GYMNASIUM

RESPONSIVE WEB DESIGN

Lesson 12 Handout

Responsive Web Design Workflow

ABOUT THIS HANDOUT

This handout includes the following:

- A list of the core concepts covered in this lesson.
- The assignment(s) for this lesson.
- A list of readings and resources for this lesson including books, articles and websites mentioned in the videos by the instructor, plus bonus readings and resources hand-picked by the instructor.
- A transcript of the lecture videos for this lesson

CORE CONCEPTS

1. The traditional "waterfall" approach to web design/development involves a relatively linear workflow as follows: a.) research (including UX concerns) b.) creation of wireframes c.) static mockups in Photoshop d.) Coding in HTML/CSS/JavaScript e.) testing and launch. This workflow is inefficient in Responsive Design due to the lack of flexibility with image-based (Photoshop) mockups as well as the number of different layouts typically introduced with media queries.

2. Responsive Design workflow benefits from extra time spent designing "from the content out." This typically includes a higher priority on structuring text and organizing content before moving to a wireframe or prototyping stage. Additionally, a major trend is the move toward designing in the browser earlier and more frequently, as well as testing in mobile devices earlier and more frequently. This method reduces the reliance on Photoshop mockups and allows for increased feedback and evolution before spending time and effort on moving to production code.

3. The complexity of Responsive Design is reflected in the number of tools used in design, such as browser developer tools, text editors, GitHub, third-party web font hosting services, and CSS pre-processors, as well as a general lack of one single software solution for creating responsive websites.

4. Responsive layouts and web pages can be handcrafted by focusing on creating fluid grids and images, responsive typography, and custom media queries based on the range of devices and scenarios your project requires. CSS pre-processing is an optional, but increasingly more popular, way to address the complexity of these responsive components. CSS pre-processing makes responsive design easier by automating CSS in a number of consistent ways. The benefits include the ability to save time, to perform complex math behind the scenes (so you can focus on more enjoyable tasks such as refining your grid layout), and to speed up collaboration among team members.

5. A few CSS pre-processors exist, such as Sass, LESS, and Stylus. Many of the concepts are the same across the systems. A pre-processor has a specific syntax that ends up generating CSS code across your stylesheets. We focused on Sass in this lesson, which includes variables (a variable is defined once and then can be reused as a CSS property value throughout your stylesheet) and mixins (a mixin defines an entire CSS class that can be reused throughout a stylesheet).

ASSIGNMENTS

1. Quiz
2. Go finish your responsive portfolio already!
3. No really. Finish it *and* put it online. It doesn't count unless it's online.

RESOURCES

READING

- http://www.vitamintalent.com/vitabites/top-free-portfolio-sites

   An article on the "Top 5 Free Portfolio Sites"

- http://responsive.rga.com

   A list of the latest links, news and thinking in responsive design

WEBSITES MENTIONED IN THE PRESENTATION

- v2.happycog.com/contact/contact.doc
- http://dillinger.io/
- http://alistapart.com/column/windows-on-the-web
- http://sass-lang.com/try.html
- http://css-tricks.com/sass-vs-less/
- http://twitter.com/rwd
- http://rottercreativegroup.com/
- http://www.j-parker.co.uk/
- http://www.julianabicycles.com/
- http://mobile.smashingmagazine.com/2013/05/29/the-state-of-responsive-web-design/

INTRODUCTION

(Note: This is an edited transcript of the Responsive Web Design lecture videos. Some students work better with written material than by watching videos alone, so we're offering this 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.)

Welcome to Responsive Web Design, an online course developed by Aquent. Responsive Web Design-- or Promote Yourself Responsively: Build a Portfolio For All Devices. This is Lesson 12, Responsive Workflow, and as always at the end of this lesson, there'll be an assignment and a brief quiz.

Be sure to use the pause button at any given point in order to stop, work with code, or simply absorb the concepts. And finally, if you have questions, be sure to hit the Forum in the classroom. There we'll have teacher assistants, instructors, and most importantly, your other classmates available to help you out.

Let's just talk a little bit about the road map for this lesson. We're currently in Chapter 1, The Intro. Chapter 2, the Evolution of Web Workflow and Chapter 3, Design from the Content Out-- there'll be a presentation, some demos, but no coding for you.
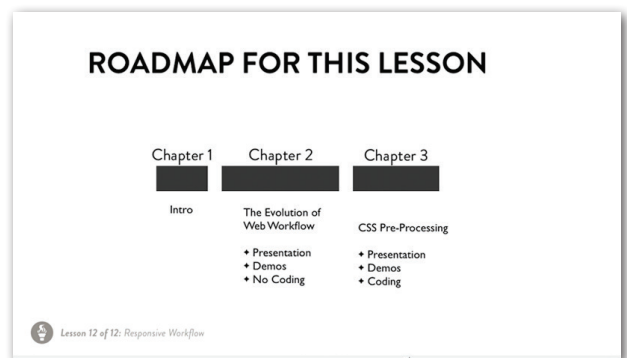
THE EVOLUTION OF WEB WORKFLOW

So to begin, we're going to talk about the evolution of web workflow, and specifically, we're going to be comparing the classic workflow of a web designer to a responsive workflow. The point here is to take a look at the concepts, the tools, and the best practices out there to help you succeed in your future responsive projects.

So, let's begin by talking about workflow. What do we mean by that? The truth is, everyone has a workflow. It's kind of like most everyone has a desk, and some desks are messier and less organized than others. So, what we're really talking about here is not just a generic workflow, but an ideal workflow.

So, the first step is to talk about the benefits of having a good workflow. Now again, this is kind of a review. I'm assuming that you might know some of these. These are called the usual suspects, but let's just make sure we're all on the same page here.

When you've got a great workflow, you know exactly what you need to do at any given point, where you're headed, and you're not making it up every single time. Another benefit is when you've got a structure in place, you can do more of the stuff you like to do, so if you like to work with layout, and you like to work with grids or graphics or whatever it might be-- if you have a good foundation, that takes away some of the complicated, annoying things that you have to do or make up on the spot. And you get to do the fun stuff.

Another benefit is streamlining collaboration. So when all the people on your team are familiar with the work-flow, and everyone is doing exactly what they need to do, it makes the whole project go by faster, and that leads to the next step. When a project goes faster, generally speaking, there are lower costs, more profit for you. You can do more of the fun stuff when you get out of work, and this is really the whole point of what we do, right?

And lastly, we have repeatable results, so when you've got this great workflow, you stick with it. You get projects going. Everyone can predict exactly when something will start and when something will end, and in general, if it isn't broke, don't fix it.

Let's go ahead now and talk about the classic approach, and this is sometimes called the waterfall approach-- and again, chances are you may be familiar with this term and this concept. It's pretty straightforward.

The way it works is, we've got step one. You do some research and planning. Step two, you create some wireframes or prototypes. Step three, you make some static mockups in Photoshop of your website.

Next, you send those mockups to someone else, or maybe you're doing it yourself, and you code those mock-ups in HTML/CSS and JavaScript. And finally, you launch. Now obviously, all workflows have their differences, but I think it's fair to say that this is a commonly accepted practice.

Now, another way to describe this is maybe not the waterfall approach, but the assembly line approach. Projects go through a series of stages, and they get passed along from one person to the next as needed. Of course, the dilemma with this is, if there are problems that start at the top or the beginning of the assembly line, they sometimes ripple through to the very end, and there's no heading back.

Responsive workflow almost forces us to take a different approach. We mentioned this in the last lesson, but there's a very common process of feedback and evolution. You start with something like a prototype or wire-frame, and you send that along to members of your team or clients.

They go ahead and they give you some feedback. You take that feedback, and you pour it into your prototype or wireframe. And eventually, you end up with the final product.

In the classic workflow, Photoshop or Photoshop mockups are often a huge part of this: feedback and evolution gets poured into this fixed-width, static file. So, Photoshop does not offer us responsive or liquid layouts. Everything is fixed, and the problem with that is that it doesn't really reflect the true nature of what you're building.

So, let's talk a little bit about the changes that are part of responsive workflow, and to do that, we have to start with some things that maybe aren't so different-- topics such as research.

Research is something that you would do regardless of what workflow you're using, hopefully. Questions like, why would someone come to your site? What's the main goal you're trying to achieve? Who are your main competitors?

There's a lot of other lessons here, and I'm not trying to do an exhaustive survey of this. In fact, I'll give you a hot tip. If you do a Google search for project planners, you will come up with some nice results, such as the design agency Happy Cog here, that provides project planners for their clients, and you can go ahead and make use of these, and apply them to your own projects as well.

So, let's talk about planning. Again, there's a lot of aspects of planning. I'm just pulling out two here. One is user scenarios.

So, user scenarios are part of UX design, most commonly, and the way it works is this. You assign a character or a name, an age and occupation to a fictional person-- such as Julie, age 27, a photographer. She's on a break during a shoot.
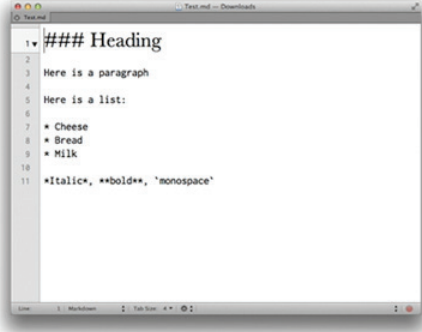
Maybe she met you at a party, and she's using her iPhone to look at your portfolio page and then contact you. And if you set up this scenario, this is going to help you build your page, and maybe change the way you design it based on this workflow.
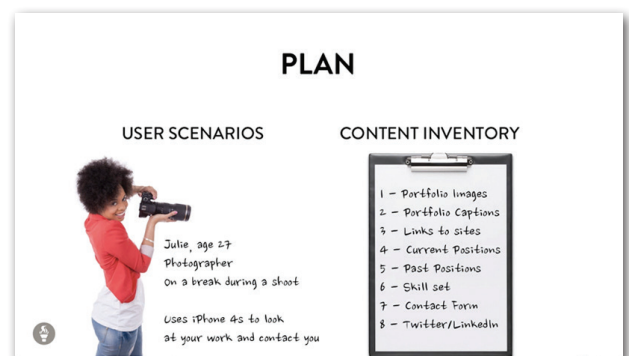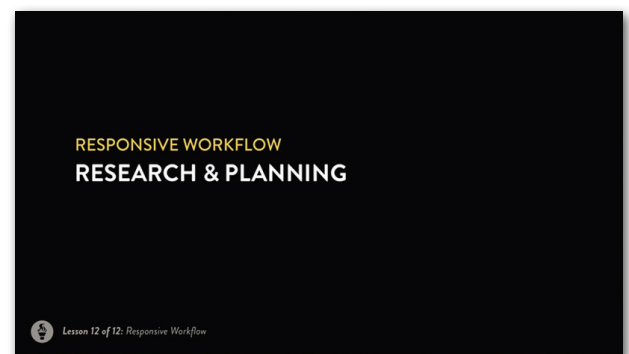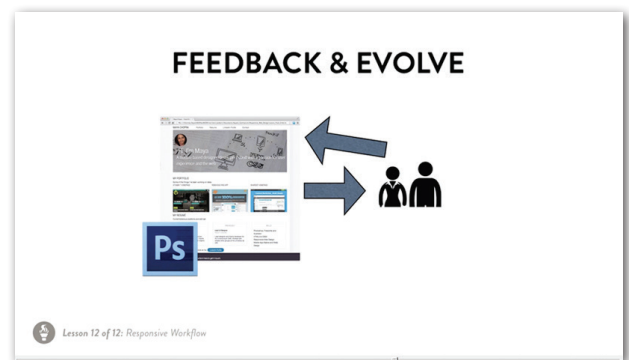
Another useful planning tool is a content inventory. So in this definition, a content inventory is simply a list-- a list of all the things that might appear on a page. So again, in the portfolio site we've been using as an example. You might have portfolio images, captions, links to sites, your current positions, past positions, your skill set, contact form, and so forth. By defining this content inventory, you make sure that everyone's on the same page. And you can begin to plan your layout and design before you even start coding.

DESIGN FROM THE CONTENT OUT

Now, one thing that responsive workflow really benefits from is this concept, Designing From the Content Out. And again, this is not necessarily radically new, but the way that this works is we start with text design. You get as much real content as you possibly can upfront, and you begin to mark it up, or you begin to structure it.

And this has some real benefits. In addition to forcing you to deal with the content and to work with it, which is going to create a more realistic site, you can also start thinking of, how does this text look in a single column for mobile? And it also forces you to think about accessibility. So the accessibility, the raw text, is there to begin with.

Now, you can do this in standard HTML or a word processing document, but I do want to show you a couple options that are fairly popular out there that really force you to think about the structure of the content. So, I'm going to do a quick demo here of a syntax language called Markdown.

And the way I'm going to do this is I'm going to go to this site, dillinger.io, and what this is is a Markdown converter to HTML. This will make a little more sense as we begin to see it. So, essentially, Markdown is a language, and you can see it here on the left. It's very, very simple text.

You don't have to know HTML tags to create an HTML document. What you do is you use very simple syntax, and you can see some of that here on the left-- so, dashes, brackets, and so forth, but really no tags, and then those get converted to HTML.

One of the driving forces for the creation of Markdown was so that everyone could create HTML documents without having to know HTML. It's a very super, simple way to do this, and it means lots of people can collaborate on the same document. And as we can see here, by going into the Utilities Section, it converts everything to HTML. Very clean, easy to use HTML.

Lesson 12 of 12: Responsive Workflow
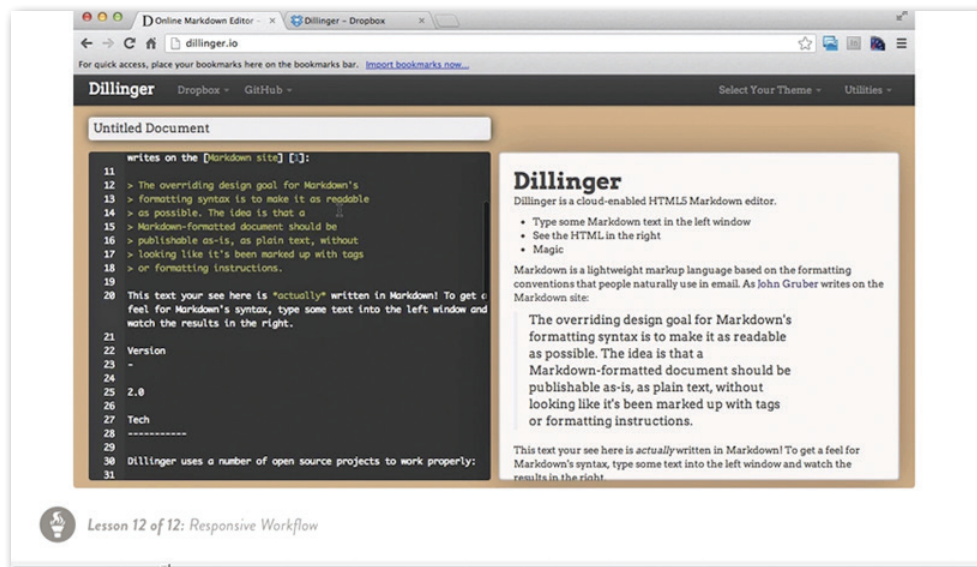
So, this page here has some default HTML. I'm going to strip that out, and I'm going to just call this 'Test', and let's just do a really quick run-through of how to write Markdown. As you can see here, I just put in three hash marks, or three pound signs, and that's actually a heading three-- h3. That will be converted to a heading three-- h3.

Two of those is a heading two-- h2. One of those is a heading one-- h1. So again, once you learn that syntax, [it's] really easy to repeat and remember. I'll go ahead and add a paragraph here, so nothing special needed. Here's a paragraph that will be converted to a paragraph. I'll go ahead and add an unordered list by simply putting in these asterisks here, so we'll put in three items.

And again, you can see that the bullet points are being created on the right-hand side. I've gone ahead and cheated a little bit here. I've just pasted the syntax for italic, bold, and even monospace. Again, on the right-hand side, you can see how those have been formatted.

And we'll stop for now, and I want to show you some of the utilities that are available in this particular web application. It's kind of nice. You can export this as a Markdown file itself, and open it up in a text editor. I'll show you that in a second. You could export it as an HTML file, and if you need to learn more about the syntax, you could look at the cheat sheet. So when we click on the cheat sheet here, you can see the basic syntax. And this is again, really simple. I truly believe you could teach someone how to create Markdown in about 10 minutes.

So, having said that, this web application also has some other nice features. So, you can go ahead and type whatever you put in here and save it to Dropbox. So there's a link here between this file and my Dropbox account. There it is. I'm going to open it up. That's what it looks like.
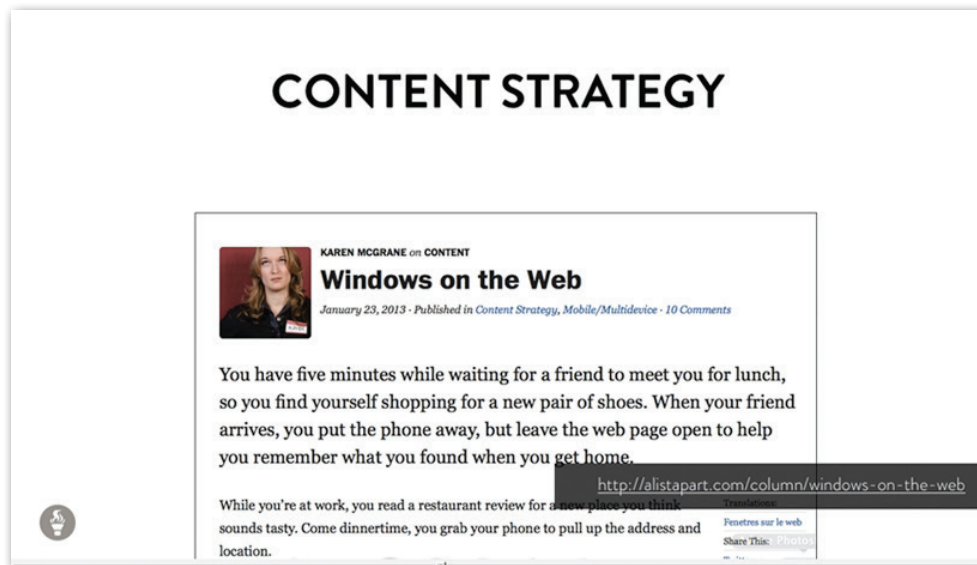
But I can also download it, and when I download it, I'll open it up in my text editor, and there it is. So in theory, I can go ahead and pass this onto someone else, or convert it to HTML, or what have you.

Now, another way to frame the 'content out' philosophy is to actually focus on the content. Crazy idea, right? Well, there's a whole category of web design and development called content strategy, and I don't want to dive too deep into it now. It could easily be its own lesson.

This article here by Karen McGrane is one I highly recommend, and the URL is down there at the bottom-- and there's also a link in the classroom. To make a long story short, content strategy is simply thinking about how your content can be used in all the different contexts that we see today, as well as related concepts, such as who are you writing for? What's your audience? What sort of tone are you using? Where are the different places your content might be seen? HTML, email, what have you.

So, we move on now to things like sketching. Again, we're still staying away from code as long as possible. We're working in the pre-production mode, and we're sketching out some ideas. What does our page look like in the mobile view, or the tablet view, or the desktop? The benefit here is that this is very fast, easy, and disposable. We're not really investing a lot of time and money to get to a consensus.

So, the real dilemma with responsive workflow is that when we're creating prototypes or even our final versions, we're not just creating one way, we're typically creating four. And again, this is something we mentioned in a previous lesson, but this increases the amount of work. And if you think about what you would need to do in Photoshop to come up with mockups or prototypes in this sort of workflow, well, it would take a lot of time. You'd have to make a mobile document. You'd have to make a tablet. You'd have to make at least two desktop versions, and if you make changes in one, you'd have to make them in all the other. And this creates these ripple effects that are very hard to keep track of and can create serious problems when one version starts to divert from the other.
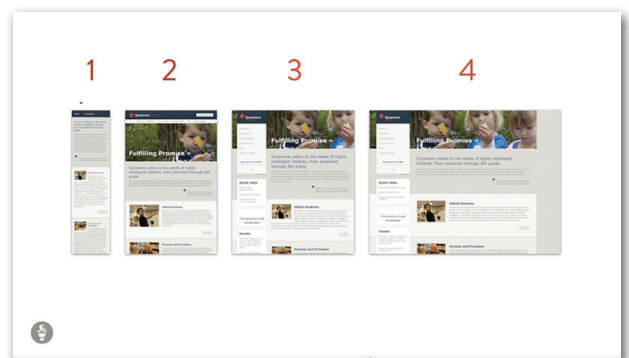
So, it's not to say that this is wrong in all cases. Mockups still definitely have a place in certain workflows, but if I was to isolate one major difference between the classic workflow and the responsive one, it's this. You tend to design in the browser earlier and more, so let's just break this down a little bit. Again, the traditional waterfall method here in gray-- these are the things that we've really covered already.

Then, we would move onto static mockups in Photoshop. We would code in HTML/CSS and JavaScript, and then we launch. So, let's compare that to the responsive workflow. Again, we still have the research and planning part, but we move to visual design in the browser much earlier. So in this case, HTML/CSS is being used. Photoshop is now a supporting character. It's not the main character.

And so we create graphics as we need to. And then, we begin to test in devices very early on. Once we test in those devices, we get some feedback, and we bring it back to step two. So in many ways, having steps here, it's not that it's invalid, but it's much more organic. There's a loop here that we work with.

After evolving the visual design, you then go back again to testing and devices, and you get more feedback-- and this loop might continue for a while, until you're satisfied, until you have something solid. And then, and only then, do you move from development to production in certain cases.

Traditional "Waterfall"	Responsive Workflow
1. Research & Planning	1. Research & Planning
2. Create Wireframes	2. Visual Design in Browser
	(HTML/CSS foundation & graphics created as needed)
3. Static Mockups in Photoshop	3. Test in Devices
4. Code in HTML/CSS & JavaScript	4. Get Feedback and Evolve
5. Launch	5. Move from development to production

TRENDS IN RESPONSIVE WORKFLOW

LESS RELIANCE ON STATIC MOCKUPS

WORKING FROM CONTENT OUT

MORE TOOLS USED, NOT LESS
(TEXT EDITOR, DEVELOPER TOOLS, PHOTOSHOP/ILLUSTRATOR, GITHUB, SASS)

NO (SINGLE) INDUSTRY STANDARD SOFTWARE

Now, essentially, you have learned the responsive workflow by going through these last 10 or 11 lessons. This is essentially what we did. We worked with typography. We added in images. We always tested in the mobile view, and we had fluid grids and images. And so this response workflow is now something that you're familiar with.

This concept still makes a lot of designers and developers nervous, though. And for some folks, it almost feels like walking on the tightrope without the net. Some of this, I think, is just a fear of change, perhaps, but I also think some of it is just the way that the technical skills that we're used to have changed-- and the workflow has changed.

So, to sum it up, we have these trends in responsive workflow. So, there's less reliance on static mockups. We work from the content out. More tools are used, not less. So again, this is kind of an important part. It used to be in the "old days," you would work with a program like Photoshop, or maybe Dreamweaver, or a text editor, and that was it. You were done.

These days, it seems like we have more tools-- so text editors, but also the browser, developer tools. You can also use Photoshop, but maybe Illustrator. Then you might also be referencing JavaScript or other files that are on GitHub, and you might also be working with a CSS pre-processor, which is something we're going to be talking about just a little bit later. But the point is there is no single industry standard software anymore.

So, the industry standards are in a state of chaos right now. In many ways, this is kind of exciting, because we're all figuring out what works for us as professionals, and it's kind of cool to be a part of this change, in my opinion. OK. So coming up next. We take a look at CSS pre-processing, and more specifically, how it makes responsive design easier. And of course, this is the last section of the last lesson. This is Responsive Web Design.

CSS PRE-PROCESSING

So in this section, we'll talk a little bit about CSS pre-processing, and in particular, you'll be learning why CSS pre-processing makes responsive design easier. And you'll also be learning the bare essentials of Sass syntax, and we'll be talking just what that means in a second.
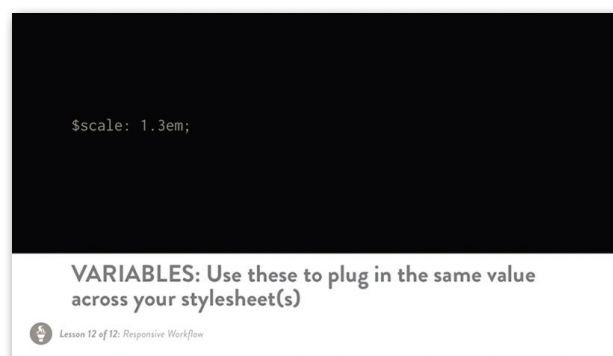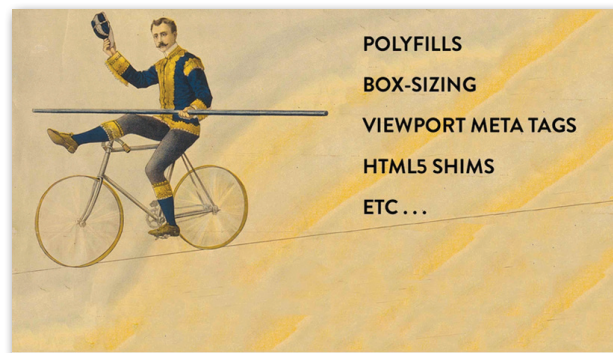
Now, the objective here is to introduce you to these concepts of pre-processing in the hopes that you'll see these benefits, and you'll begin to explore further. I should mention that this is quite a large topic. It's a little dangerous to introduce this in the last lesson, but I do believe that it's useful, and as you'll see in a second, it opens up the door to a whole new chapter in designing responsive websites.

POLYFILLS
BOX-SIZING
VIEWPORT META TAGS
HTML5 SHIMS
ETC . . .

So having said that, let's just go back to the beginning, or more specifically, Lesson 2, where I talked about how responsive web design was hard. And as we've walked through these last lessons, you can see that there are a number of things that are complex in responsive design, and that can translate to hard. So, things like responsive typography, fluid layout, media queries, all of these concepts introduce potentially complex calculations that you have to do when you're dealing with fluid layouts and percentages, and media queries add extra code, for example. So, I think it's fair to say that when you add all these up, responsive design is complicated.

Now hopefully, if I've done my job right, it's less complicated than it would have been otherwise. But again, we've got all these new concepts that are related to HTML5 or CSS3, and we've got terms like polyfills, box-sizing, viewport meta tags, HTML5 shims, and more. When you take a look at all these concepts and the various dependencies that your sites require, it's no wonder that sometimes it feels like creating responsive sites is a little bit like balancing on that tightrope, but this time on a bicycle.

So, what's one way to simplify all these complex elements? We come to pre-processing, the goal of which is to make your new responsive life a little easier, so let's just frame the discussion and talk about the traditional workflow and how pre-processing fits into it.

So we're going to start at the end. Here, we have a finished website or a web page, and so the question I pose is, how do we get to here? And the answer is, well, we have an HTML structure and we've got some standard CSS. Of course, all of the elements that we just talked about are there-- media queries, fluid grids, and so forth.

`$scale: 1.3em;`

VARIABLES: Use these to plug in the same value across your stylesheet(s)

Lesson 12 of 12: Responsive Workflow

Now, what pre-processing does is it adds another step before the standard CSS, so what we have is pre-processor syntax. So, this is very particular and specific syntax that gets converted to standard CSS. So, when you're working with pre-processors, you're working with code that kind of sort of looks like CSS, but it kind of looks like a programming language also, which is essentially what it is.

So, to make this a little more clear, let's just dive into the specifics, and we'll talk about this concept here of variables. So, variables look like this. We've got a variable called scale, and there's a dollar sign in front of it, and the way that variables work is we use them to plug in the same value across your style sheets. A variable is defined once and then reused multiple times.
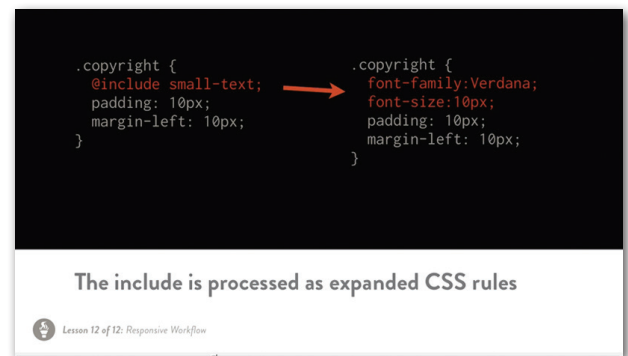
So again, scale, 1.3em. We've got a standard CSS rule. The value for Font is scale, or the variable. We plug that variable in, and it gets converted to the real code at a certain point.
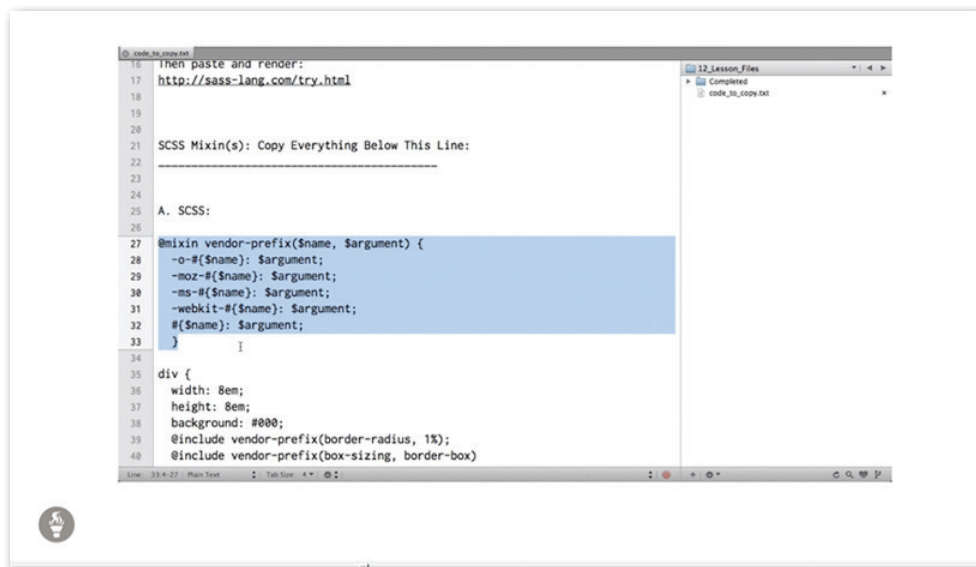
So, to illustrate this a little further, let's actually do this, and you can do this yourself. You can go into your Lesson file and choose code_to_copy.txt, and here we have two variables, one for scale and one for sans-serif fallback, and the value there is Helvetica, Arial, sans-serif. Now, our CSS has a font value of scale and the sans-serif fallback, so we're just going to be plugging those variables in. And if we look down at the bottom in the Article tag, we've also got a variable for scale for the Padding property.

So go ahead and copy everything here as I've done, and then we're going to go to a site where we're going to convert it. What's the site? The site is here, sass-lang. com, and this is the headquarters for Sass, and you can go in and explore the documentation and some other things on your own time.

But what we're going to be doing here is we're going to be using this online rendering system. So we're going to be pasting in the Sass code into this place here, and this is just a placeholder text so let's strip that out, and let's go ahead and paste the code that we just had there. Press Render, and that gets translated to CSS. So this is very cool. And again, this is just a variable.

Let's go ahead and look at another concept. This is called a mixin, and mixins are sort of similar to variables, but they are used to define styles that can be reused through your style sheets. Now, this one here, actually, is a nested mixin, and we'll talk about that in a second, but what we can see here is we've got the @mixin syntax, and we're calling it small-text, and what's nested inside there is font family Verdana, size 10 pixel.


```
h1 {
    font: 1.3em;
}
```

The variable gets converted to "standard" CSS

Lesson 12 of 12: Responsive Workflow


```
@mixin small-text {
  font: {
    family: Verdana;
    size: 10px;
  }
}
```

Mixins: define styles that can be re-used throughout the stylesheet. You can even nest mixins

Lesson 12 of 12: Responsive Workflow


```
.copyright {                    .copyright {
  @include small-text;            font-family:Verdana;
  padding: 10px;                  font-size:10px;
  margin-left: 10px;              padding: 10px;
}                                 margin-left: 10px;
                                }
```

The include is processed as expanded CSS rules

Lesson 12 of 12: Responsive Workflow

```
16   Then paste and render:                                    □ 12_Lesson_Files          ▾ ◂ ▸
17   http://sass-lang.com/try.html                             ▸ □ Completed
18                                                                □ code_to_copy.txt          ✕
19
20
21   SCSS Mixin(s): Copy Everything Below This Line:
22   ------------------------------------------
23
24
25   A. SCSS:
26
27   @mixin vendor-prefix($name, $argument) {
28       -o-#{$name}: $argument;
29       -moz-#{$name}: $argument;
30       -ms-#{$name}: $argument;
31       -webkit-#{$name}: $argument;
32       #{$name}: $argument;
33   }
34
35   div {
36       width: 8em;
37       height: 8em;
38       background: #000;
39       @include vendor-prefix(border-radius, 1%);
40       @include vendor-prefix(box-sizing, border-box)
```

So, you can probably guess what's going to happen here, but if not, I'll show you. Let's say we've got a standard class, in this case, copyright. We're now going to use this other syntax called an '@include', and we're going to include that small-text mixin. So again, it's a very similar concept to the variable. Wherever this shows up, it's going to expand or be replaced with the final code.

So again, this is very cool. What this means is you can go ahead and define one of these mixins, put it into your styles, and you've got multiple locations throughout your style sheet where these group of rules will be used. The benefit here is if you need to make any changes, you could go back to the original mixin, make a tweak, process it again, and get code spit out through your entire style sheet.

Let's just take a look at an example here. Go into that same code_to_copy.txt file, and we've got mixins here, and this one is designed to take care of the vendor prefixes that we've been talking about. So, we know that some of the cutting-edge CSS3 properties need to have specific prefixes for different browsers. What properties? So for example, border radius or box-sizing, so we can see here that there's two includes for those.

So, let's just see how this works. Go ahead and copy all that code, go back to this renderer-- I've reloaded the page so we can just zero that out-- paste it, and render. What happens if we look at this div code now? That div code has written out all those browser prefixes for us, almost 10 lines of code, more or less. This is great, really cool. We just don't want to have to write all that by hand. And you could easily make a mistake and leave a dash out or what have you, so this just takes care of a lot of problems behind the scenes.

So what do pre-processors do for us? What are the benefits? Well, they let you focus on the fun stuff, so if you can take care of all the browser prefixes automatically, then you can spend some time actually using those cool CSS3 styles. They also save time, as we've seen here. They also streamline collaboration, so if multiple people on your team are all familiar with the syntax, then you can work very quickly updating and creating new style rules.
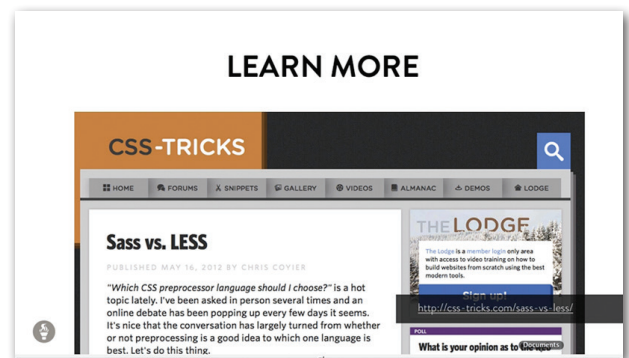
All that sound familiar? These are all similar to the benefits of an improved workflow. It's just that we're amplifying it. We're turning it up a notch. What this means is once you learn one of these pre-processing languages, you can be more efficient, and all of those good things that we just talked about.

Now again, we're including this in the discussion because in a previous lesson we talked about a lot of grid systems, these days, are beginning to use pre-processors. So at the very least, you should be familiar with what's happening behind the scenes and how they work.

Having said that, if you want to learn a little bit more, if you want to explore this world-- again, we just touched the surface-- this article here at css-tricks.com is a good place to start, and you'll see that there are actually a number of different pre-processors. This one here talks about Sass, which is what we looked at, versus LESS, which is another CSS pre-processing language.

So again, I don't want to get into the details here because that opens up a can of worms, but for the most part, just realize there are no standards yet for CSS pre-processing. We like Sass. It seems to be a very reliable option. But again, you'll have to look at this on your own.

One thing is clear, however; Pre-processors are going to be an increasingly large part of many people's workflow because, of course, the responsive train has long ago left the station. There is no doubt in my mind that pre-processors are going to become more and more a part of people's workflow as mobile devices and responsive design begins to get more traction-- and it will get more traction.

Here is a recent graph in June 2013 that's projecting the amount of data traffic by application type over the next 10 years or so. So, here we are in 2013, and what you want to look at is that red curve going upwards. That's the web browsing graph. And essentially, what this is saying is we're just going to be seeing more and more web browsing, more and more download of data, more and more people using these devices. So really, this is a no brainer. Responsive design is here to stay, and I hope that I've helped you get a little further along the way.

So on that note, remember the goal of responsive design is to make websites that look good and work on all sized screens and devices, but we can't lose sight that it's not just the code that's important, but it's whether our websites enhance and improve people's lives. Think of it this way. Somewhere out there in the world, there's a little girl. The only way she can access the internet is through her phone, and she's going to visit your website today, and in five years, she's going to be your customer, and in 10 years she'll own her own business-- all because of a web page she happened to see one day on her phone, and that web page was yours.
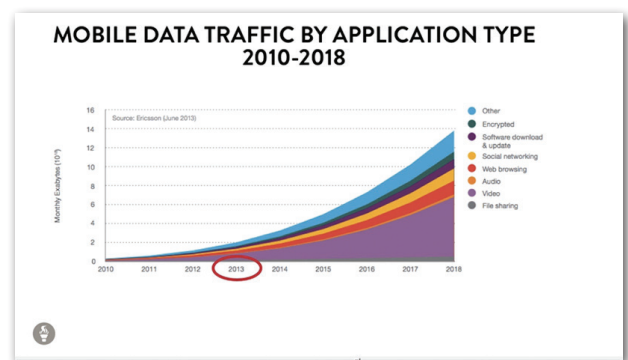
So, I hope you remember this. I hope this course has helped you get to the next level. It's certainly been my pleasure to teach it. So coming up next, some homework for you. Just kidding, no homework. Well actually, yes, homework. So, there is a small assignment, a short quiz, as always, at the end of this lesson, but no other assignments.

We do want to talk about next steps-- so the next steps are you want to show off. You want to show off your portfolio. You want to show off your responsive design skills, and how do you do that? Let me give you some leads here.

So here we have an article at vitamintalent.com-- and the URL is there at the top; It's also provided in the classroom-- and this article is for the top five, free portfolio sites. It's an extremely popular site at our friends at Vitamin T, and let's just walk through this. They did a survey and they took a look at some of the most used and important portfolio sites out there.

So number one is Coroflot, or "coro-flow," depending on your preference-- so there's one portfolio site, very popular. Another one is at behance.net, so the Behance Network. Again, we've got number three here, Carbonmade at carbonmade.com. Number four, Cargo Collective, and the dark horse coming in at number five, but a favorite of Gymnasium, or at least of mine, is Dribble-- dribble.com.

So, go ahead and submit your portfolios on one or all of these sites, no reason not to. I wish you great success. I'll see you in the Forum. Thank you.