, and this includes all the code that is going to target our different images.

So, we briefly discussed how picturefill worked in the last section. We have these three different images, headshot-s.png, headshot-m.png, and headshot-l.png. Essentially, I just doubled each of these images, so the small.png is 100 by 100, medium.png is 200 by 200, and l.png is 400 by 400. Notice that the medium one is being targeted at a media query of 45em, and the large one is being targeted at a media query greater than 60em. We do need a fallback here, so we're using a 'noscript' tag, and the fallback will actually be the smallest one-- so headshot-s.png.



The rule is:

To prevent unnecessary loading of images, target screen sizes with media queries and do *not* put a background image in your default CSS.

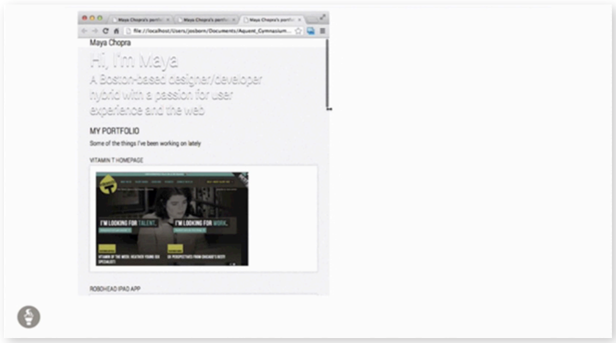
Lesson 8 of 12: Responsive Images

OK, so let's see how this works. Let's choose File, Save All. Make sure that everything is saved. Check this out in the browser, and there we can see our image. So, there it is at its largest size, and we might need to play around with the padding of this at some point, but let's go ahead and resize and just make sure that this works. So, there is our medium, and if we keep going, there is our small.

So, how do we know this is actually working? So, let's peek into this a little bit using the developer tools. So, I'm going to resize this to the larger size, and I'm going open up our developer tools. We want to peer all the way down here, and it's going to be a little tricky to see, so I'm going to zoom in just a second. But essentially, I'm going to find this whole div here with the data-picture, and you can see that img headshot-l.png file is being used right now.

So, as I began to resize, notice right there it changes to the medium one, and as I keep on resizing, it's going to change to the small one-- right there. That tells us it's working. The most relevant image is being loaded at any given breakpoint, and of course, if we came to the site in a smartphone to begin with, we would only get the small image, not the medium or the large ones.

OK, that's it for now. Coming up next in the next lesson, we talk a little bit about our friend, navigation. That's right, navigation-- super important. We've got to tackle it, and we'll be doing that shortly. Let's just summarize briefly what we did here. We added a fluid media foundation for flexible images and other media. We took a look at CSS3 box sizing, and it's primarily used as an insurance policy to prevent gaps in unknown sizes. We also targeted screen sizes using media queries. Be sure not to add background images to your base CSS, and finally, we added picturefill to serve inline images as needed for different screens.


file://localhost/Users/josborn/Documents/Aquent_Gymnasium...
Maya Chopra
Hi, I'm Maya
A Boston-based designer/developer hybrid with a passion for user experience and the web
MY PORTFOLIO
Some of the things I've been working on lately
VITAMIN T HOMEPAGE
I'M LOOKING FOR TALENT.
I'M LOOKING FOR WORK.
ROBOHEAD IPAD APP

A lot of stuff going on there. To reinforce your knowledge of what we just talked about, we've got some homework. As always, there's a short quiz. Assignment number two-- be sure to read the use cases for responsive images, and I pointed out the URL earlier. Here it is again, usecases. responsiveimages.org. Please, please, please go to the site. Read it. There's a lot of great information here. Be sure you understand the scenarios that they're proposing, and if you don't ask on the Forum.

Assignment number three-- add a responsive background image or images to your page. Be sure to use background sizing as we did here, and be sure to ask questions if you run into any problems. You can do that, of course, on the Forum.

There is extra credit. If you want to use picturefill, if you want to get some experience doing that, try using it on the thumbnail images for the portfolio site. Be sure to optimize and resize the images. You're going to need those. If you want, you can use our portfolio site that we've been working on or your own site if you've been working on it up to this point and if it's relevant. That's it for now. I will see you in the Forum, and see you in the next session.