

AQUENT

GYMNASIUM

RESPONSIVE WEB DESIGN:
BUILD A PORTFOLIO FOR ALL DEVICES

LESSON 11

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.

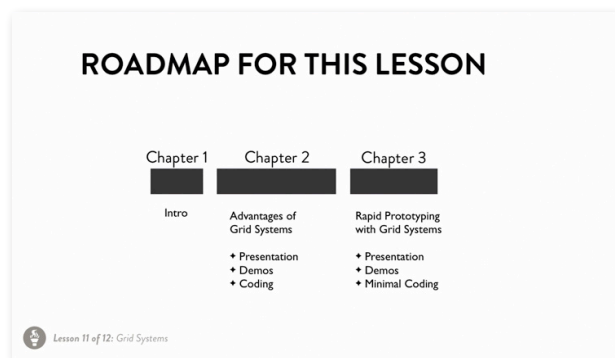
CHAPTER 1: GRID SYSTEMS OVERVIEW

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 11, Grid Systems.

As always, at the end of this lesson there will be an assignment and a brief quiz to test your knowledge and to reinforce the concepts herein, and be sure to use the pause button at any given point in order to work with the code or simply to absorb some concepts. And finally, if you have questions, be sure to hit the Forum. There we have instructors, TAs, and most importantly, your other classmates available to help you out.

Let's talk a little bit about the road map for this lesson. We're currently in Chapter 1, the intro. Chapter 2, Advantages of Grid Systems. In that chapter, we'll be going through a presentation with some demos, and you'll be doing a little bit of coding. In Chapter 3, Rapid Prototyping with Grid Systems. Here there'll be some presentations, some demos, and just a little bit of coding for you.

This is Responsive Web Design.



CHAPTER 2: ADVANTAGES OF GRID SYSTEMS

In this section, we talk about the advantages of grid systems, and by default, we'll be talking about the disadvantages. But the objective is to leverage your responsive web design skills into the ability to understand and work with any grid system or framework.

So what does this mean? Well, you've been doing an awful lot of work. You're toned. You've learned how to create fluid grids and fluid images, and you've worked with responsive typography, and you're ready to go. At the end of this lesson, we hope you have the confidence to build a custom responsive portfolio page, and we want to make sure that you can work with the number of different grid systems and frameworks out there without having to be tethered to a single one.

Now, in preparation for talking about grid systems, let's talk about why this matters. Well, when you learn how to work with grid systems and frameworks, you get bigger and better projects, bigger and better clients, and hopefully, bigger and better paychecks. And that little asterisk there is, of course, from our lawyers telling us that this is not guaranteed. However, it stands to reason that if you increase your skillset you can increase the opportunities available to you, and that's

really the point of this entire lesson.

To begin, let's just go through an overview of grid systems. In a previous lesson, we talked about the fact that there are a number of different grid systems out there in the wild, and it can be a little confusing to dig through these. And this is, again, one of the objectives of this lesson. But let's take a look at what these different systems have in common.

First of all, we have to tackle the naming. So grid systems versus frameworks, and specifically, responsive grid systems versus responsive frameworks. Let's just try to clarify what the difference is here.

So a responsive grid system typically includes the following. Templates for layout-- that includes things like rows and columns, gutters, baseline grids, multiple break points-- and some sort of mobile design. In fact, these are many of the things that you have worked with over the last 10 lessons or so.

Now, a responsive framework will take all of those things and include them in a wider set of items. For example, UI components like forms and buttons, navigation, perhaps, JavaScript alerts, such as modal windows, so there's a lot more to a framework. And we could go as far to say that all frameworks generally include some sort of grid system, but not the other way around.

Responsive Grid System

- Prebuilt CSS template for layout
- Rows & Columns
- Gutters
- Baseline Grids
- Multiple Breakpoints
- Mobile Design

Responsive Framework

- Prebuilt CSS template for layout
- Rows & Columns
- Gutters
- Baseline Grids
- Multiple Breakpoints
- Mobile Design
- UI Components (Forms, Buttons)
- Navigation
- JavaScript Alerts (Modal windows, etc.)


Now, of course, there are exceptions to this, and we'll talk about some of these. But when it comes to layout, generally we are presented with two different types of layouts-- twelve column grids versus sixteen column grids. Now, again, there are some exceptions to the rule, but for the most part, most frameworks and systems use 12 or 16 columns.

The main difference between them, as you might guess, is simply one of complexity, and we'll tackle that a little further on as well.

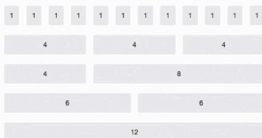
BIGGER AND BETTER PROJECTS

BIGGER AND BETTER CLIENTS

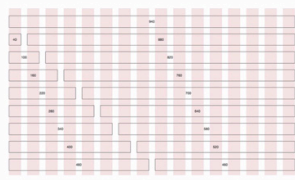
BIGGER AND BETTER PAYCHECKS*


 Lesson 11 of 12: Grid Systems

12 Column Grid



16 Column Grid



 Lesson 11 of 12: Grid Systems

Now, another way to look at it is to think about what are the things that you provide versus what does a grid system provide. So generally speaking, you provide HTML content, background colors, and background images at the minimum. The grid system gives you breakpoints and media queries and those rows and columns that we just talked about, so an actual grid structure for you to hang your content on.

But there's obviously other elements as well. And so for example, typographic styles. Well, you might provide those, or the grid system might provide those. HTML5 and CSS3 support-- well, you might be putting those in, or it might be provided for you in the system.

You Provide	Grid System Provides
HTML content	Breakpoints & media queries
Background colors	Style rules for rows and columns
Background images	
(Typographic Styles)	(Typographic Styles)
(HTML5 & CSS3 Support)	(HTML5 & CSS3 Support)
(Older browser support)	

The same thing with older browser support. You might be doing this or the grid system might.

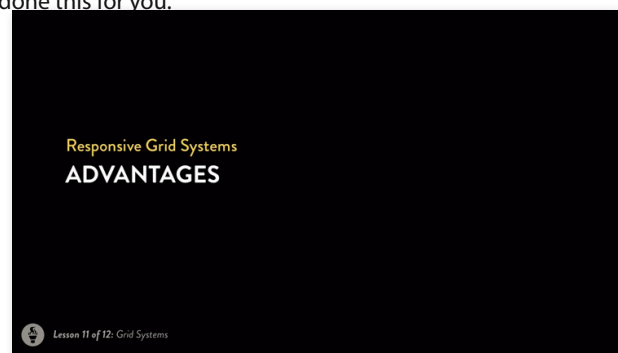
So, let's talk about some of the advantages of grid systems. They make web development faster and easier. Breakpoints are already taken care of, so all of the hard work that you have to do with a custom layout and figuring out where breakpoints need to come for mobile layouts, tablet layouts, and so forth-- generally speaking, someone has already done this for you.

Additionally, there's less need to work with the nuts and bolts of CSS layout. So things like floats and clears and positioning. All you have to do is learn the syntax of the framework. You don't necessarily have to get into the weeds of CSS.

Additionally, bugs and browser issues in older versions of IE-- all of these things, in theory, have been solved, and what this turns into is less testing. In other words, some other hopefully smart folks have done this for you, and that reduces your need to spend time doing it.

And lastly, when you've started a new project, you've already worked with the framework before, in theory. And so you can reuse the code, and you can jump right in without having to think too deeply about where you're headed.

On top of all these advantages, grid systems and frameworks are generally free, and this makes the folks who pay the bills happy. There is less development cost involved if it's done right.



So, what is not to like? Everyone loves grid systems. However, there are disadvantages, and so we need to go through these as well.

So, we'll start with this quote here. I suppose it's tempting if the only tool you have is a hammer to treat everything as if it were a nail. What does this mean? Well, responsive systems are extremely powerful. They're designed for multiple use cases. So, you can create them for huge sites, and you can create them for small sites. And what this means is you can end up using a responsive system even when you don't need it.

So our portfolio page, for example. It's a single page. It's got a very nicely crafted set of rows and columns, and we're in control and understand every component. Do we really need a grid framework for that page? Probably not.

Now, this ties in with the next concept, which is the learning curve. So many of these systems and frameworks take a while to figure out. There's a lot of code there. There is a number of supporting JavaScript files and CSS files. And you have to understand the entire concept of laying things out on someone else's grid, so this comes with a price. You have to learn the framework, and you have to commit to that framework as well.

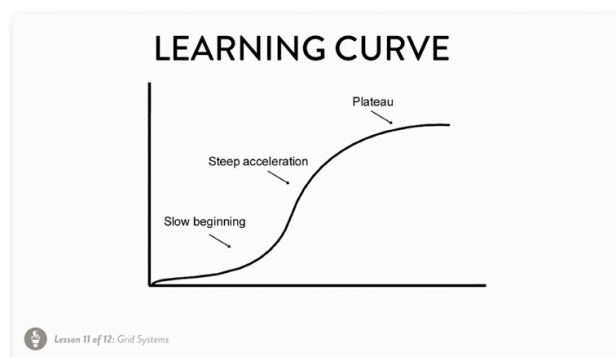
As I mentioned before, grid systems are usually designed for multiple scenarios, and what this means is you're generally including excess JavaScript and CSS code that you're not even using on your site. And again, these are there for multiple scenarios, none of which might be yours. So, you just have to keep in mind that especially with things like mobile design, we're very, very conscious about how big our site downloads are.

Now, from an aesthetic standpoint, we typically have these abstract grids that we work with within systems and frameworks. And as I mentioned before, they're typically 12 or 16 columns. However, grids are supposed to help us with visual harmony, and they're not supposed to limit our creativity.

But in some cases, grid systems tend to make everything look the same, so we have to be very careful about this. We're supposed to become more creative when we're working with grids, not less creative.

Additionally responsive systems are always changing. They're always being updated, and that just adds an extra layer of complexity that you have to pay attention to.

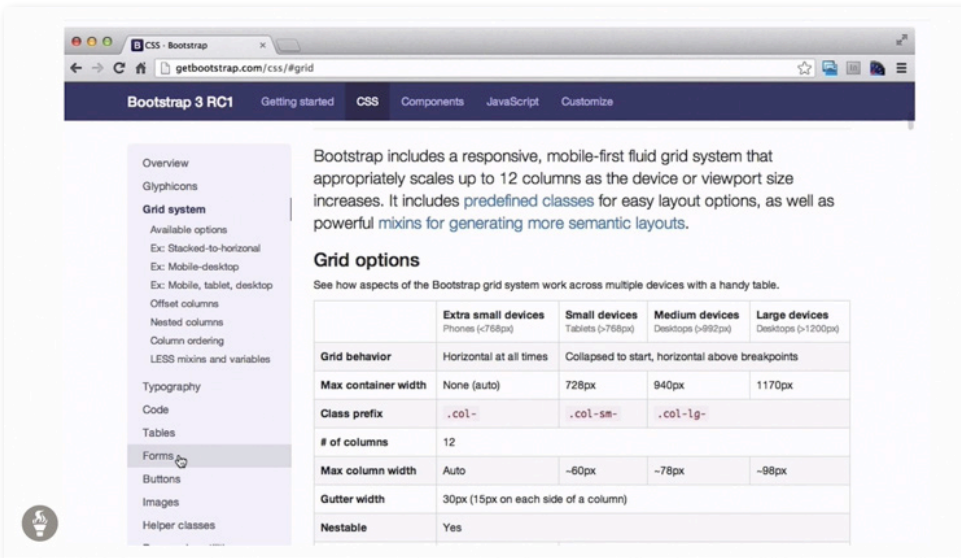
On a related note, because these are third party systems, support and documentation is an important part of it, and oftentimes, this documentation can be lacking. So you're kind of left on your own. And if you don't understand the framework, well, you could be in for a rough time trying to figure out what's going on. And this can be even as drastic as someone pulling the plug, ceasing development on a framework that you've grown to know and love as every other framework out there begins to advance.



So at this point, I figure it might be a big help to talk about a real world example, to put things in context. Let's take a quick look at Twitter Bootstrap, and I'll demonstrate some of the pros and cons.

Bootstrap is a great example of a responsive framework. This means there's a fluid grid system at its core. It's really easy to use.

In addition to this grid system-- which is based on mobile first principles, by the way-- there are additional components such as forms, buttons, built in handling of images, as well as a whole suite of UI elements, such as navigation bars and drop-down menus, and these have default behaviors and styles that you can use. All of those features, as well as excellent documentation and support, have made Bootstrap a really popular and accessible framework.



The screenshot shows the Bootstrap 3 RC1 documentation page. The left sidebar contains a navigation menu with categories like Overview, Glyphicons, Grid system, Typography, and Forms. The main content area is titled 'Grid system' and includes a description of the responsive, mobile-first fluid grid system. Below this, there is a section titled 'Grid options' which contains a table detailing the grid system's behavior across different device sizes.

	Extra small devices Phones (<768px)	Small devices Tablets (>768px)	Medium devices Desktops (>992px)	Large devices Desktops (>1200px)
Grid behavior	Horizontal at all times			
Max container width	None (auto)	728px	940px	1170px
Class prefix	.col-	.col-sm-	.col-lg-	
# of columns	12			
Max column width	Auto	~60px	~78px	~98px
Gutter width	30px (15px on each side of a column)			
Nestable	Yes			

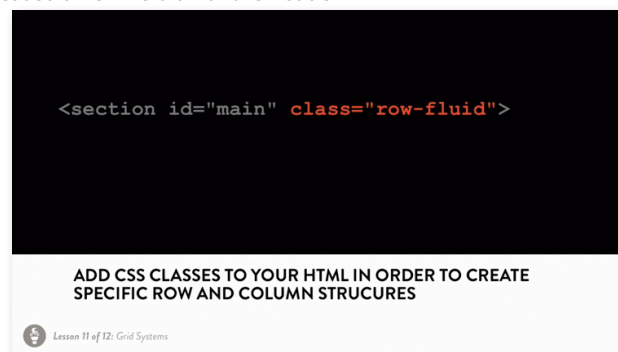
So, let's just take a quick look at how Bootstrap works. Because all grid systems essentially work in a very similar way. You have a number of pre-defined CSS classes. These come packaged up in a big old stylesheet. And once you've linked your HTML document to that stylesheet, you have access to all the code within, and you can use it for your layout.

So, the first step is to learn the class names, and then, when you add these to your HTML, you get your instant result.

So for example, here's one of Bootstrap's naming conventions. If you add class="row-fluid", that's going to use this rule here, where the width is 100%.

Now, as an example of what can happen, recently, Bootstrap released a new version of their code base. So they went up from 2.32 to a release candidate version of 3.0 at the time of this recording. In that new release, there were a number of additional features and streamlined code that were added.

So, Bootstrap has these pre-defined class names that we just talked about, and in version 2, if you wanted to create columns in your layout, you would use the following convention-- div class="span4" or div class="span8". So this special naming convention would give you a column that spanned four columns or a column that spanned 8 columns.



The screenshot shows a code editor with a dark background. The code being edited is `<section id="main" class="row-fluid">`. Below the code editor, there is a white box with the text 'ADD CSS CLASSES TO YOUR HTML IN ORDER TO CREATE SPECIFIC ROW AND COLUMN STRUCTURES'. At the bottom left of the white box, there is a small icon and the text 'Lesson 11 of 12: Grid Systems'.

Now, in version 3, if you wanted to do the same thing, well, those pre-defined class names have changed. Now you need to say `div class="col-lg-4"` or `div class="col-lg-8"`. And the new version actually is more flexible and powerful. The point here is not the details, but the point is that it's changed.

If you're a busy developer and this happens to you, you essentially have two choices. You can either stick with the old code until you have time to learn the new stuff, or you take time out of your schedule to learn the new stuff, and then you apply it to your project.

In the end, though, these are all just trade-offs. You're getting the benefit of all the great work that someone else has been doing to make your life easier. The flip side is that when a third party like Bootstrap makes a change to their framework, they're going to do it on their schedule and not yours.

In any case, in the next chapter, you're going to take a look at two other systems, and we're going to focus in a bit more on the use of responsive grid systems for rapid prototyping. This is Responsive Web Design.

CHAPTER 3: RAPID PROTOTYPING WITH GRID SYSTEMS

In this section, we'll talk a little bit about rapid prototyping with grid systems. The objective here is to talk about the way that grid systems and frameworks can speed up your design process. So to begin talking about that, we have to briefly cover rapid prototyping.

Now generally speaking, when you're designing a website it goes through a series of familiar processes. Oftentimes you start with some sort of raw mockup or maybe a sketch. And this sketch goes to a series of other folks for feedback. So maybe people on your own team or a client.

They give you that feedback. You filter that back into your design. Your design evolves, and you go through that process over and over again, depending on the size of the project. Now, where does the rapid part come in? Well, the traditional method for doing this often involves using something like Photoshop. So, Photoshop might be used to create prototypes or mockups of a web page. And we get that feedback and so forth.

Now, the problem with this is not that Photoshop is not a great tool. It is. However, it's not particularly well-suited toward responsive design. Because of course, in responsive design, we often have multiple versions of our site. So, for smartphone or tablet, the different desktop sizes, there may be up to four or more layouts. And honestly, when it comes to feedback and evolution, you can get caught into a time-consuming trap if you're stuck in the traditional prototyping workflow.

When it comes to doing prototypes, we want something that's fast, easy, and disposable. And layout gets designed right in the browser, and so that helps us figure out what things would look like. And ultimately, the holy grail is we can take that prototype code and, in some cases, convert it

Version 2

```
<div class="row">
  <div class="span4">...</div>
  <div class="span8">...</div>
</div>
```

Version 3

```
<div class="row">
  <div class="col-lg-4">...</div>
  <div class="col-lg-8">...</div>
</div>
```

Lesson 11 of 12: Grid Systems

FEEDBACK & EVOLVE

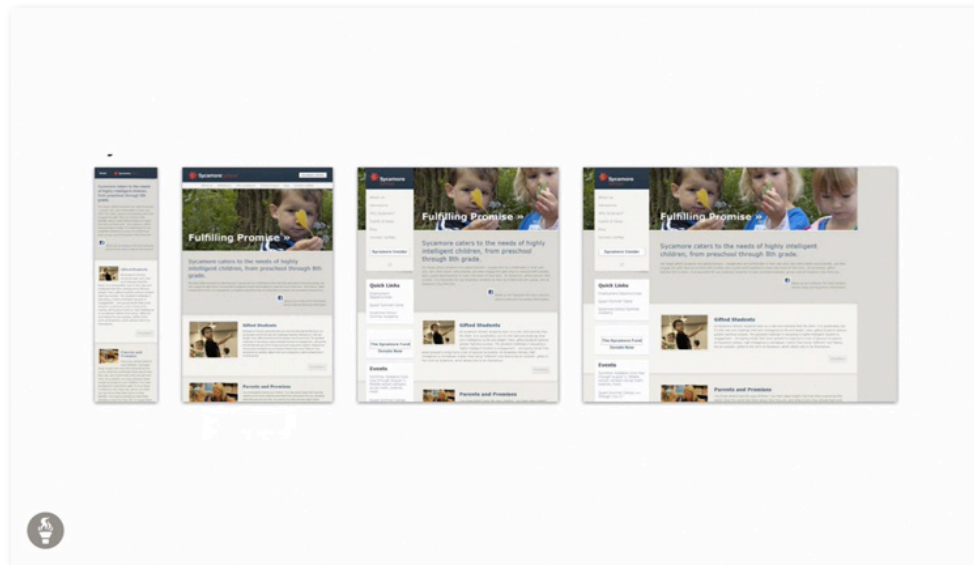


Lesson 11 of 12: Grid Systems

to production.

Now, there are a couple different options for doing this. I'll just show you a brief demonstration of one, and this is from a company called ZURB, and they have a framework called Foundation, which is very popular and very well-documented and very good. And they talk a little bit about their rapid prototyping support.

So, what you're seeing here is an example of something that has been prototyped or mocked up using some built-in assets that they include with the framework. So, these assets include things like the placeholders and the colors of the buttons and some of the components, as well as the support for the fluid grids with rows and columns.



And it even goes so far as to give you some interactive elements, so when I click on this Donate Now button, I get this little pop-up modal window here. Of course, it doesn't do anything because it's not active, but you get the idea.

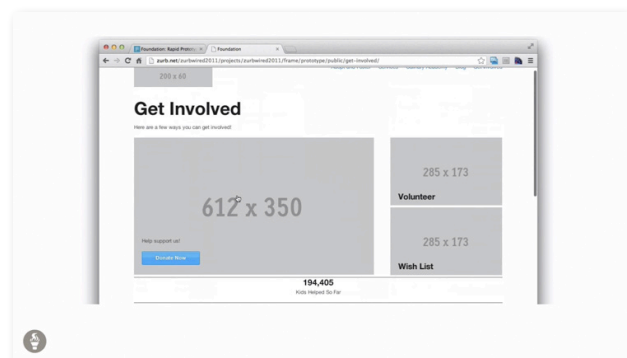
So this is very cool. This combines the benefits of the grid system, everything that we talked about in the previous section, but it gives us a little more. It gives us some assets that we can use to help us create initial prototypes and wireframes.

Now as always, we have to come back to the hammer. Do we need this for every situation? Because of course, as I mentioned, something like the ZURB prototyping assets are very powerful, but we still have to learn the whole framework to begin with. We've got to bulk up. We've got to figure out how everything works. And again, that could be an investment in time.

So is there anything lighter? Is there anything that we could use that might make the prototyping process faster? So, I'm going to show you one option for this-- a demo of a system called Gridset, and we'll take a look at how this works.

GRID SYSTEMS CAN HELP WITH THAT!

- ✦ Fast, easy and disposable
- ✦ Layout is designed natively in the browser
- ✦ Can be converted to production code in some cases



Now, there is some demo code that I'll be using. You can't necessarily use this code without setting up a Gridset account, but I'll walk through the whole steps of this, so if you want to use this file, you can feel free to do it. But of course, you're going to need to create an account, as you'll see in just a second.

So, what this page is is very simple. What I've done is I've stripped down a page even further, so I've taken away all of the base styles, except for a few font styles, and I've added these colors for the different divs here, just so we can see how they interact.

But this is a very stripped down file, and we're going to be doing some basic grid layout here using Gridset. So as I mentioned before, Gridset is a web application at gridsetapp.com. And what I've done here already is I've created a generic grid set. So I've done a little bit of work ahead of time.

What I've done is I've created two different grids, one that's mobile and one that's desktop. And what we're going to do is we're going to apply these layouts to that HTML that you just saw. A few things to keep in mind here. You'll see that there is a field for adding a class name. And this should look a little familiar. So, as I mentioned before, this is a common technique. You create a class, and then apply that class for the layout in your own HTML.

Now, this particular one has 320 pixels for our mobile layout. Of course, you can choose other options. Two columns, which if you wanted to, you could make it one. But we'll make it two, just to make it interesting, and you can also change the ratio of these two columns to Even by choosing that option and reloading.

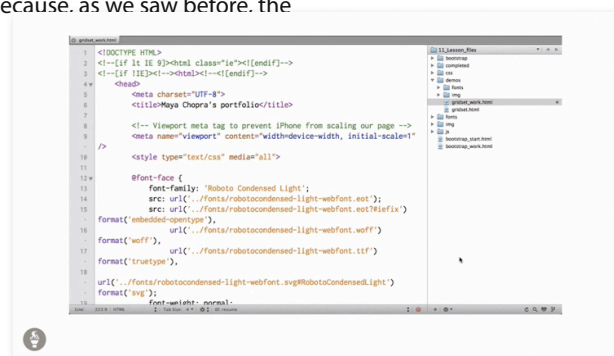
Now, the desktop version, I'm not going to walk through each detail. What I do want to show you, though, is that you can manually resize the column widths here. So, this is very cool for visually oriented folks, and you can simply change the column width to your liking. If you like it, you can save it. But let's talk about how you can use it.

We click this little Use button here, and then click on Embedded CSS Link, what we get is a absolute URL that we can copy. And then we can paste this link to an external stylesheet in our HTML, which looks like this. So that's the first thing that we can do.

An optional thing, which I also want to do just because it's cool, is we can copy this script code. And this script code will allow us to toggle off and on a grid, so we can actually see our content aligning to the grids that we create. So, let's put this into action. [It's] very similar to what we did in Bootstrap.

We're going to add a class here, and this one will be class="d1". Because, as we saw before, the prefix is d, but we're going to say class="d1-d2". So, this content will span those two columns. So d, again, is the desktop. We're going to be adding m for the mobile in just a second.

But this content here we want to be in the third column, so class="d3". Let's go ahead and save this, and oh, that doesn't look good. What's going on there? We can see that the pink has spanned two columns, and the green has spanned one, but this orange, what's it doing? Well, it's ignoring the fact that these other two exist because we didn't put in any rules for those.



So, let's go down to this `div id="content"` and say, `class="d-all"`, and this is going to fix it. This is going to make sure that that spans all three columns. But there we go. We have begun to create a very basic but flexible three-column layout.

And if we reduce the width of this, well, we haven't touched the mobile yet. Before we touch the mobile, let's just do one other thing. What if we want to tweak those columns? Well, we can go back to our code, and for the first one, instead of `class="d1-d2"`, let's just go ahead and say `class="d1"`, and let's make this one `class="d2-d3"` to span those last two columns.

We reload. We see how it interacts. Very cool. Again, all responsive. All flexible. We don't have to look at a single piece of code.

OK. Let's turn our attention to the mobile view, and we're going to go ahead and split this up into two columns. So not all mobile views have to be a single column. Let's go ahead and quickly do this by adding `m1` for that first nav section, and then `m2` for this header section. Now, we've learned our lesson well. We should also put in `m-all` for the portfolio section.

If we go ahead and look at that, we can see that it worked. We've got two columns now, and if we scroll down, we can see the beginning of our single-column layout below.

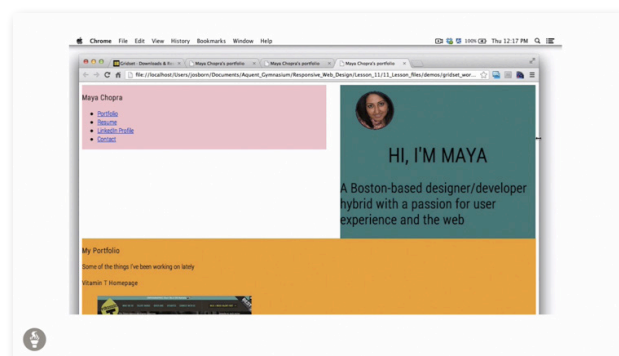
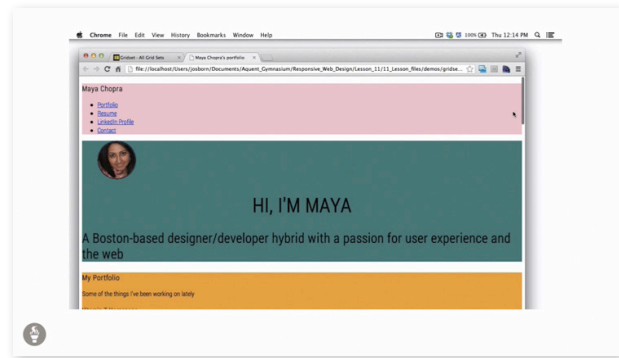
So, this is pretty nifty, and of course, as we expand the browser, it flips back to our original three column. And just like that, we have created a fluid grid layout that responsively changes based on the width of the screen.

Now, that grid overlay that I talked about earlier, you can activate that by pressing Command-G if you're on the Mac, or Control-G on the PC. What this does is trigger that JavaScript, and we can now see there's a great overlay. We can see `d2` and `d3` here, and the content aligning. In the mobile view, we can see `m1` and `m2`.

Now, one other thing that we'll take a look at here. Let's say we want to tweak one of our styles a little bit. Let's go back to the mobile style, and I didn't really like the width of those columns. So, I'm going to reduce the width of the first column-- so `m1`, and then I'm going to click Save.

Because this is an external stylesheet, if we go back to our page and reload it, and let's go down to our mobile view. Guess what? It's automatically been updated. So, this is very cool. What this means is we've got this interactive, fast way to create responsive grids.

Now, you might wonder, where is all the documentation for the `m1`'s and the `m2`'s, and `d1`'s and `d2`'s, and so forth? Well, if we go back to the Use section, we can go down and we can find the cheat sheet. So the cheat sheet actually gives us all of the class names, and it shows us exactly, visually how they will appear.



And it gives us the code for them. So we could actually integrate that code into our own styles if we wanted to. It's a really nice, really well thought out, great system for both prototyping, as well as production if you need it to.

Let's just go through a quick review of what we covered in the lesson. As we noted, grid systems share common elements. So if there are a number of things, such as twelve column or sixteen column or the ability to add class names, these things exist in virtually every system.

The other thing we talked about is learning these frameworks are going to take time and commitment, so you really have to make sure that you understand the details of these, and if you don't, you can quickly get overwhelmed and frustrated.

Additionally, don't be afraid to tinker around with your frameworks styles, but make sure that if you do this, that it's fitting in with your organization's best practices. So, if you're messing around with Bootstrap CSS, make sure that everyone knows you're messing around with it, and have a discussion as to whether that's a good idea or not. And finally, realize that you can use grid systems both for prototyping and production, as we saw.

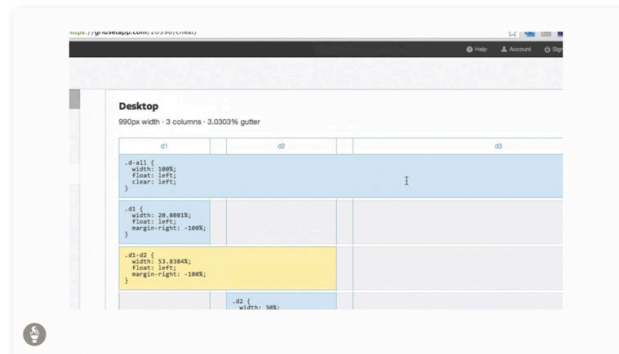
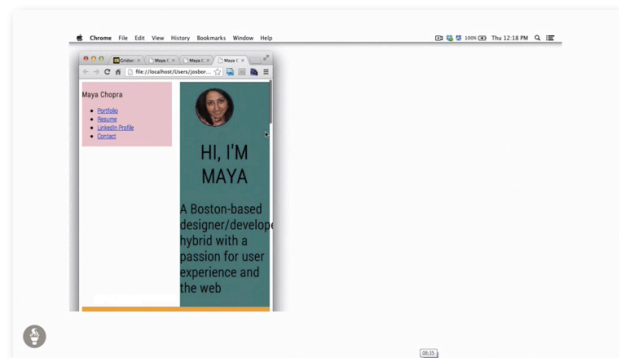
Now, one more thing to talk about. And this begins to lead into Lesson 12, the last lesson, just a little bit. If you begin to explore grid systems or frameworks just a little bit, you're going to run into CSS preprocessing. This is something we mentioned way back in Lesson 2, I believe, where we talked about preprocessors. Things such as SASS, and LESS, and Stylus.

These are all in this category of CSS preprocessing, and again, I don't want to dive too deeply into it in this lesson. Just realize that many grid systems and frameworks use these preprocessors in order to make the code simpler and to do a lot of complicated work behind the scenes.

So, we'll talk a little bit more about that in Lesson 12, Responsive Workflow. Next up, though, there is some homework. So assignment number one, as always, a short quiz in order to reinforce the concepts in the lesson and to test your knowledge.

Assignment number two, go ahead and download and explore one of the following systems and frameworks. So, the three that we looked at here, Bootstrap, Zurb Foundation, and Gridset. I would recommend looking at one of these at the very least, or if you're very adventurous and/or curious, all three.

Go ahead and play with some of the examples, or apply them to your own code just to see how it fits. That's it for now. I'll see you in the Forum, and I'll see you in the next session.



REVIEW

GRID SYSTEMS SHARE COMMON ELEMENTS

LEARNING MOST FRAMEWORKS WILL TAKE TIME AND COMMITMENT

DON'T BE AFRAID TO TINKER, BUT MAKE SURE THIS FITS YOUR ORGANIZATION'S BEST PRACTICES!

CAN BE USED FOR PROTOTYPING AND PRODUCTION

Assignment #2

DOWNLOAD AND EXPLORE ONE OF THE FOLLOWING SYSTEMS/Frameworks

**BOOTSTRAP
ZURB FOUNDATION
GRIDSET**

Lesson 11 of 12: Grid Systems