AQUENT
GYMNASIUM

RESPONSIVE WEB DESIGN:
BUILD A PORTFOLIO FOR ALL DEVICES

LESSON 9

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.

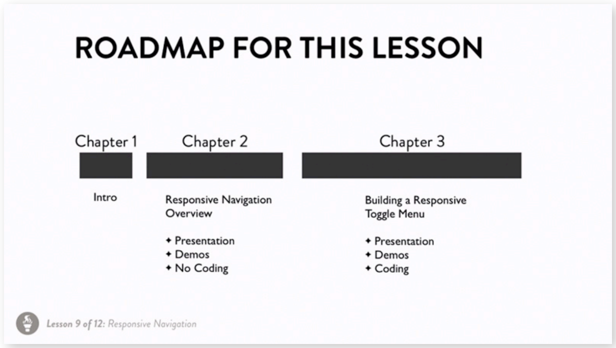
CHAPTER 1: RESPONSIVE NAVIGATION INTRODUCTION

9. RESPONSIVE NAVIGATION

Welcome to Responsive Web Design, a Gymnasium course brought to you by Aquent and Vitamin T. Responsive Web Design-- or Promote Yourself Responsively: Build a Portfolio For All Devices.

This is Lesson 9, Responsive Navigation. As always, at the end of the lesson, there will be an assignment and a brief quiz. Be sure to use the pause button at any given point in order to stop, take notes, or do the code.

And finally, if you have questions, be sure to hit the Forum. There, we'll have instructors and TAs, as well as your other classmates available to help you out.



ROADMAP FOR THIS LESSON

Chapter 1	Chapter 2	Chapter 3
Intro	Responsive Navigation Overview	Building a Responsive Toggle Menu
	• Presentation • Demos • No Coding	• Presentation • Demos • Coding

Lesson 9 of 12: Responsive Navigation

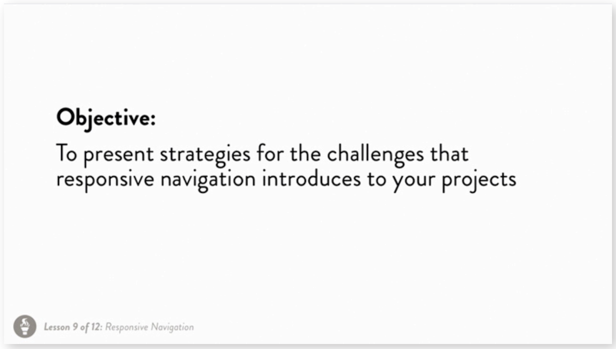
Let's talk a little bit about the roadmap for this lesson. We're currently in Chapter 1, the Intro. In Chapter 2, we'll be taking a look at an Overview of Responsive Navigation, and specifically, we'll be doing presentation demos, but no coding.

In Chapter 3, Building a Responsive Toggle Menu, there will be a little bit of presentation and demos, but mostly coding. In this case, building a responsive toggle menu.

This is Responsive Web Design.

CHAPTER 2: RESPONSIVE NAVIGATION OVERVIEW

In this section, the responsive navigation overview, I've got a few objectives. The main one is to present some strategies for the challenges that responsive navigation is going to introduce to your project, so just like the last lesson, responsive images, responsive navigation offers a number of different scenarios that you can use, and we'll talk about some of the pros and cons of each.



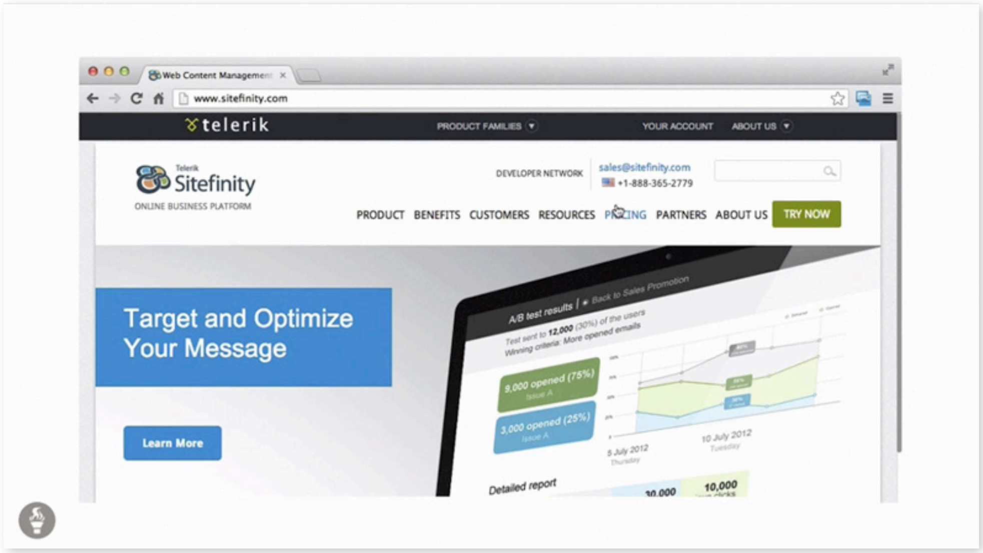
Objective:

To present strategies for the challenges that responsive navigation introduces to your projects

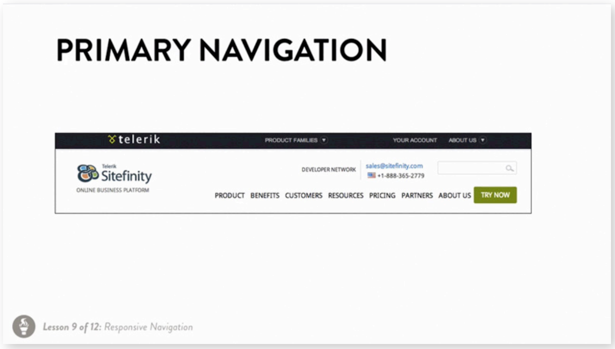
Lesson 9 of 12: Responsive Navigation

So to begin, navigation is something that designers and developers tend to struggle with, and the reason for that is it's complicated, or it can be complicated. And there are a number of reasons why we have to pay attention to navigation, but we're really concerned with responsive navigation. Now, users tend not to think of navigation until something goes wrong. And when it goes horribly wrong, well, they may not even come back to your site.

So, let's look at some of these challenges, so we can avoid this shipwreck. Responsive navigation needs to be understandable, needs to be usable, and it needs to be pleasing to the eye across a wide range of devices. So, in many ways, this is similar to our objectives for everything in responsive navigation, including layout, tables, forms, and so forth. The main difference here, of course, is that navigation is interactive. This is how users navigate your site, and it's important that we get it right.



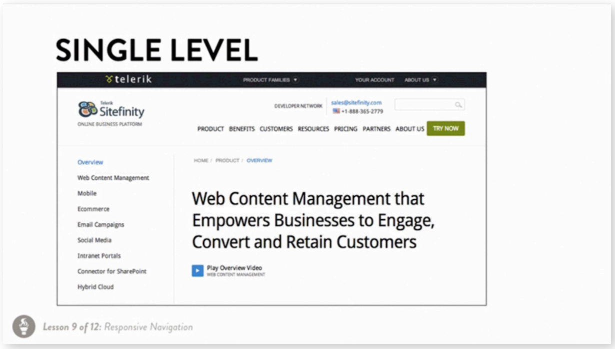
So let's just make sure that we're all on the same page and talk about some of the components of navigation. To begin, we'll take a quick tour of the Sitefinity site. So first of all, we have our primary navigation up top here, sometimes called the global navigation, and this primary navigation tends to stay the same regardless of what page you're on.


PRIMARY NAVIGATION

telerik
PRODUCT FAMILIES
YOUR ACCOUNT
ABOUT US
Sitefinity
DEVELOPER NETWORK
PRODUCT BENEFITS CUSTOMERS RESOURCES PRICING PARTNERS ABOUT US
TRY NOW
Lesson 9 of 12: Responsive Navigation

Additionally, we also have the secondary navigation at the top. Now, it so happens that this particular secondary navigation has a few bells and whistles, and we can see when we click on this link, we get a little drop down menu here, and we've got another drop down when we click on About Us, which is a little simpler.

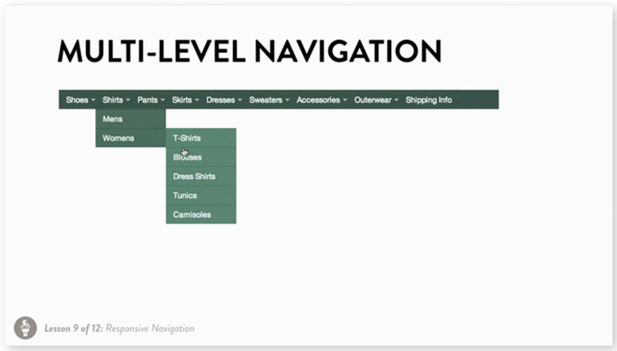
Additionally, if we click on any of these links and go to an inner page, we can see that this site uses a single-level navigation. So there's no drop down menus or anything of that sort. What we have instead is the navigation on the left-hand side here in the sidebar. We also have breadcrumbs here, which are another component of navigation, and if we click on any of the links, we can see-- again, we go to the inner page and all of the navigation remains the same.


SINGLE LEVEL

telerik
Sitefinity
PRODUCT BENEFITS CUSTOMERS RESOURCES PRICING PARTNERS ABOUT US
TRY NOW
Overview
Web Content Management
Mobile
Ecommerce
Email Campaigns
Social Media
Intranet Portals
Connector for SharePoint
Hybrid Cloud
HOME / PRODUCT / OVERVIEW
Web Content Management that Empowers Businesses to Engage, Convert and Retain Customers
Play Overview Video
Lesson 9 of 12: Responsive Navigation

So let's just break down some of the components. Again, primary navigation looks like this. Secondary navigation tends to stay on the top, and it seems to settle in the top right-hand corner on desktops. Single-level navigation, as we mentioned, has two elements-- one here and then a second on the side, and just as contrast, we'll take a look at a multi-level navigation example, and here's that drop down that I was mentioning before. So in this case, we try to conserve space by putting our navigation on one level and expand as needed.

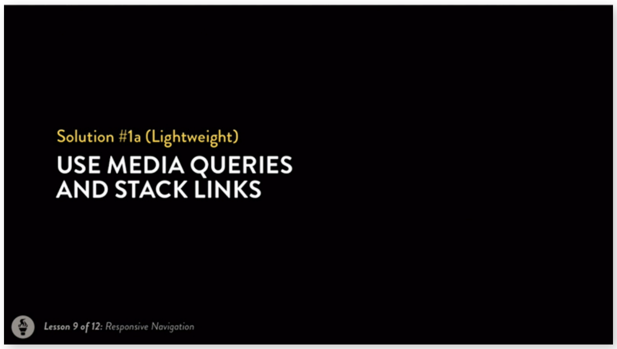
So, our fundamental issue when it comes to responsive navigation is we need it to adapt with our flexible layouts, so just as an example here, we'll take a look at our page as if we were to put in a simple navigation, and let's see what happens when we reduce the width of the browser to the Portfolio, Resume, LinkedIn Profile, and Contacts sections. So, what we'll see is that when we hit a certain point, all it does is it wraps down to the next line.


MULTI-LEVEL NAVIGATION

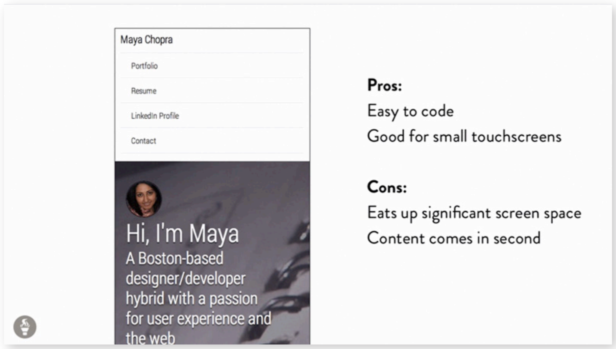
Lesson 9 of 12: Responsive Navigation

If this works for your site, go ahead and stick with it. Just keep in mind that there are some disadvantages, and we'll talk about some of those. Imagine what would happen if we had six or seven items on this page. Would it wrap just as well? And what happens to the text below it?
So there's all sorts of consequences that you should think about.

So one solution that I'll call the lightweight solution is we use media queries and we stack the links, and this is what that would look like. So, instead of having that horizontal list, we simply stack them from top to bottom, and we're done. So, the benefits of this is that it's very easy to do. It's very good for small touchscreens because we're giving it a large touch target.


Solution #1a (Lightweight)
USE MEDIA QUERIES AND STACK LINKS

Lesson 9 of 12: Responsive Navigation


Maya Chopra
Portfolio
Resume
LinkedIn Profile
Contact
Hi, I'm Maya
A Boston-based designer/developer hybrid with a passion for user experience and the web

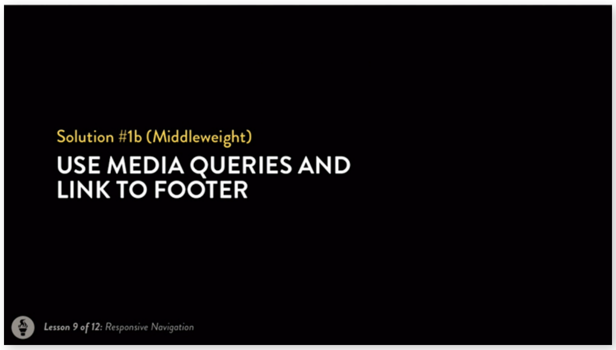
Pros:
Easy to code
Good for small touchscreens

Cons:
Eats up significant screen space
Content comes in second

The main disadvantage here is that it eats up that screen space, and especially on a smartphone, you might end up having your navigation take up the entire first view of the page, and this can be a little unfortunate. What we want to do is promote content to be first, and of course, this is part of the mobile first philosophy. So, we're going to come back to this option shortly, but let's just take a look at option 1B, the middleweight option.

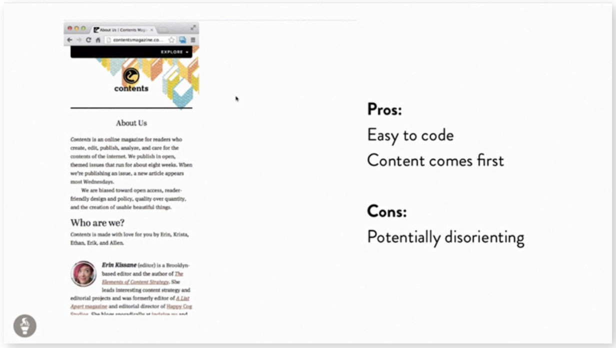
In this example, we use the media queries again, but we link to the footer text. So here is an example from contentsmagazine.com, and what we'll do is we'll resize this to a smartphone size, and notice what happened there. Our navigation turned into an Explore button on the top right corner. And so what happens when the user clicks on that? They actually jump down to the bottom of the page. Now, at the bottom of the page, they can click on any of those links and simply go to one of the pages, such as About Us.

So again, some of the advantages of this technique is that it's relatively easy to code and the content comes first because we've saved all of that screen space with the menu. Now, the main disadvantage of this technique is that it could be potentially confusing for a user to jump down to the bottom of the page. They might get used to that after a while, but first-time users may be a little disoriented when they see this technique. Still, totally valid.


Solution #1b (Middleweight)
USE MEDIA QUERIES AND LINK TO FOOTER

Lesson 9 of 12: Responsive Navigation

The next option I'll call the heavyweight, and this is a flyout menu, and you'll tend to see this in large sites. So Google, for example, implements this on their mobile site, and this is just a screenshot of the smartphone. So, when the user clicks on this menu in the top left corner, this screen will slide to the right and will expose this little shelf on the left-hand side, and we still see a peek of the right-hand side page so that we know that it's still there.



About Us

Contents is an online magazine for readers who create, edit, publish, analyze, and care for the contents of the internet. We publish in open, themed issues that run for about eight weeks. When we're publishing an issue, a new article appears most Wednesdays.

We are biased toward open access, reader-friendly design and policy, quality over quantity, and the creation of usable beautiful things.

Who are we?

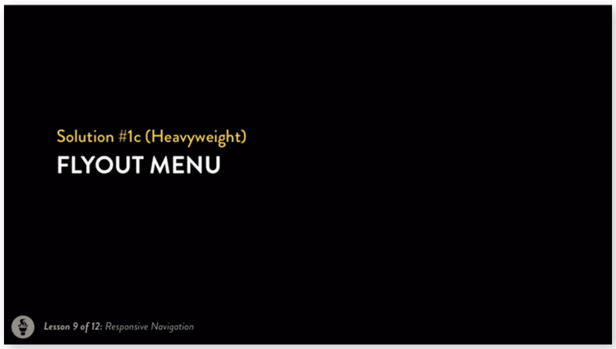
Contents is made with love for you by Erin, Krista, Ethan, Erik, and Allen.

Erin Kissane (editor) is a Brooklyn-based editor and the author of *The Elements of Content Strategy*. She leads interesting content strategy and editorial projects and was formerly editor of *A List Apart magazine* and editorial director of Happy Cog Studios. She blogs sporadically at incisive.nu and

Pros:
Easy to code
Content comes first

Cons:
Potentially disorienting

So, the benefit with this technique is we've got lots of room for extra navigation, so we can put lots of stuff in there and we can even scroll down. It looks pretty slick. So, there's an animation here that's missing that is kind of nice when we see it on the phone, and it's becoming an emerging standard. So, people are beginning to expect this because we are seeing it in large scale sites like Facebook and Google, among others.



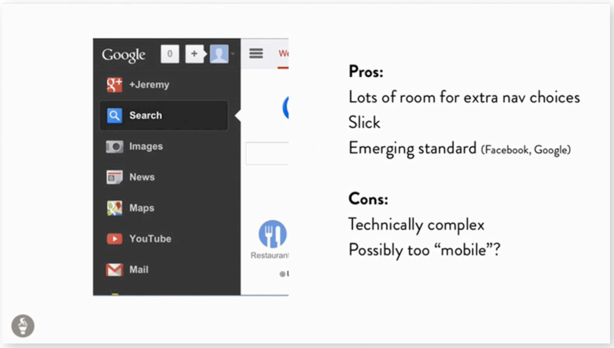
Solution #1c (Heavyweight)

FLYOUT MENU

Lesson 9 of 12: Responsive Navigation

Now, the disadvantages here is that it's technically complex to implement, so you might not need to do this on your small portfolio site. And also, it might be a little too mobile. What does that mean? Well, what I mean by that is the desktop navigation is so different from the mobile one that you might actually end up creating two separate navigations, and then you have to maintain those both.

So this is not meant to be an exhaustive survey of navigation techniques. In an earlier lesson, I talked about this site, Responsive Patterns on Brad Frost's website. And here, we can see that there's a number of options for navigation-- so single-level, multi-level, and even breadcrumbs.



Pros:
Lots of room for extra nav choices
Slick
Emerging standard (Facebook, Google)

Cons:
Technically complex
Possibly too "mobile"?

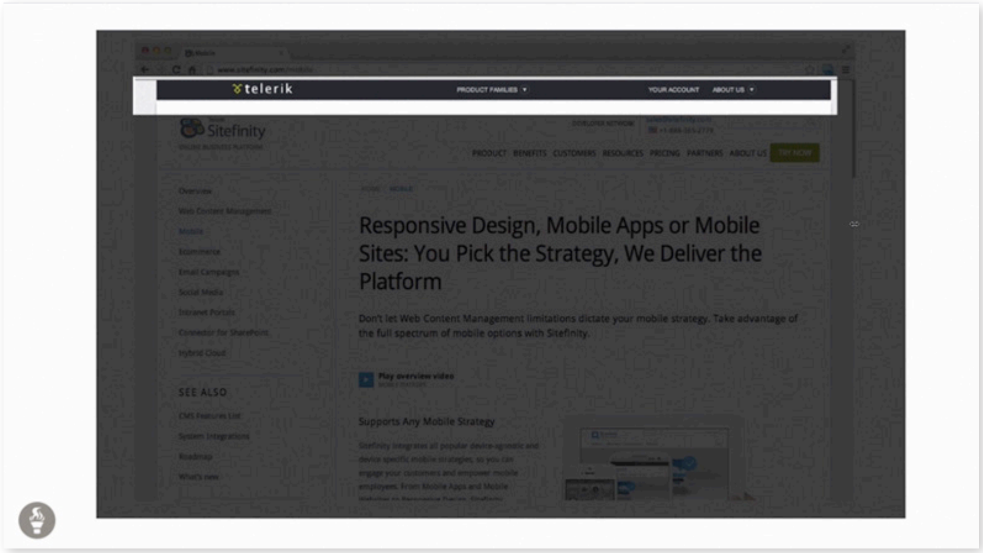
I'll just show you one here. This is for a multi-level option. And what we'll see here is that we can do a multi-level option for a smartphone. However, it's going to take up a lot of space. But if we need to do it, we can.

The other thing to keep in mind here is that on large scale sites, we're not limited to just one technique. So if we go back to the Sitefinity site that I was showing earlier, let's just take a look at what happens to the secondary navigation, and then we'll take a look at the primary navigation. So, for the secondary navigation, we have three elements on the widest desktop screen. As I begin to resize the browser, it goes down to two. And then as we go down to the smallest layout, we get down to one. And in many ways, this is adhering to the mobile first philosophy. Clearly, the designers of this site felt that your account is the most important navigation item to have on the smallest screen.

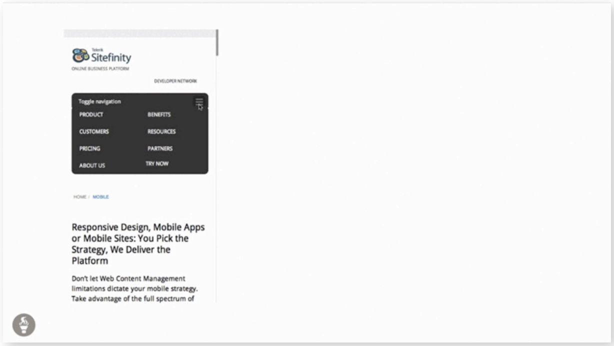
Now, there were some other things happening as well down below. I'm going to switch gears, and I'm going to show you what this looks like without the secondary navigation. So here, as we begin to move down, at a certain breakpoint, we see that the primary navigation turns into a toggle navigation bar. In fact, the entire design for this has changed.

And so again, we'll go down to the smallest view here, and let's just see what happens when we click on Toggle Navigation. When we do that, we're going to expose the primary navigation but collapse that secondary navigation. So again, this is one particular choice that these designers and developers have made. Just keep it in mind that you can mix and match these as needed.

Now, let's look at another problem-- navigation that has to work both on the desktop as well as touchscreens. Now, this example is not meant to pick on Bootstrap because Bootstrap is awesome, but this is the Bootstrap 960 pixel framework put onto a smartphone. So it's not scaling, it's not responsive, and we can imagine if we try to click on any of those navigation choices, we're going to be really frustrated. We either have to pinch and zoom, or we have to realize that we're probably going to click on the wrong link and be really annoyed, especially if we've got large fingers like our friend here.


telerik
PRODUCT FAMILIES
YOUR ACCOUNT
ABOUT US
Sitefinity
PRODUCT BENEFITS CUSTOMERS RESOURCES PRICING PARTNERS ABOUT US TRY NOW
Responsive Design, Mobile Apps or Mobile Sites: You Pick the Strategy, We Deliver the Platform
Don't let Web Content Management limitations dictate your mobile strategy. Take advantage of the full spectrum of mobile options with Sitefinity.
Play overview video
Supports Any Mobile Strategy
SEE ALSO

A quote by Luke Wroblewski: "Most navigation systems today are designed for a mouse and keyboard world," and this is true. We still are coming to terms with the fact that touch is as important, if not more important, for many users. Now, there have been studies done on the ideal touch target size. Now, these examples here are obviously not to scale, but 7-9 millimeters is around the ideal sweet spot for touch targets, and if you have two touch targets next to each other, the spacing between them is 2 millimeters.


Sitefinity
DEVELOPER NETWORK
Toggle navigation
PRODUCT
BENEFITS
CUSTOMERS
RESOURCES
PRICING
PARTNERS
ABOUT US
TRY NOW
Responsive Design, Mobile Apps or Mobile Sites: You Pick the Strategy, We Deliver the Platform
Don't let Web Content Management limitations dictate your mobile strategy. Take advantage of the full spectrum of

So, why are we using millimeters here and not pixels? Screens these days have different pixel densities. So we've got our Retina, or high-definition displays versus our standard displays, and so we can't just use width and height for pixels because that will actually differ. So, this is the physical unit that we're talking about, seven to nine millimeters. You could even use points-- so Apple does that. I think they have 44 by 44 points, because again, this is a physical measure of the screen and not one that's relative, such as pixels.

Problem #2

NAVIGATION NEEDS TO WORK ON DESKTOP AS WELL AS TOUCH DEVICES

Lesson 9 of 12: Responsive Navigation

If you need to learn a little bit more about some of these guidelines, a good place is the iOS Human Interface Guidelines at Apple, and there's the website there. In the next section, you'll get a chance to explore one technique of responsive navigation, and so get ready for that. Fire up your text editors, and we'll get started.

CHAPTER 3: BUILD A RESPONSIVE TOGGLE MENU

Luke Wroblewski

" TODAY, MOST WEB NAVIGATION SYSTEMS ARE DESIGNED FOR A MOUSE AND KEYBOARD WORLD.

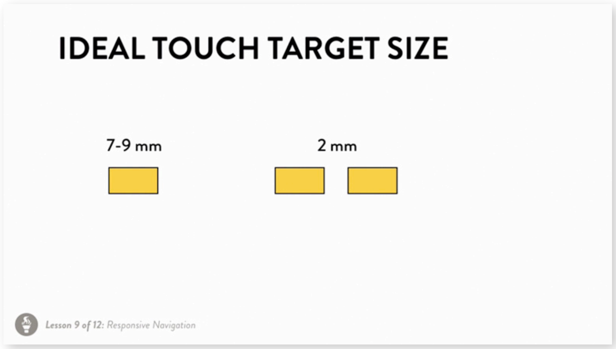
HTTP://WWW.LUKEW.COM/FF/ENTRY.ASP?1649

Lesson 9 of 12: Responsive Navigation

In this section, you'll be building a responsive toggle menu. Specifically, here's what you'll be doing. You'll be adding some markup and style for a single column, mobile layout. Additionally, you'll be adding a drop down menu, as well as a navicon using something called the Advanced Checkbox Hack, and finally, you'll be creating a horizontal menu for your large screen with media queries.

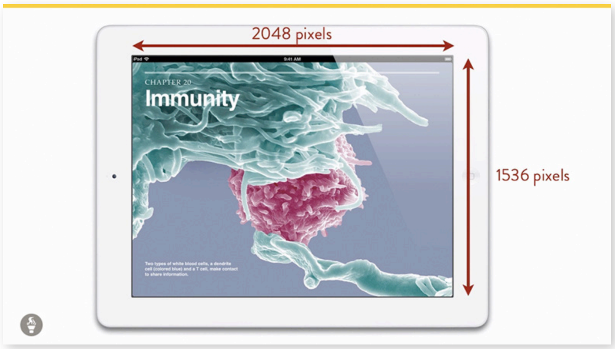
To begin with, as always, you're going to need to open up the file, portfolio_start.html, in your lesson files. Go ahead and rename this portfolio_work.html. This creates a backup.

Just a few things that I've done here between lessons. Inside the Style section, I've removed all of the previous styles, and I put them into style.css. We also have this file code_to_copy.txt, and inside this document, we've got some code that we'll be using, so you don't have to type.



IDEAL TOUCH TARGET SIZE

Lesson 9 of 12: Responsive Navigation

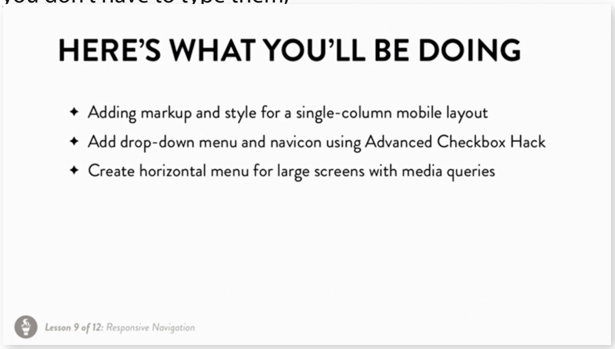


But first, before we do that, let's go ahead and take a look at this page in the browser just so we know what to expect. Now, we need navigation to be present in all views, obviously. But on the desktop, we're going to have a horizontal list as our navigation, and then in the smaller smartphone view, we're going to be using a single column, and you actually saw an example of that in Chapter 2.

The first thing you need to do is locate this whole nav section and remove the comments that were put on here earlier, and this exposes that list. And if we take a look at the blank, unstyled list, it looks pretty straightforward.

Let's go ahead and go to the code_to_copy.txt file, and what I want you to do is to copy this entire nav section. This is the exact same nav section as the one we just saw. The only difference is I've put in the hyperlinks to the different parts of the page for you, so you don't have to type them, but let's go ahead and copy that whole block and replace this one entirely.

We can see that there are links to a Portfolio, Resume, LinkedIn, and Contact section, although the LinkedIn one isn't actually active yet. Let's reload our page and make sure those work, and so the Portfolio section goes there. Resume section goes there. LinkedIn, as I mentioned, is not active.



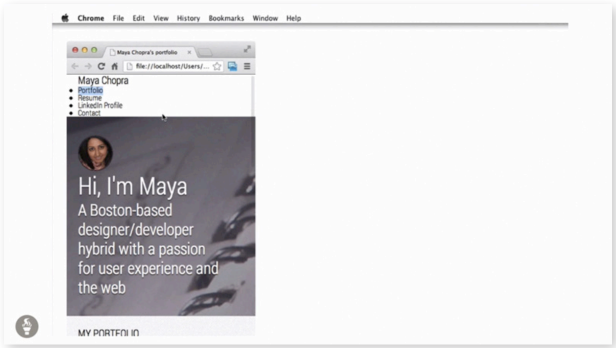
HERE'S WHAT YOU'LL BE DOING

- Adding markup and style for a single-column mobile layout
- Add drop-down menu and navicon using Advanced Checkbox Hack
- Create horizontal menu for large screens with media queries

Lesson 9 of 12: Responsive Navigation

Contact is not active, and the reason for that is because we added the Contact section in an earlier lesson and we didn't update that. So, let me just show you how to do this. We'll go ahead and we'll locate the form located at the bottom of the page, and we just need to add an id-- form id="contact", and we're all set. This will now work.

To get a better sense of what we're aiming for, here's a screenshot of our target style. So in the upper right-hand corner, we have this small navigation icon, and when we click on it, we want it to expand, and we want our list to appear. And when the user clicks on that again, we're going to have it collapse-- so there's the toggle.

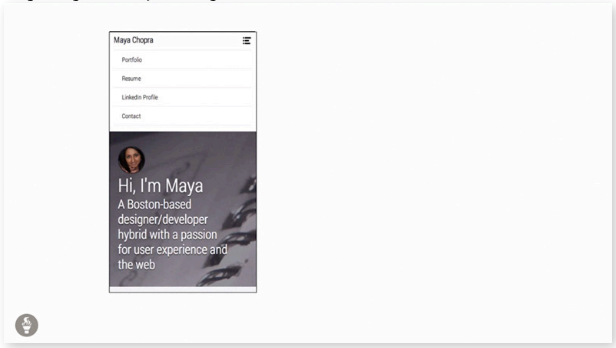

Chrome File Edit View History Bookmarks Window Help
Maya Chopra's portfolio
file:///localhost/Users/...
Maya Chopra
Portfolio
Resume
LinkedIn Profile
Contact
Hi, I'm Maya
A Boston-based designer/developer hybrid with a passion for user experience and the web
MY PORTFOLIO

So obviously, before we do anything, we've got to create that single-column layout for the navigation, and let's go ahead and do that. Everything we need to do is within this section-- element, as well as this nav element here. But let's go ahead and steal some styles from our code_to_copy.txt.

First of all, we'll steal this nav style with some padding, and we'll put it inside of our internal stylesheet-- so padding of 0.65em for the top and bottom. Let's go down and take the second style, nav h3, and what this will do is we'll float the nav to the left and we'll see that that list will then fall into place, although it's going to look a little broken.

But let's just see what this looks like. So, we'll open this up and ooh, not looking so good. But this is all intentional. So, we need this to float so that we can then begin to stack these.

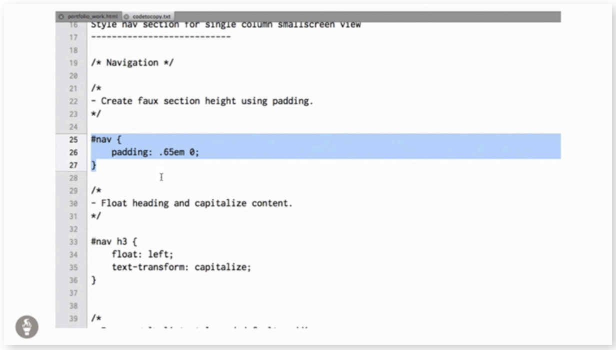
Now there's quite a lot happening in these styles, so I'll just quickly go through it. We're removing those bullet points. That's going to make it look a little better. We're going to be putting borders below each list item, as well as on the whole list itself, and we're going to be adding a little margin to add some space.


Maya Chopra
Portfolio
Resume
LinkedIn Profile
Contact
Hi, I'm Maya
A Boston-based designer/developer hybrid with a passion for user experience and the web

So let's go ahead and copy those two styles. Let's put them here in our stylesheet, and again, let's just take a quick look at what this is going to do. Here we go. We're getting the borders between each item.

Next up, let's go ahead and take this whole section-- and this is the 'nav a' style rule. This is going to do the bulk of the work. We're creating a display of block for each hyperlink in here, and we're adding some padding to the top and to the right.

At the same time, we're making sure that the font size matches our typographic style that we've worked with in previous lessons, and let's take a look at what this does. So let's copy that style, paste it here below our nav list rule, save our document.

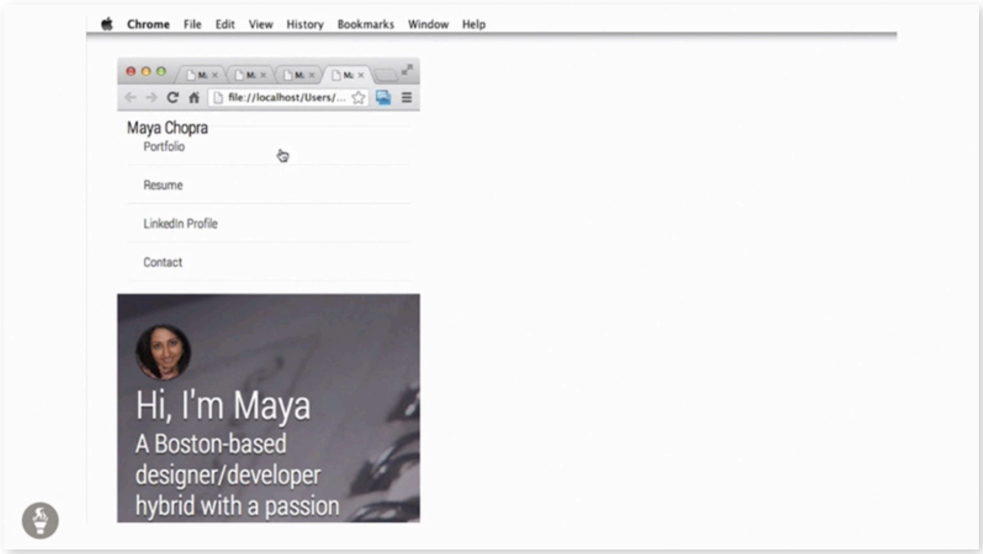


```
Style nav section for single column smallscreen view
---------------------------

/* Navigation */

/*
- Create faux section height using padding.
*/

#nav {
    padding: .65em 0;
}

/*
- Float heading and capitalize content.
*/

#nav h3 {
    float: left;
    text-transform: capitalize;
}


/*
```

And let's open this up in a new tab so we can compare the old one versus the new one-- so there we go. We're getting pretty darn close. So, this is that stacked effect we're looking for. If you're taking a look at the Maya Chopra name and it looks a little misplaced or a little strange, don't worry. We're going to be putting a navigation icon in there that will push that whole list down.

OK. There's one more style here. This is just for the #nav a:hover, and we can just copy and paste this and put it into our other styles. We don't need to preview this one. All this will do is trigger a hover effect when the user places their cursor over the items.

Now the last piece here is the bridge to the toggle menu. So this style here, #nav nav, gives us a height zero and overflow hidden. If you need to learn a little bit more about the technical details, there's a URL up above at 37signals.com, but to make a long story short, this is a technique that we're using to collapse or to hide that navigation section so that later on we can turn it on with the toggle.


Chrome File Edit View History Bookmarks Window Help
file://localhost/Users/...
Maya Chopra
Portfolio
Resume
LinkedIn Profile
Contact
Hi, I'm Maya
A Boston-based designer/developer hybrid with a passion

So, we need to take that code and put it into our style rules. Let's just go ahead and save the document. Double-check that this indeed will disappear when we come back, and sure enough, it's disappeared. So there's the previous. Here's the new.

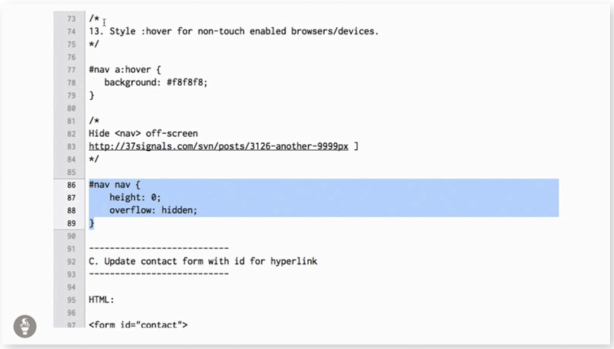
So, next we need some technique to expand or to toggle that navigation section open, and to do that, we're going to turn to a time-honored technique, one that does not necessarily apply just to web design, and it's called a hack. So a hack is essentially when you use a specific component or technique for something it wasn't intended to do, and that's exactly what we're going to be doing with this Advanced Checkbox Hack.

And here's how it works. It uses a label and checkbox input, which you most typically find with forms. What we're going to do with this, however, is we're going to use the property of the checkbox to trigger this other element, in this case the div that holds the menu. So when the user clicks on the checkbox, that will expand or toggle open the menu, and when they click on it again, it will collapse it.

Again, not really what this was intended for, but it's pretty reliable. And it's been tested on a number of platforms, and we'll use it until something better comes along.

Like most hacks, some of the logic ends up being a little twisted when you follow it. What we're doing here is we're going to be hiding the checkbox entirely using CSS. When we click on the label, that will toggle it off and on. And we're going to style that div based on the checked state of the input.

And just one other thing before we get into the code for this: the code will also use some CSS selectors that you may not be familiar with. This is using a combinator. Here you can see 'p span', and that little squiggle between the 'p' and the 'span' is the CSS selector we're using.



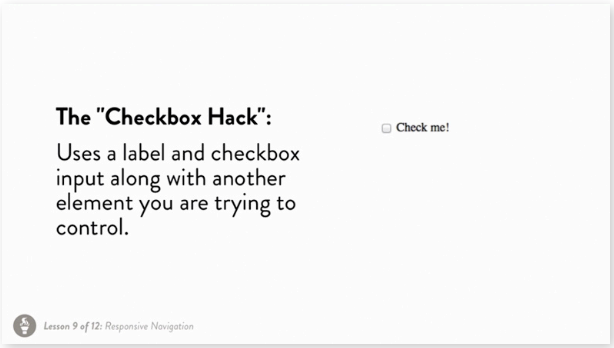
```
/*
13. Style :hover for non-touch enabled browsers/devices.
*/

#nav a:hover {
   background: #f8f8f8;
}

/*
Hide <nav> off-screen
http://37signals.com/svn/posts/3126-another-9999px ]
*/

#nav nav {
    height: 0;
    overflow: hidden;
}

---------------------------
C. Update contact form with id for hyperlink
---------------------------

HTML:

<form id="contact">
```

The way this works is it will match the second element, in this case the span, only if it's preceded by the first, which in this case is the paragraph. So as an example, you can see the code here on the right. There are two spans. The first one does not get targeted. The second one does get targeted, because it comes after a p.

So our first step is to add a little bit of HTML markup. So we want to grab all of this code here, and we're going to put it inside of the navigation section, so right above that nav tag.

If we save our document and we just see what this does, yes, it adds a checkbox right there underneath the letter m. However, that's going to disappear shortly, but it does exist.

Now comes the real hacky stuff. So, this is called Advanced Checkbox Hack, because originally there was something called the Checkbox Hack. But that technique was known to fail somewhat in iOS and Android in certain versions. So the really hacky part is here, these webkit-animation styles. Essentially, what these do is fake out Android and iOS browsers into thinking that something is happening.



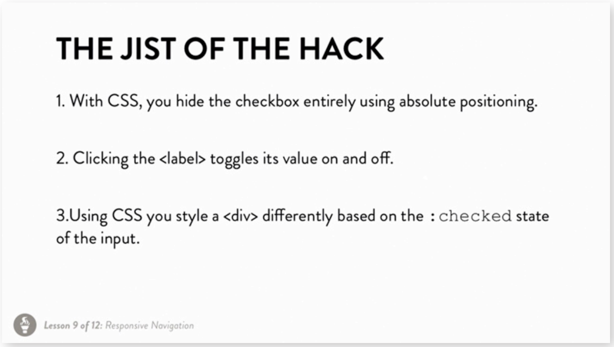
The "Checkbox Hack":

Uses a label and checkbox input along with another element you are trying to control.

Lesson 9 of 12: Responsive Navigation

If you want to read more about this, there's a link above for the Advanced Checkbox Hack and you can look at all the gory details, but we have to use it. So, let's go ahead and copy that code, and again, we're going to put it inside of our internal stylesheet.

Let's go ahead and take this code, and this code is going to hide the navigation off the screen. So, this uses a technique of absolute positioning, and it positions this to the top and to the left 999ems, negatively, so way above the top left corner of your browser. This checkbox will never ever be seen.



THE JIST OF THE HACK

1. With CSS, you hide the checkbox entirely using absolute positioning.
2. Clicking the <label> toggles its value on and off.
3. Using CSS you style a <div> differently based on the `:checked` state of the input.

Lesson 9 of 12: Responsive Navigation

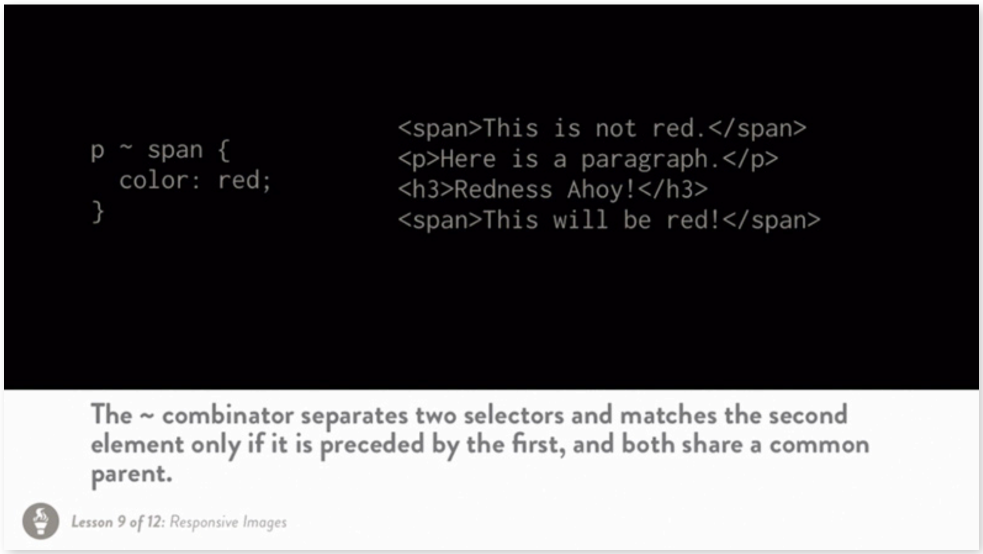
So let's go ahead and do that. We can test it. Sure enough, our checkbox is gone. So that's good.

Let's continue. This section of code does quite a bit. What we're doing here is we're creating a style for the label inside of the nav section.

And this is where we're putting in the background image of the icon menu. So at this point, we're going to get a graphic in the top right-hand corner. And we're also doing a few other things here, making sure that the cursor acts naturally as well as a few other styles for WebKit, Mozilla, and Microsoft browsers.

Let's go ahead and take that whole chunk of code, and let's paste that here. Let's save our document, and make sure that that graphic showed up. Sure enough, it showed up. However, if we click on it, nothing-- nothing will happen, so we need to add just a little bit more to make everything come together.

This is the last piece of code here, and this is the combinator selector that I mentioned earlier. So that top line of syntax looks a little wacky if you're not used to it. But essentially, we're saying, let's target the checkbox inside the navigation section when it's checked and go ahead and style the nav-- and there's our little squiggly there telling us that this is what we want to do.



```
p ~ span {
  color: red;
}
```

```
<span>This is not red.</span>
<p>Here is a paragraph.</p>
<h3>Redness Ahoy!</h3>
<span>This will be red!</span>
```

The ~ combinator separates two selectors and matches the second element only if it is preceded by the first, and both share a common parent.

Lesson 9 of 12: Responsive Images

And what do we want it to do? Well, we want it to be a height of auto and overflow hidden. That probably looks familiar. It's the same styles that we saw earlier, and essentially, this will allow us to toggle that entire navigation section off and on.

So let's put that code in here, save our document, and let's double-check and make sure that this works. Sure enough, we're in good shape.

When we click on that, it expands. When we click on it again, it will collapse. We've got our nice hover styles there. Boom. We're done.

We've got a very nice responsive menu. It gets out of our way until we need it, and users are now somewhat familiar with that icon, so they know to click on that if they need to navigate.

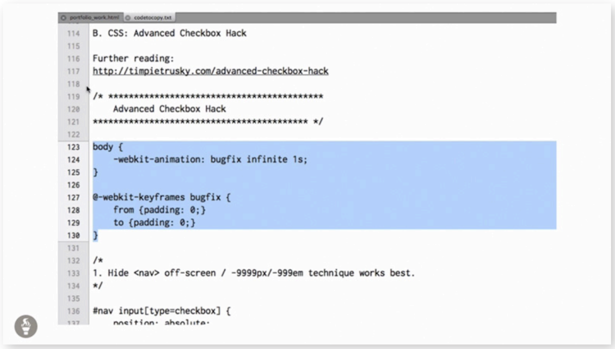
OK. So now let's turn our attention to the wide-screen navigation, or the desktop navigation. This is our target. This is what we're looking for in that size screen. So we're trying to create a basic, simple, horizontal list.

Right now, this is what it looks like if we expand our browser window. So we get the mobile navigation that stretches all the way to the edge of the screen, and it's going to look really weird and cheesy. So we don't want that.

We're going to need to use media queries. Let's go back to our code_to_copy.txt, and let's go ahead and take this whole style here called @media screen and (min-width: 45em).

And we're actually going to paste this in our internal stylesheet for now. Again, this is something we've been doing off and on throughout these lessons. Eventually we're going to want to put these media queries in one place, the external stylesheet. This is just an easier way to do it for this lesson.

So, this particular style doesn't do a whole lot. It basically takes the padding and sets it to zero for the navigation, but let's just go ahead and save our document. And let's just make sure that this is doing something, so that the media queries are working.



portfolio_work.html codetocopy.txt

```
B. CSS: Advanced Checkbox Hack

Further reading:
http://timpietrusky.com/advanced-checkbox-hack

/* ****************************************
    Advanced Checkbox Hack
**************************************** */

body {
    -webkit-animation: bugfix infinite 1s;
}

@-webkit-keyframes bugfix {
    from {padding: 0;}
    to {padding: 0;}
}

/*
1. Hide <nav> off-screen / -9999px/-999em technique works best.
*/

#nav input[type=checkbox] {
    position: absolute;
```

And again, if we toggle back and forth, very slight change, but this tells us that it's active. So now we'll get to the meat of it.

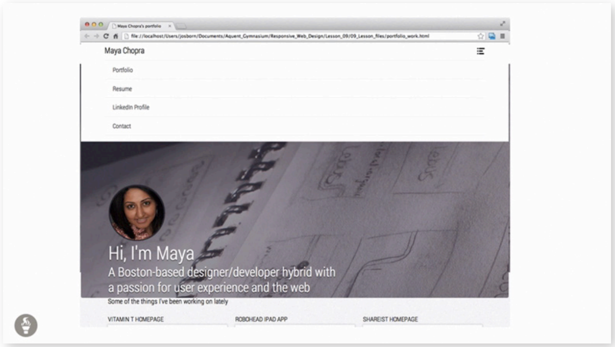
Let's go down here, and we're going to take this style, nav h3. It gives us a margin of 0.39em on the top and 1.3em on the right. Let's go ahead and paste that in.

And we don't need to preview this quite yet. Let's go back to our code. Let's take the '#nav label' style, and what this does is it sets it to 'display: none'. So this is going to be removing that navigation menu from the wide screen. We don't really need it.

Let's just test and make sure this works. So save your document, and there we go. We've just collapsed that entire section. And of course, we have no menu icon anymore.

So now, let's put in this code to make sure that the nav comes back on the screen. So here we'll put in '#nav nav', 'height: auto', 'overflow: visible'. If we reload our page, we get that navigation back up, so now it's live again.

Of course, it's still looking stacked, and we're going to have to change that to horizontal. So, a few other little details here-- we're going to take this '#nav ul' to remove the borders and the margin. And we're going to take this nav list (#nav li) item, which is going to display those list items as inline and also remove the borders for those. So those styles need to go in.



Maya Chopra's portfolio

file://localhost/Users/josborn/Documents/Aquent_Gymnasium/Responsive_Web_Design/Lesson_09/09_Lesson_files/portfolio_work.html

Maya Chopra

Portfolio

Resume

LinkedIn Profile

Contact

Hi, I'm Maya

A Boston-based designer/developer hybrid with a passion for user experience and the web

Some of the things I've been working on lately

VITAMIN T HOMEPAGE

ROBOHEAD IPAD APP

SHAREIST HOMEPAGE

We can preview this, and we're almost there. And again, the secret sauce here is going to be the last little style that's going to convert this from a vertical list to a horizontal one. And this is the following code, 'nav a', 'display: inline-block'.

So this is going to pull it all together. And it gives us a horizontal menu, like so. We've got our nice hover effect here as well.

Let's just make sure that our responsive navigation is there, and sure enough, it is. It's still working. We're in great shape.

Now there's one last little detail that has nothing to do with navigation that I'd like to do. If you see this space here above Maya's picture, there's some extra padding here that I just don't like. You will have to open up your external stylesheet, so go to style.css.

And inside style.css, you're going to need to find the #container-header section. In our stylesheet it's around line 530, and go ahead and change the padding from 10% to-- let's try half, 5%. Make sure to choose File, Save All.

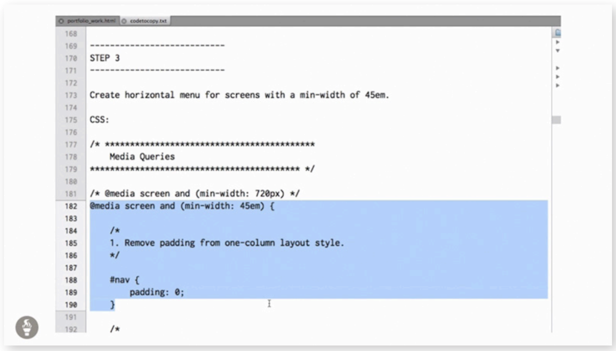
And let's just take a look at this inside of our browser. So again, there's a difference. The first one had extra padding that didn't work. The second one has less padding. If we wanted to, we could target this in other styles. But it was mainly bothering me in my desktop style.

So that's it. We are rocking out. In fact, this is your chance to par-tay, because the truth is, you have a fully styled, responsive page with images and navigation and layout, all sorts of cool things going on. So we still have a few things left to cover.

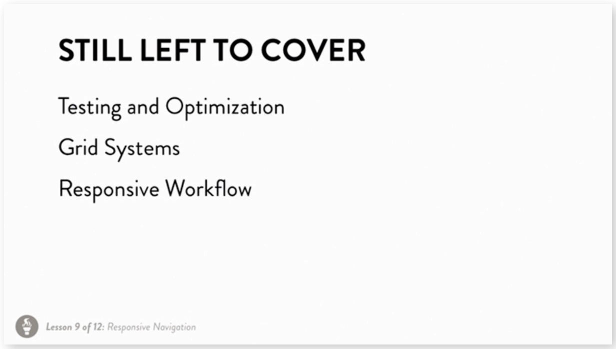
You can pat yourself on the back, just realize that there's still three more lessons here, Testing and Optimization, where we talk about making sure your responsive site works as best as possible; Grid Systems, where we talk about grid frameworks and when you might use them versus the techniques that we've been doing; and finally Responsive Workflow, where we talk about the various ways that responsive designers and developers are working today and how you're going to fit in with your new-found responsive knowledge.

However, before we get there, we still have homework. As always there's a short quiz. Assignment number 2, add a responsive navigation to your own portfolio page. Use the Advanced Checkbox Hack. It's the easiest one to do for now. If you're really adventurous, I suppose you could try some of the others.

But if you have any questions at all, be sure to ask on the Forum. So speaking of the Forum, I hope to see you there, and I also hope to see you in the next session.



```
---------------------------
STEP 3
---------------------------

Create horizontal menu for screens with a min-width of 45em.

CSS:

/* ******************************************
    Media Queries
****************************************** */

/* @media screen and (min-width: 720px) */
@media screen and (min-width: 45em) {

    /*
    1. Remove padding from one-column layout style.
    */

    #nav {
        padding: 0;
    }

    /*
```



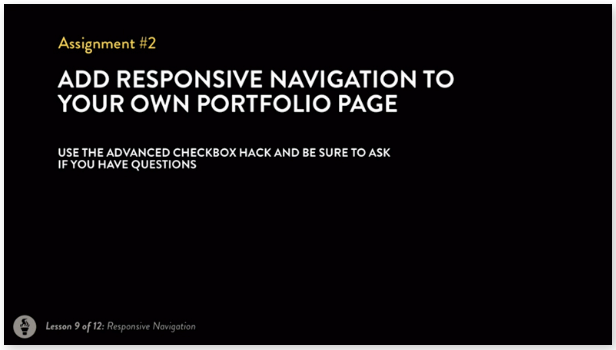
STILL LEFT TO COVER

Testing and Optimization

Grid Systems

Responsive Workflow

Lesson 9 of 12: Responsive Navigation



Assignment #2

ADD RESPONSIVE NAVIGATION TO YOUR OWN PORTFOLIO PAGE

USE THE ADVANCED CHECKBOX HACK AND BE SURE TO ASK IF YOU HAVE QUESTIONS

Lesson 9 of 12: Responsive Navigation