GYMNASIUM

RESPONSIVE WEB DESIGN FUNDAMENTALS

Lesson 4 Handout

Navigation Patterns

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. Navigation patterns are a critical component of every website since they dictate how your users discover and find the content most relevant to them. With the rise of mobile browsers, navigation that works equally well on small and large screens can be hard to get right.
2. Navigation that works well for one website does not necessarily translate to other websites. Important variables to account for include: the size and architecture of the site, the primary audience, the type of business or organization the site represents and other factors.
3. Many traditional desktop navigation menus relied heavily on the hover state because the primary interface is the mouse. However this practice is problematic for touchscreen devices such as phone and tablet where the hover state is not available. Any navigation you implement needs to take touchscreen interaction into account.
4. Mobile navigation has evolved slowly over the years. The “select menu” navigation pattern was a common choice. The “hamburger icon” pattern has also been popular, although is often criticized as not being identifiable as navigation to many users, especially when used on desktop layouts.
5. The “off-canvas” navigation pattern is useful because the entire navigation is self-contained and appears and disappears as needed, this typically means there is more room for content. In addition, off-canvas typically has a sub-menu component allowing you to map a deep navigation, which is especially useful for large sites.
6. No matter which navigation pattern you choose, it is important to anticipate accessibility concerns and add support as needed. For example, menus should be accessible with a keyboard or screen-reader, the time it takes to add this support is minimal and in return you make the web a better place for all users.

ASSIGNMENTS

- Quiz
- This second assignment is designed to help you keep thinking about design and navigation patterns. This assignment comes in two parts, although the second part is optional.
    - Part one: read this article by Brad Frost which defines an interface inventory, here's the URL: http://bradfrost.com/blog/post/interface-inventory/
    - Part Two: This section is optional and will be most useful if you have an existing website that you are hoping to redesign. Now that you know what an interface inventory is, go ahead and perform one on your current site and post your results on the forum. Alternatively, you could choose another site that you feel could benefit from an inventory and post those results.

INTRODUCTION

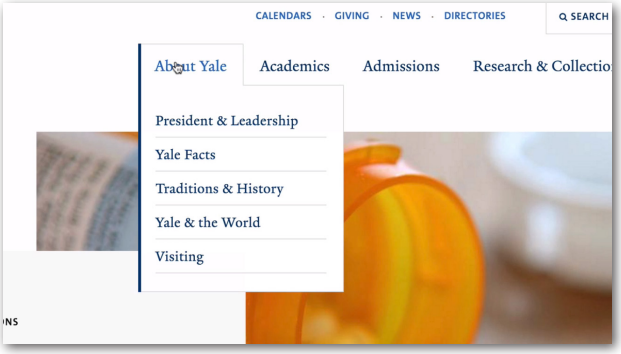
(Note: This is an edited transcript of the Responsive Web Design Fundamentals lecture videos. Some students work better with written material than by watching videos alone, so we're offering this to you as an optional, helpful resource. Some elements of the instruction, like live coding, can't be recreated in a document like this one.)

In this lesson, we're going to take a look at the history of navigation patterns that have been in use on desktop and on small screens. We're going to talk about appropriate context, and what works well on small screens and large ones. And we're going to take a look at our solution at both ends of the scale and see how that works on a phone all the way up to a large desktop.

CHAPTER 1, RESPONSIVE NAVIGATION RETROSPECTIVE

So in this section, we're going to take a look at desktop patterns for navigation that are pretty common. And we're going to take a look at how that's been reinterpreted for small screens and then see how some of those patterns have actually worked their way back into what people do on larger screens and see what's working and what might be a bit of a miss.

So I've queued up a bunch of websites for us to take a look at. And this one from Delaware Valley has a really, really tried and true main navigation coming across and nothing really happening here. So it actually works really well on the web with the mouse interaction, but also would work well with touch if you were looking at this on a tablet. If you click on one of those links, and it brings you to that section with your secondary navigation exposed. So pretty simple to implement and will work under basically any circumstances.



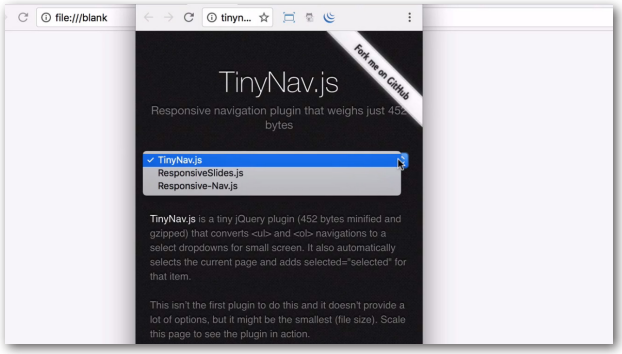
And another pattern that's a pretty familiar one that we've seen on the web for many years is a dropdown on hover. So on the Yale website, you hover over any one of these links, and a bunch of options show up. And you can click and browse to see what else is in that section.

And that evolved as we got a little bit more sophisticated into a really elegant solution that Dartmouth's put in place, that actually on hover, exposes almost like a site map all of these main areas. So it actually allows you to browse and see what else is available on the website. From really anywhere in the site, you can always get to pretty much everything

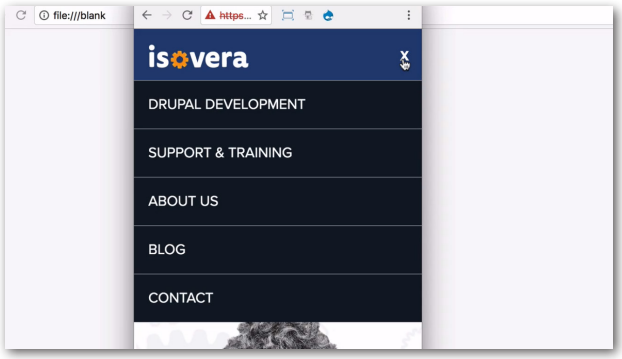
else. And that one works on hover, and it's pretty reliable, and it's got a large touch area so that you don't end up losing your way and sliding out from that menu and having things disappear on you.

Now let's take a look at the way things evolved on small screens. Now the first thing that we saw evolve when we came around to building sites for small screens was basically just taking that same navigation structure and turning it into a select menu. Now, it was easy and a lot of people gravitated to that. And you saw it all over the place, but it's actually a really tough interaction, especially, when you view it on a phone.

Oftentimes these selections will only show you three or four at a time, and you never actually get to see the whole navigation at once. So really, it was easy, but it was really not a very good interaction. So a lot of sites, if they had a smaller navigation structure ended up just not really doing anything at all. They would just leave those buttons exposed, and you can just tap to go where you need to go.

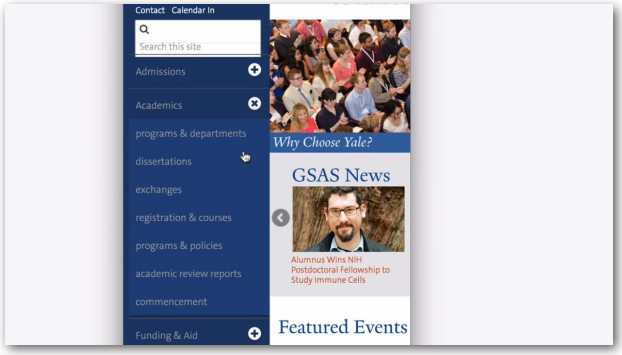


Now as we got a little bit more sophisticated, we see the evolution of the little hamburger icon, and see the menu slide down across the top of the content. This one's pretty easy to implement. You can see it in a lot of websites that have a fairly small navigation structure that is not very deep. So in those cases, it works really well on some of those relatively small scale sites.



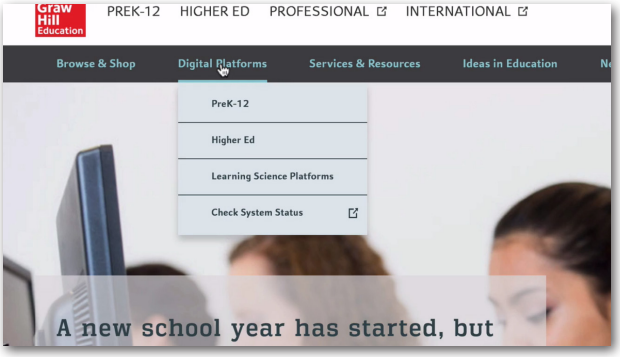
But that's often not really manageable, or sustainable. And we saw that when working on this website for the Yale graduate school. It's a large graduate school that has a lot of options in the navigation, and we knew that we'd need a little bit more support and flexibility.

So we used a pattern called an off-canvas pattern. When you click on this button, and it actually slides in from the side, and you'll see that it actually gives you the ability to expand and push down along with the content. So you can get a really deep navigation structure that actually will continue expanding as much as necessary as you get deeper into it, automatically adding these secondary little icons.



Now an interesting thing happened on the way to the small screen as some of these things started to migrate back to the desktop. So if we take a look at the McGraw-Hill site, one of the big challenges with these hover interactions that we saw on the Dartmouth site is when you're using a tablet, you don't have hover. There's no mouse. So that first interaction is a touch and most of the time people expect that to be a click. But if you have a hover, it really just enables that first view.

So here on this McGraw-Hill education site, they actually explicitly took the hover away, and you only expose that navigation on a click or a tap. So while it's a little bit disconcerting when you end up having some that expose secondary navigation and others that actually take you to a page, these things are fixable. And you could end up adding an icon to one of these so that you'd have an understanding that it's a different interaction. But it does give you some interesting possibilities to make sure that your design is going to work well on both touch screen and hover mouse interactions.



This last one, though, I think we're starting to see an awful lot of websites where they've actually taken the exact same pattern that's in use on the small screen and just made it available across the board. So with this website, not that I want to pick on this agency because they do a lot of beautiful work, but they've left us just with this icon here which a lot of people still just don't recognize. There's no additional cue, and you have to know to click on that to actually get any navigation at all.

So we've had a look at some of the evolving patterns, and we've looked at some of those strengths and challenges that each of them have. And we're starting to think about, what are these browsing patterns like, and what are the habits based on what kind of device it is that you're using? So let's take a look and see which ones are going to be really successful and which ones are going to present us with some challenges.

CHAPTER 2: CREATING A MENU BUTTON

So let's take a look at our solution. And we're going to start on the small screen. So we're going to go through creating the off-canvas style. We're going to look at how we're achieving this multilevel expansion, and we're going to talk a little bit about what the accessibility concerns are here.

Here's what you'll be building, an off-canvas navigation that allows us to access as much of the site as possible in a single screen and then disappears as needed. So we're going to open up Chapter 4, and we're going to take a look and get our index start page. That's what we have open here. And I also have that page open in the browser, and we'll take a look and just remind ourselves where we are. And we have our desktop navigation that just scales, just gets narrower.

The text reflows, but it doesn't really work all that well on the smaller screens because it just doesn't fit. So what we want to do is take this same structure and start to make it work off-canvas. So we have to do a few different things here. And one of those is we need to add the button.

So if we come down, and we look for where our navigation starts, one of the things I wanted to point out is we have this aria landmark. This navigation here is something that's really important for accessibility, and we want

to make sure we don't lose sight of that. So what we're going to do is actually bring in the button that will activate this off-canvas navigation, and we're also going to make sure that this nav region itself is recognized by screen readers and other assistive technology properly.

So we take a look at the final file and look at the difference. Really the main thing, is we have this line right here. So this is our aria control that is going to toggle this menu. So we're going to copy that line, and we are going to go back over to our start page. And that goes right inside this nav object. And just go ahead and save that. And that is the big difference between these two files.

The one other thing that we have to do is we're actually adding this one little wrapper. So we're going to add this div and make sure we close it out in the right place. And we're going to come back here. Right underneath the line we just added, we're adding this div with the class of main-nav-inner. And then we're going down to the bottom of that UL and closing that div. So all this will do is add a menu link within our navigation bar. Let's take a look.

It's not exactly what we were hoping for. So we have to style it, and we've got to move it out of the way. So let's go take a look at how we're doing that. So we take a look in here and we're going to find our menu toggle. And we've got some basic styles for it that are going to set the text, add a little gradient.

We're just going to grab these styles and bring them over. And this goes in right here where these main Navigation Styles are starting. So we'll add this in right above this block here. So this is going to give us some bit of styling for that little toggle button. And we come back and reload.

You can see now we're getting somewhere. We've got our little button, it's positioned just above where the navigation is. So we know that it's still in that aria landmark navigation area, but it's positioned outside, so we know that it will stay visible when we want it to, right tucked into the bottom corner of this header area.

Now the other thing that we want to take a look at is to see where our little icon would be and come back over here and take a look. We want to grab another little bit here. When we have font face support, we want to go and grab this little bit of CSS. And we'll get both versions for when it's active, and when it's not. And we'll grab some of these other styles for link colors as well.

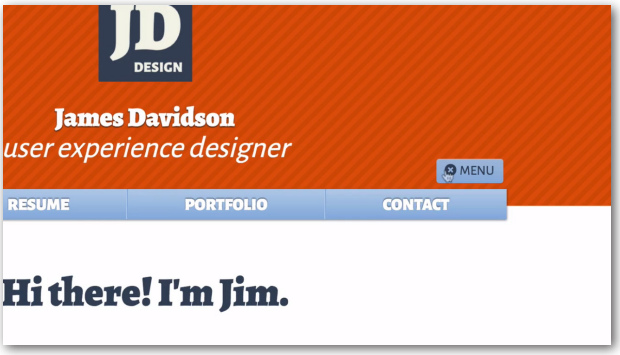
I'm going to copy this whole chunk, and that's going to go right back in where we are. Now we left out one little chunk there that's going to hide it on large screens because we just want to make sure that we have everything looking the way we want. There we go. We've got our icon. We've got our styles. That's looking proper.

So now, that still doesn't answer the question of what we're doing with the rest of this navigation. As we move this stuff around, we just got a button sitting there. It doesn't actually do anything. So let's see what else we need to do.

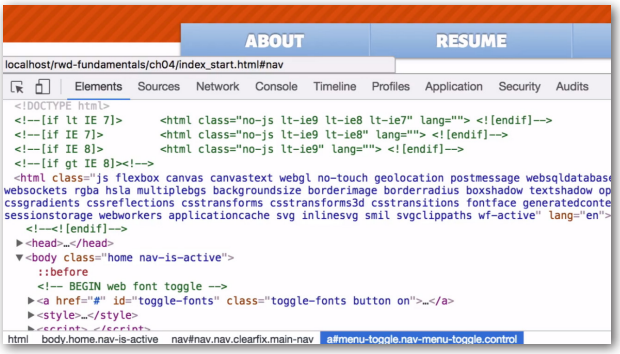
CHAPTER 3: ADDING JAVASCRIPT AND BASIC STYLES FOR YOUR OFF-SCREEN NAV

First of all, it has to act on the rest of this navigation. And that means we're going to add some JavaScript, and we're also going to add some other styles. So the first thing we want to do is go take a look at the JavaScript that already exists in our main.jsfile in the JS folder inside Chapter 4. So if you scroll down, you see we have a bunch of navigation functions in here.

Now what we've done here is actually written this in a way so it doesn't require jQuery. This is all using native JavaScript. So it doesn't have to load anything extra. It's going to go through and look for a few elements and add an event listener. So it looks for menu toggle, it adds an event listener for a click. And then it does a couple of things. It adds body classes or removes them. It does the same thing with submenus and it loops through finding all of them to make sure they can add all the icons that we need.



So when we actually take a look at this in action, since we already have this JavaScript connected, when we load this in the browser again, if we click this, we'll see something's happening. That little icon is changing from a little hamburger menu to a little x. So that's not really all that helpful because it's not actually acting on the menu. So let's see what this is actually doing to the page.



Go in to inspect this, well take a look at what's going on down here. Now, watch what's going on on the body class here. So take a look at the body tag here. The class of home, but that's it. And when we click here, we see nav is active. So this is our first little bit that we need to put in place is to get this class on the body so that we can scope our CSS to get the menu to behave differently.

So let's look at what else is going on. When that navigation is active, it's also going to start to look for sub-navigation, and look to see if it needs to add anything else there. Now, let's go back and look at the styles. So in order to get the styles to work off-canvas, we actually need to start to bring in styles based on that new wrapper that we added, main-nav-inner. And this is going to actually start to control what's going on with our CSS and with our navigation.

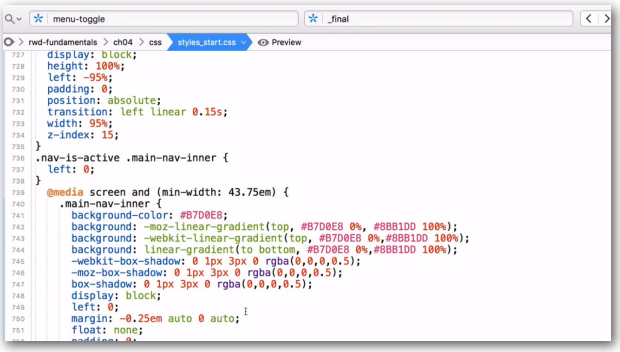
And I can see, we have this set up so that we're working mobile first, so when we add this CSS, it's actually going to move that menu around. And we're then going to reset it back to what it's doing on the larger screen

once it hits that break point right there. So if we just add the first section, and we're actually going to put it in right here above the main-nav styles.

When we save this and go reload the page, we start to see something going on. It's not exactly what we want yet, but it's shoved that navigation off to the side. And now we need to keep evolving this and bring more of this over. We've taken it so everything is being controlled by this main-nav-inner which is the wrapper that we've put around this navigation object itself.

So let's bring this media query in, so that we know we've got everything covered for large and small screen. So what this is doing is it's giving us all of the styling that we want with the background gradient and all the margins to make it look like we want it to look on those larger screens. There we go, perfect.

So this is getting it reset so on large screens it does what we want. We have our body class being added here. That's only going to do something—there we go. See, it's just playing with the positioning, just based on what's going on here.



Now again, it's not exactly what we want. We want to style it a little bit differently, so that we get something that's coming out smoothly, but it's formatted appropriately for that small screen. And going across isn't really the right way to go.

So let's take a look and see what we're going to do next. So we go back and look at this, and the first thing that we need to do is start to look at styling this with Flexbox and make sure that we have all the right styles being applied. Most of that should be in here already.

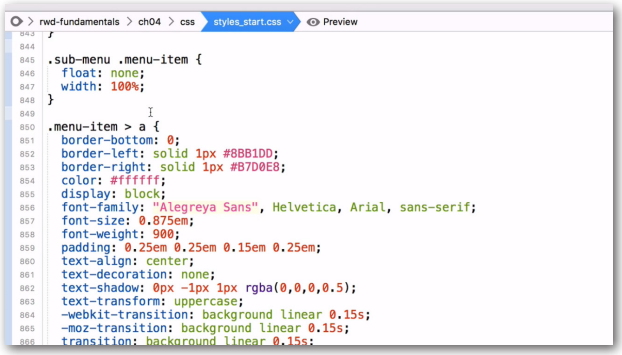
I've got nav menu styling, but this is actually not keyed into our media queries. So let's look at some of the differences here. Because the nav-menu styling, we only want to apply that Flexbox styling when it's on that larger screen.

So we're going to come in and replace a little bit more of these styles. Styling the nav-menu elements and when you have Flexbox. We're going to apply the Flexbox styling, but again, only when it's on that larger screen. As we come back over here, and we take a look at where these styles are applied, it's right here all the way down to here. And when we replace that, we've shifted all of our styles for Flexbox into only being executed when you're on that larger screen.

So again, if we come back, we're going to start to evolve this a little bit and see what the differences are. Nothing happening here, of course. We'll need that screen smaller. Notice how we've taken the Flexbox styling away. We start to see how some of this styling is starting to collapse. But it's still not doing quite what we want.

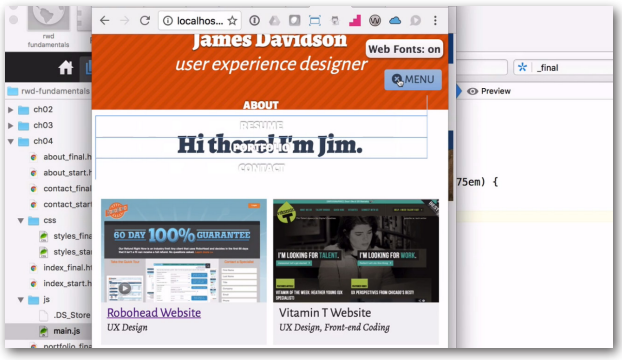
So again, we're going to go back, and we are going to take a look and see how we're working these styles back into the

smaller screen. And we look at what we're doing with each menu item. So this is getting into each item in that top level where we have in our CSS—when we look at the styles that we have here, it's got some of these borders, as well as some of the borders on the left and the right. As you see down here, we know that some of these styles are really only relevant on the larger screen, and we want to take a look at styling them a little bit differently.



Now, let's take a look and see how those things are shifting around. When we look at these menu items, first we'll want to display them as a block before we start floating them. So let's do this. We're going to take this little chunk, and then we're going to add in the media query, but we're not going to put any styles in there yet, so we can see what the differences are.

I'm going to put these styles in here, and then we're going to close out that media query. Now we want to take these styles away and make sure that we're only styling things the way we want on the small screen first. So let's take a look at this again. When this comes out, now we're starting to see something that might be more appropriate. We've got things one above the other, starting to be in a format that we think might be a little more appropriate for that small screen.



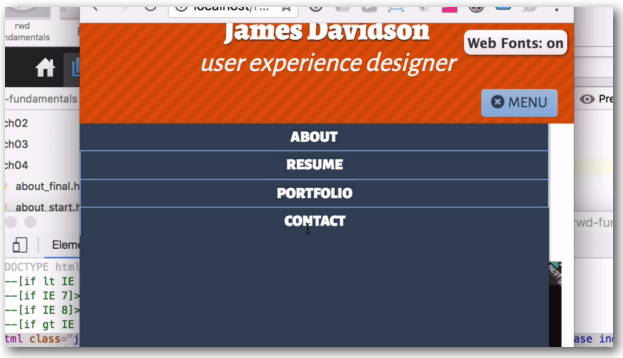
So right away, just with some simple positioning, we've got it sliding out from the side, and we've got it stacked one on top of the other. Obviously, it's not that usable yet, but we'll get there. So let's go back and start to look at some of the other options.

CHAPTER 4: ANIMATING & ACTIVATING YOUR OFF-SCREEN NAVIGATION MENU

OK, so one of the things that we did while we were working on this was start to bring in some of these styles up above for this new wrapper that we created, for menu-nav-inner. But one of the things that we have to make sure we do is actually get rid of the older styles. So these styles right here, main-nav and main-nav-aria-expanded true because our wrapper has actually taken care of that.

So we can take this chunk and delete it, save that. Let's go back and reload our page. We're starting to get a little closer. So there we go. We've got a nice blue overlay, it animates out smoothly. And we've got our links here, probably still want to style that a little bit. And you'll also notice that we're still not getting exactly what we want here because we can't reach down into that portfolio section.

So let's go back and see what else we need to bring in. So we come back and we look at our code here. We know we're going to have to style those buttons, and we're also going to take a look at a little bit more and a little bit more detail, how are we making that animation work so nicely? We know that it has to do with the class that's been added. But let's go back and take a quick look here.



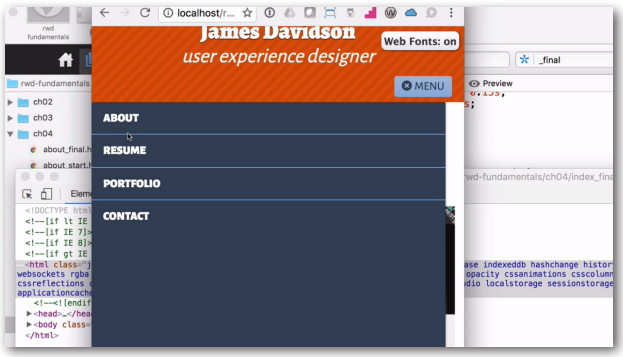
We've added our nav menu toggle. And I'm going to come down, and we have our main-nav-inner. And then we have nav is active, main-nav-inner. So we notice that it's positioned off screen. It's got a width of 95%. And it's positioned minus 95% way off to the left. So that hides it effectively just off screen. And when that class is active, it brings it back to 0, meaning it covers 95% of the screen.

And the way we make that work so nicely is with this transition. It transitions from the left in a linear path and in easing and it takes 0.15 seconds. So we can speed that up or slow it down, but that one simple little CSS transition, all that we need to trigger it is the JavaScript that is swapping out that nav-is-active class.

So that's how we get it to slide across. Now we want to go into those elements and actually style them a little bit better. So we want to look at those menu items themselves at that top level. So if we go back over to our final file, and we are going to take a look at these first items, these first menu items. And we have some styling for that first level link. And we want to make sure that we get this just right. We'll go ahead and grab this whole chunk, and we're going to place the whole bit that we have here in our file.

So everything from this line all the way down here. We save that. We're going to come back and take a look again and see how our styles are evolving. Look at that. Now we're getting somewhere. So we've got our text line to the left. We've got a little bit more room. We got a little bit more padding above and below. And this is starting to line up in a way that looks just about right.

And you'll also notice that, see how that link, that hit state, it's actually all the way over here. Because what we've done is made sure that this A is display a block. So that means it's going to fill that entire area until we put something else in there and make the whole hit state active. So when we touch based interface, it works really nicely.



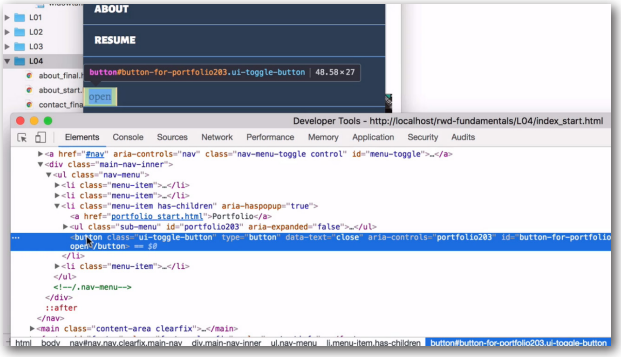
Now one thing that we're still missing is how do we expand here to actually see what's in this portfolio section? So let's go back over to our JavaScript file and have a look. And one of the things you might notice is there's a bunch of stuff that's commented out. So the first thing we're going to do is remove those comments and take a look at what's going on.

So I'm going to remove this. I'm going to remove this at line 107 and go back up and delete that at line 60. Now, in this submenu section, a few things are going to happen that are really important. So first of all,

it's going to look for any sub menus. It's now going to go in and create a button for us. It's going to add an element that is going to create this UI toggle button, and it will set a bunch of attributes here and name them randomly for each instance. So that way you can be sure that the button will bind uniquely to each submenu.

It's also going to add a has-children class to the parent, or the li. And then it's also going to set an aria role that expanded equals false. So all of these things together tell the browser that there are children that we can target with CSS, that it's not expanded yet, but it is adding the toggle button for us.

So if we do nothing else and simply save this file and go back and reload the page and take a look at this, you'll see right away we have a nice ugly little button that says open. And if we inspect this, you'll see that this button has now been created right next to the submenu. You'll also notice that we've set a unique ID. The aria-expanded role is there. And we have a button that targets it. So this is what we're going to be able to use with our CSS and JavaScript together to expand and show the contents inside right here.



One thing that's worth looking at while we still have this open is that if we click on this, and I'm going to move this window over just a little bit, when we click on this, notice a few things are happening. Aria-expanded is now true, and the text changes from open to close. And that tells us what will happen when you click on that button. Now it's not showing any styles, but at least we know the JavaScript is working as it should be.

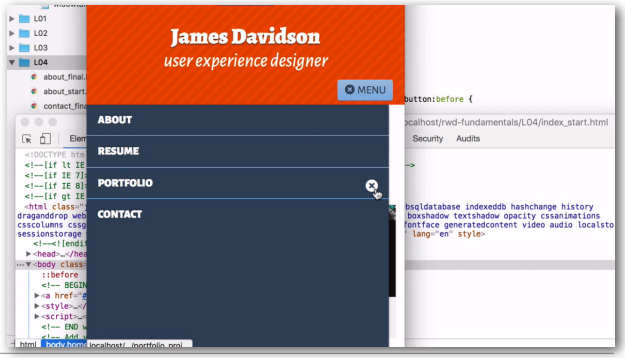
CHAPTER 5: STYLING & ACTIVATING THE SUB-MENU

So let's go grab the styles that we need to make this all work. If we look in the styles final file, and we look around for a UI toggle button, that starts right around line 981. And we're going to take all of the styles that are related all the way down here to around line 1039.

Copy all those styles, and we're going to go back and add those in here. Again, that was starting around line 980. So I'm going to paste these in and hit Save. So these styles, I'm going to walk through these quickly. A basic style for this UI toggle button that's going to set an actual style for it, it's also going to hide it when the screen is big enough. And it's also going to attach a font awesome icon and then hide the actual text.

So we still have a button, but it has an icon instead of the text. And it will show up exactly how we like it. And it has an on and off state to it, and we've again, scoped this properly so it will only work if modernizer detects font face support.

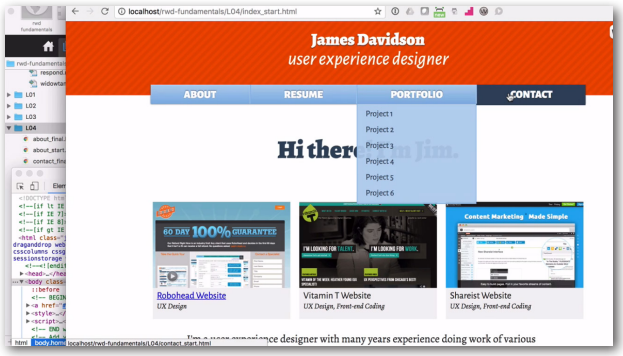
Now again, if we go back and reload our page, now we've got exactly what we were hoping for. We've got this little icon here that shows we can expand and see what's there. We click on



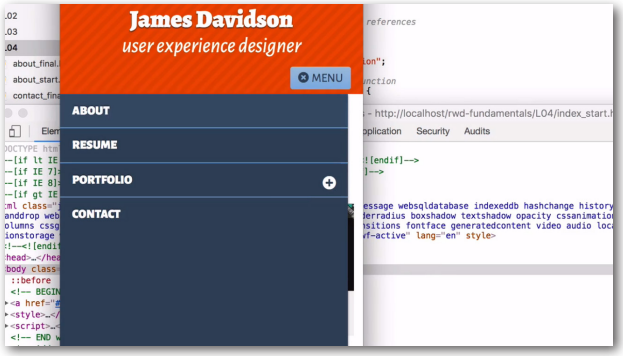
it, and we don't see anything at all. So we have something functioning properly, but it's not actually showing the styles. So let's go see what we're missing, that's probably some styles for the submenu itself.

Let's take a look at the styles that are present here. So the styles for submenu, it's set here, so that when aria-expanded is false, there is display none. But we also see that it's also shoved way off to the side, and that means it's not really going to show up until we set some actions to happen. So if we look in the styles final and look at the submenu section here, you'll notice that some of this looks the same, but there's also the addition of some styles that are going to bring it back in line once it's actually focused properly.

So if we grab this down to here, copy it, and we're going to replace the existing submenu CSS here. Now this should cover us for a variety of scenarios, and when we go back and reload the page again, we see the icon. And when we click on it, now we get our sub-navigation.



Now let's double check and make sure that it works just as well on the big screen and we slide our cursor over, and everything's working exactly as it should. So now we have all the styles that are necessary for both small screen and large. And we also get the added bonus that by supporting the aria roles and also the added JavaScript functionality that we have down here for this drop navigation, thanks to Scott O'Hara, we've got some great code that will tie into whether or not the aria roles are present in that submenu that will also enable this to work when we're tabbing. So people with accessibility concerns, alternate means of input, they're going to have just as good an experience.



So we take a look and see how that works here on the large screen. I'm just going to start tabbing through, and you'll notice that it opens up what's below right away. And we can tab through each one in turn. And then if we go back to the small screen, it works here too. So we just tab. You can see the outline around the logo, the name, now around the menu. Hit Enter, start tabbing again.

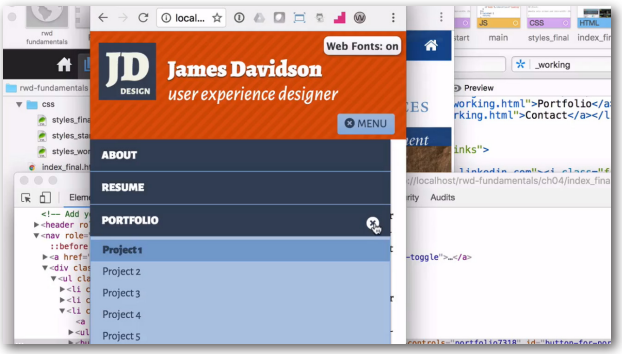
Now our little plus icon expansion UI toggle is active, enter again. It pre-selects the item just below so that people can move through the navigation really quickly. Then we can close it again.

And now all we have to do is go back to index start html and take the things that we changed, which are really only a couple of minor things. We've added the nav menu toggle and the wrapper for the menu inner, and we changed the reference to the JavaScript file, which is all the way down here at the bottom. It's just going to this JS main folder. If we make those same changes in the rest of the pages in the site, we'll have this great accessible navigation across small screen and large on our entire site.

So I'm going to grab this whole nav block, and we're going to bring that into our other pages, OK. Now you may have noticed that I started out doing this work in index start. So probably should have been doing that in the working file. We're going to copy that over. And we'll bring this over into the rest of our pages.

So again, we're replacing this whole nav chunk, so that it's got our main-nav-inner and our button. And I'm going to go back to each of these working files and do the same thing. Just a couple more. You know, we'll just make sure that we can move around as we know we should be able to do on a small screen. And in the next section, we'll go through and add some more tweaks to the large one. And we'll take a look at what's making all of that work.

We have to make sure that they all are linked to the right JavaScript file, as well. So I'll scroll down to the bottom and make sure that we update that link. So now, we reload here, and we go to any of these other pages. There we go. So now I bring this on the small screen, bring our menu up here, and everything is working nicely, all right.



So there's our small screen navigation. It's going to work with no matter how many links we throw at it. It's going to work nicely on all these different screens. And it's going to animate nice and smoothly and still respond well to alternate forms of input, OK.

Once again, wrapping up that section, we have a menu that hides off-canvas on a small screen. It works for multiple levels of drop-downs. And it's working almost entirely the way we'd like on a large screen. But I'm going to take a look at a couple of things that we're doing there just to make it work just that little bit better.

CHAPTER 6: IMPROVING NAVIGATION ACCESSIBILITY

So let's look at our solution at larger scales. Now we have a CSS-only solution for the dropdown menus, but there's a couple of accessibility things that we want to look at and see why we're using a little bit of extra JavaScript and CSS to make this work just a little bit better, a little bit more enhanced to make sure that people using alternate forms of input like the keyboard are still going to have just as good an experience. We have our site, we know it works on a small screen. And we know with the mouse, it's doing exactly what we want.

And this is actually working really nicely and doesn't require any JavaScript at all. And the way that works is by actually hooking into a pseudo-selector. So that when we're hovering over that item, it's actually the fact that we're hovering over it, and we can check the event listener that triggers it. And once that's been triggered, it expands that entire list item so that we can slide our mouse down to any of these items in between. And it will always work even if we don't have JavaScript, which is a really nice way to handle this and make sure it works in a broad variety of circumstances.

However—let me get rid of that—when we start to think about keyboard access, and we start to tab, do we see the focus rectangle around these links? Now it disappears. So somebody with a keyboard isn't really going to know it's going to happen. If we put the mouse over it, we see that it actually did go and select that next item. But we're actually not able to target it to keep it exposed because we're not allowed to force the hover event on

that enclosing list item. So my friend, Scott O'Hara, actually helped me out with this a little bit, and we have a little more JavaScript and a few more styles to make it work.

So let's go and take a look at the code, and we're going back over here to our JavaScript file. And if you look down below the area that we were looking before to make the dropdown menus tabable—thanks Scott—we have another bit of code that's actually going to go through and create a function that's going to look around and see if it can find a submenu.

And when it's there, and we're actually focusing over it, it's going to add this attribute, aria-haspopup. And that's actually going to be the trigger to give us this class. If it's present, we'll add a has-focus class when we're mousing over it. So we need to add a little bit more CSS.

We're going to come back over to the styles here. And we're going to grab everything from menu active trail all the way down to our UI toggle button. There's a few little bits and bobs in here that we need. So when we paste all of this code into the right file, into styles working, save it, and make sure we go back over here, reload the page, and then start tabbing through our navigation, there we go. Perfectly keyboard accessible navigation, which means it's also going to work with voiceover and other sorts of assistive technologies.

So there we go. We've got really nice off-Canvas navigation on a small screen that works on a whole range of devices and all levels of navigation and slides right back out on the big screen. It works really nice with the mouse and responds well with keyboard access as well.

So there's our navigation. And it's going to work well for our project, and it's going to work well for a lot of other uses as well. So we've added dropdown styles and tested our keyboard access, and we've tied everything together in our navigation patterns for both small screens and large. So in the next lesson, we're actually going to start to look at some advanced enhancements and talk about adding responsive images and try that out and look at some other things coming along like element queries which will take our responsive design even further.