HighEdWeb Presentation

Slide 1: opening

Welcome to Let's Get Sassy, Responsive Design with Foundation and Sass! Hopefully some of you made it to the Bootstrap presentation just before this one – awesome presentation, and super useful for comparing that framework with this one if you're unsure which direction to take. Just to let you know, this presentation will be a little technical – I'll show some code and it's best if you know some HTML and CSS. But if not, no worries, I'll explain everything and I think you'll understand just fine. If you go to the link for my presentation on the HighEd Web website, you can download my slides and follow along if it's difficult to see things up here.

And a disclaimer: I LOVE cats, and as we all know cats rule the Internet, so they were the obvious choice for my theme. And here we have this sassy cat telling us to get started, so here we go!

Slide 2: me

So a little about myself – my name is Ali Gray and I've been the web designer at Portland Community College in Portland, Oregon for just about 2 years. I got my Bachelor of Arts in Media Arts from the University of Montana in 2012 and worked for a small agency as a web designer for 2 years prior to moving to Portland and PCC, where I finally got back into higher ed which is where I wanted to be to begin with. There's my Twitter handle – feel free to contact me about anything anytime. Aaand, this is my cat Kona. I couldn't resist showing her off because seriously, look how adorable she is. And she's a little sassy, so perfect. But enough about me.

Slide 3: PCC website

This is our new responsive website, at pcc.edu! We launched it on May 4th (yep, geeky). We couldn't change the overall look, feel and content from the old website because we're waiting for a larger rebranding effort from Marketing, but it is now responsive, which was the big thing, and we started from scratch with modern technologies like HTML5, CSS3, better accessibility, and of course Foundation and Sass.. We'll have to change it again once we do get the updated branding, but using a responsive framework and more flexible CSS (Sass) will make this much easier. We'll get into that later. PCC has about 90,000 students total, including online, community ed, our small business development segment, and the credit students. We have 4 campuses and numerous centers around the Portland metro area. The website has about 8,000 pages, so it's pretty hefty, and everything is housed on this single website with mostly a single HTML/CSS template. Some is on WordPress, which runs on a version of the main template, and the rest (hopefully) will be before too long – it's still running Dreamweaver/Contribute, unfortunately. Feel free to snoop around our Foundation code and CSS if you want, but unfortunately that you can't see the Sass because it isn't included in the HTML pages.

Slide 4: Why

But why should we be building responsive websites? By now, we've all heard about the importance of making our content accessible and to provide a pleasing user experience across all devices. Mobile use is on the rise and it isn't going to stop. Maybe we've taken some first steps towards making our websites responsive using things we learned about responsive content strategy and architecture at past presentations like this one. But how do we actually build a responsive website? It's pretty daunting, and higher education websites are complex things – we have thousands of pages, dozens of content contributors, vendored applications, small web maintenance teams or small budgets. It may be difficult or impossible to set aside the time and resources to build a responsive website from scratch.

Slide 5: Responsive the fun way

But that's why tools like Foundation and Sass exist. They speed up responsive website build time, and make the experience less intimidating and actually pretty fun. We could build our own responsive framework, but why would we when great options like Foundation are already out there? And CSS works, but it can work much better. So let's get started!

Slide 6: Foundation

Building a responsive framework from scratch is yuck. It's time-consuming and just not necessary. There are too many great front-end frameworks that do all the heavy lifting for you, and all you have to do is learn some HTML structure and class names. Responsive frameworks like Foundation, which we'll be covering in this presentation, are really the nuts and bolts of your responsive website. And Foundation is built on…

Slide 7: Sass

…Sass! Plain CSS is a pain in the ass, so let's use Sass. Really, Sass is one of my favorite things. And writing plain CSS anymore makes me feel like this grouchy cat. Sass doesn't make your website responsive, but it's another tool to make building and maintaining your website quick, efficient and fun.

Slide 8: Foundation intro

Let's start with Foundation. Foundation is responsive out of the box. It gives you the HTML, CSS and JavaScript tools necessary to build your responsive website, and does most everything behind the scenes. You just install it, write a little extra code into your HTML, and voila! And you get access to all sorts of awesome responsive plugins and components that will make your life so much easier.

Slide 9: Foundation website

Here's the website, where you can download Foundation and learn how to install it (I'll only brush on that here), and more details about how to use it. It's foundation.zurb.com in case you can't read it. And yes, the fact that their mascot thing is a space yeti makes it just that much cooler.

Slide 10: Why use Foundation?

Take these two cats in the window…frame (is that a stretch?) – they're in the same frame, but they're two completely different cats. Foundation is like that – you build on top of it, and with CSS (or Sass!), you can make whatever kind of website you want. It'll be responsive, and you don't have to do everything yourself. It also has awesome responsive plugins like sliders, pagination, and navigation menus that are ready to copy and paste. It's completely modular – grab everything or only what you need. It's also fairly accessible, and is becoming better. But best of all, using Foundation (correctly) doesn't require much testing – they've already done it all.

Slide 11: Foundation good to know

Foundation is a front-end framework, like Bootstrap or Skeleton. It was developed by the company Zurb and released in 2011. It's all open-source, and is built on Sass. If you've followed mobile design trends, you've heard of mobile first, or desktop first philosophies – Foundation is mobile first. In other words, make sure everything is optimized for mobile first, then add detail as necessary for larger screens, instead of retro-fitting existing designs to work on small screens. Foundation is wonderful for building complete websites, but it's also great just for building quick HTML prototypes, like fancy wireframes. You could either use the prototype on its own, or use it as the "foundation" for your complete website.

Slide 12: Installing Foundation

Foundation can be downloaded and installed in 4 ways. (Just read through the slide.) All the methods work just fine – we used the custom install and only picked the features we wanted, but we're eventually going to re-install using the Sass method to get the custom settings files. It just depends on how much control you want versus your comfort with the product and how much you need from it.

Slide 13: The grid intro

The grid is really the heart of Foundation. If you only use one thing, use this. It's a little hard to wrap your head around first, so don't feel bad if you're confused. But the grid built on 12 columns, and you divide it up however you want. Each row has a set of divs with columns that add up to twelve. You can divide up the number of columns differently for small (phone), medium (tablet), and large (desktop) screens. In this example, you can see the row repeated three times for small, medium and large screens, with the corresponding column counts for each size. If a column won't fit on the same line, it'll just wrap to the next line, as seen in the small example. Try to make them add up to 12 though!

Slide 14: Nesting the grid – visual

Nesting the grid is where it really gets powerful. Here you can see the header, which will be 12 columns on all screen sizes (no need to specify medium or large). The content is 8 columns, and the sidebar makes up the final 4 columns of the 12. But nested inside the content div is another row with two 6-column divs, which also adds up to 12.

Slide 15: Nesting the grid – code

And here you can see the code from the visual on the previous slide. (Explain the code). It can be a little hard at first to figure out how to divide a design into rows and columns, but it gets easier the more you do it.

Slide 16: Incomplete rows and source ordering

Here are some fancier things you can do with the grid, like pushing incomplete rows to the right or centering incomplete rows. You can also visually render code in a different order than it is in the code, so in the code, .small-7 is listed first but it's displayed second.

Slide 17: PCC grid

So here’s a real-world example. I’m not going to go through the whole thing, but it gives you an example of how a page might be broken down into rows and columns. Notice that it’s pretty much just basic HTML structure with classes on the divs.

Slide 18: PCC grid code

And here’s the code for that page. You can download my presentation or take a picture of it or whatever. So the basic structure is the row, then the columns adding up to 12. Remember that nested columns must also add up to 12, even if they’re in a column less than 12, as seen in an earlier slide.

Slide 19: Templates

Are you confused? I know I was when I first started. Conveniently, Foundation offers a variety of grid templates you can use in a project or just play with to get a better feel for the grid.

Slide 20: Navigation

The other main responsive feature Foundation offers is the Top Bar. It’s a bit finicky to use and style, but is super useful. It automatically collapses to the little hamburger icon on small screens, and can include things like buttons, dropdown, search bars and stick-to-top. And Foundation offers 7 other types of responsive navigation menus for you to use.

Slide 21: Visibility

Foundation offers some really powerful built-in tools for controlling visibility and making the responsive experience better for those using screen readers. You can show and hide content based on screen size, orientation, and touch detection. You can also show and hide content for screen readers without changing the way the website looks, and create skip links for accessible navigation menus. We’re using these classes on the PCC website – the main navigation on small screens uses the Foundation Top Bar, but uses a plain HTML5 menu on large and medium. We’re using Foundation’s visibility classes to show and hide the correct content when needed, and all I had to do was add a class.

Slide 22: Media

Foundation offers a variety of methods for displaying media responsively. My favorite is the block grid, which is great for creating galleries of thumbnails or small images. You just build an unordered list with images in each li, and give it a class telling it how many images to display per row per screen size, and it does all the percentage calculations and image resizing for you! There is also a slider, lightbox, modal and responsive video handler built in. Some of these things aren’t fully accessible yet, but it seems like Foundation is really trying to improve accessibility across all their features with every new release.

Slide 23: Sass definition

So that’s a basic overview of Foundation! Using all or some of these features can really get you on your way to having a responsive website quickly and efficiently. Feel free to ask any questions you may have at the end, or hit me up on Twitter. So now let’s move on to Sass! This is a …sassy cat? … Screaming with happiness cat? I don’t know, you can decide. Anyway, Sass stands for Syntactically Awesome Style Sheets – which they are.

Slide 24: Sass website

Here’s the website, where you can learn how to install Sass (I’m not going into that here, it’s much simpler than installing Foundation), and read more details about how to use it. It’s sass-lang.com in case you can’t see it.

Slide 25: Why use Sass?

Sass is DRY (unlike this cat). DRY stands for Don’t Repeat Yourself, and that is the most important feature of Sass. CSS can be fragile and difficult to maintain – it really isn’t DRY. Color values are repeated dozens or even hundreds of times. Font stacks are repeated. Common chunks of styles like clearing floats are repeated. Your master CSS file is thousands of lines long so it can be difficult to find things. It’s easy to overwrite important things in the cascade. Yuck. Sass fixes these problems, making it clear, efficient and easier to maintain. Yes, there’s still repeating, but the computer does it, not you. Computers are good at repeating – let them do it. Also, there are cool developer tools like using Source Maps in Chrome that let you edit your Sass file live IN THE BROWSER. Magic.

Slide 26: Sass good to know

Designed by Hampton CATlin in 2006, it’s coded in Ruby (you need Ruby on your computer to run it), it’s all open-source with an active community. It extends CSS to provide features available in traditional programming languages. To avoid confusion, here are two file extensions – .sass (old) and .scss (Sassy CSS, use this one). If you don’t want to use command line, there’s a desktop app called Scout, and there’s also a handy online Sass editor called SassMeister. Sass is similar to CoffeeScript and LESS, which you may have heard of. I believe there’s a Bootstrap and LESS presentation here if you want to go to it and compare the two!

Slide 27: How Sass works

Sass is a CSS pre-processor – you write SASS, fancy things happen in the background, and it spits out CSS. Browsers can’t understand Sass, so you still attach the CSS file to your HTML documents.

Slide 28: Partials

Sass partials let you divide your code into small, logical files. Here is a sample of the file structure I’m using on the PCC site. You then import them into a single Sass file, which is compiled to a single CSS file. You attach this file to your HTML. Note the underscores – in this example, class-schedule.scss won’t compile into the same file as the others.

Slide 29: Nesting

Nesting makes the code more DRY – you don’t have to repeat .header, .header, .header…. It also follows the HTML structure, making the code easier to read and view the relationships between rules. Also, say .header became .main-header in HTML, but now you only need to change it in one place! Note the ampersand – the previous selector. &:hover nested inside the a compiles to a:hover. But nesting can get out of hand, so don’t go crazy. Too much specificity isn’t good either.

Slide 30: Variables

The. Best. Thing. Ever. Variables let you store values, which you can reuse throughout your Sass. And you only have to change them in one place. Variables are declared using the dollar sign. I'm using them for color values here, but they can be used for anything, like a border declaration, font stack… anything that is included in a single declaration. For chunks of code, we can use…

Slide 31: Mixins

…a mixin! Mixins are great for really WET things like vendor prefixes. Declare them once, then just re-use them however many times you want using the @include declaration.

Slide 32: Mixin arguments

Mixins can be used for any chunk of code, like this button declaration with many different values. This mixin takes arguments, in this case two color values and a width value. You can pass variables as arguments. There are also some built-in Sass functions, like darkening and lightening colors. The border-bottom here is darkening the $background variable by 10%.

Slide 33: Mixin arguments, any number

Mixins can also take any number of arguments, of any type, as shown here with transition. Sass extends and placeholders are similar, but a little different. I don't find them as useful and they can be a little dangerous to use, so I'm not going to go into them here.

Slide 34: Math

Sass can do math! The example here is for making columns – this is actually the logic Foundation uses (it's built on Sass), to create its responsive framework. Also notice here the inline Sass comments, which aren't compiled to your CSS. They're helpful in the Sass file and don't take up space in the final CSS file. Super handy.

Slide 35: Expressions and functions

We're not going into this here – I just thought I'd show you how far you can take Sass if you feel the need or desire. For those of you who care, Sass can get crazy, and it really is like a programming language. Here's an example of an expression and a custom function. These are for special cases, not really for day-to-day coding. Check out sass-lang.com for more.

Slide 36: Wrap up

So that was kind of a ridiculous amount of information. It may feel like you're at the bottom and there's a long way to climb, like this little kitty, but it really isn't as daunting as it may seem. Anyway, you probably have a bunch of questions, and if you don't feel like asking it here, hit me up on Twitter and I'll do my best to answer. So, any questions? Thanks everyone, and enjoy the rest of the conference!