AQUENT

GYMNASIUM

JQUERY BUILDING BLOCKS:

THE JQUERY CAROUSEL(S)

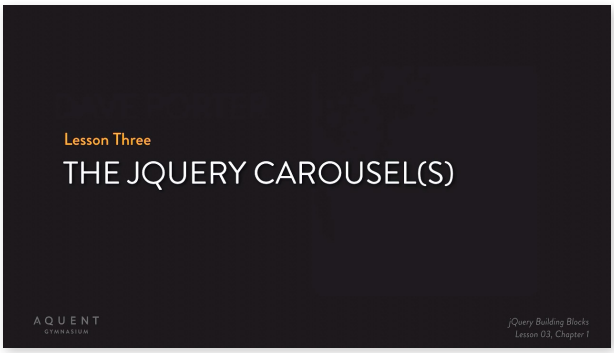
LESSON 3

ABOUT THIS DOCUMENT

This handout is an edited transcript of the jQuery Building Blocks 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.

INTRODUCTION: THE JQUERY CAROUSEL(S)



Welcome to jQuery Building Blocks, Five Techniques to Cut Your Web Development Time In Half. This is an Aquent production. I'm your presenter Dave Porter.

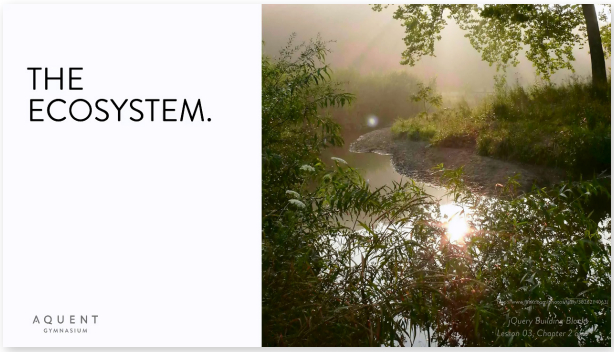
This is Lesson 3, the jQuery Carousel(s).

Chapter 1, Introduction, in which I will introduce you to, among other things, Chapter 1. In Chapter 2, we're going to take a look at the jQuery plug-in ecosystem. We're going to take a look at the really fantastic breadth and depth of the plug-in ecosystem that jQuery has built up around it.

In Chapter 3, we're going to talk about carousels, how to use them well in your websites, and how to avoid misusing them. In Chapter 4, we're going to take a look at the carousels in jQuery, the plug-ins that are available to you, and we're going to pick one and make some good use of it.

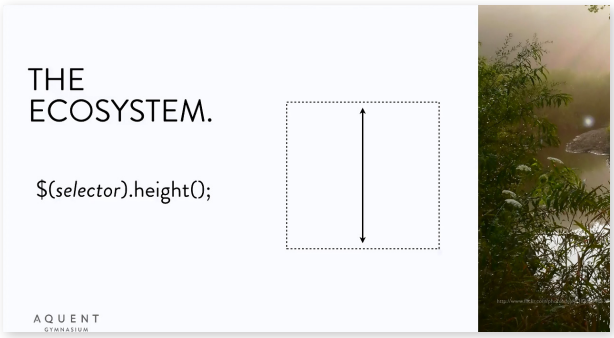
Then in Chapter 5, we're going to actually get our hands dirty. We're going to create two carousels today. One from a bunch of photos, just a simple photo slide show and one that the user is going to have full control over. And in Chapter 6, I'll wrap it all up and everyone can move on with their day.

CHAPTER 2: THE JQUERY ECOSYSTEM



This ecosystem is really one of the top line features of jQuery. It's one of the major ways that jQuery is going to be able to help you out. But what is a plug-in, beyond the obvious: something that plugs in and gives you more capability? The major way that you use jQuery is you get some elements together and then you do something to them, or you get some information from them.

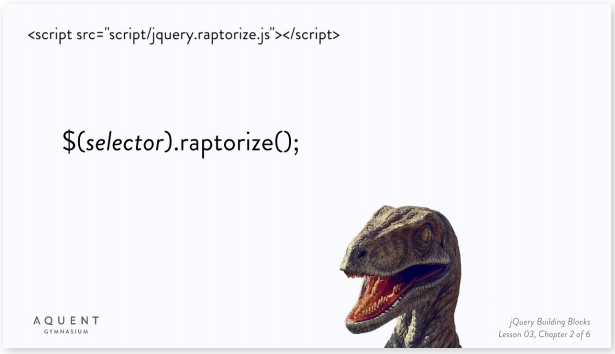
You do that by calling a method. Specificially, you use a selector to get your elements, and then you call a method. Now most of the time, what plug-ins do is they give you new methods. Which means that they give you new things to do to elements and new things to find out about elements. For example, height is one of those methods. It's a built-in one. You don't need a plug-in to get it. It comes with standard jQuery.



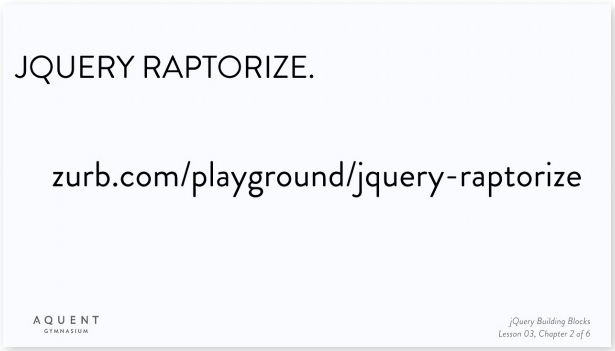
This means that if you want to find out the height of an element, you just call height on it and right away you get a nice clean answer. Now, if you want to do something else, let's say you want to raptorize your elements, whatever that

might mean. What you would do is you'd call the method. But unfortunately, it doesn't exist in vanilla jQuery. So you're going to get an error.

But if you go out to the Internet, you track down the plug-in, you add it to the top of your page and then you call this method, now it's going to be there. And you're going to have whatever strange effect it is that you wanted to get. So most of the time, plug-ins are going to give you a new method like this to call. Sometimes they improve existing methods too. Plug-ins can be absolutely enormous— jQuery UI is probably the best example of a really big plug-in.

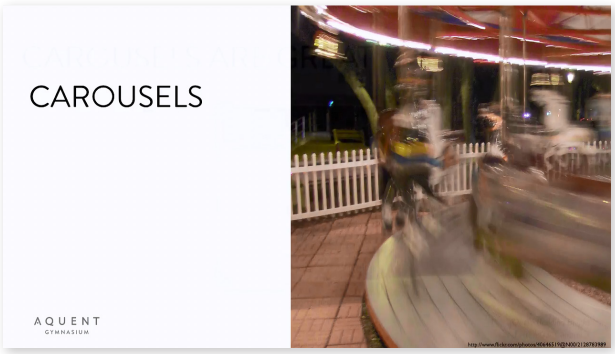


This plugin simplifies important interface processes like drag and drop. Or reorder, which makes reordering items very easy. It also implements complex widgets for you like sliders and color pickers like this and makes them truly plug and play easy. You can check this out at jqueryui.com although we're not going to be looking too closely at it today. So they range from the really huge to the really focused and the really well done single feature plug-ins.



This is Easings from easings.net, which gives you access to the timing functions from jQuery UI without having to load all of the extra widgets that jQuery UI brings with it. This is something called Ken Burns plug-in. This is from my own website, and I use this to put together a really simple Instagram photo slide show. So plug-ins can range from the small and focused like this to the silly. This raptorize thing actually exists, and it's pretty silly, but it's pretty awesome at the one little thing it does. I recommend you go check it out at the website here.

CHAPTER 3: CAROUSELS

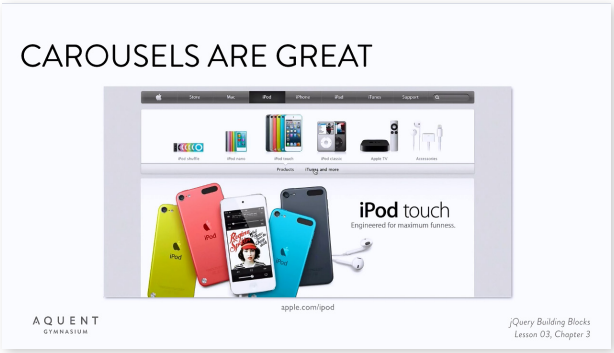


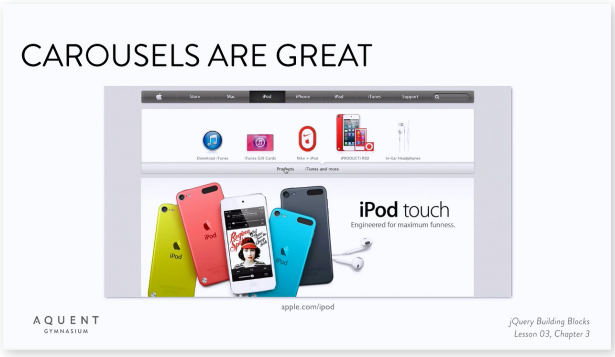
if what you need is a carousel, then jQuery's got you covered there, too. Before we get in to picking a plug-in, though, I'd like to cover carousels themselves, specifically the user interface concept and their uses and misuses here in Chapter 3. Now, carousels are great. They're great for browsing photos, like my personal website from the last section. They're really nice for browsing unordered, slow moving content.

This is hulu.com. They've got a ton of content, and they don't really much mind where you settle in. If you see something that piques your interest, then Hulu has won. They're great for these corporate impression sites. You know, where you've got some light interactivity, maybe some calls to action, but mostly it's just conveying feelings. This is the front page of wholefoodsmarket.com.

And you'll notice I'm saying the word browse a lot. The reason for that—that's on purpose. I mean browse like cows browse grass. It doesn't much matter what part of the field the cow's in. It doesn't matter where in the content you as the user are, as long as you've got some content in front of you, it's kind of interesting, you've got the ability to change it. It's changing as you go. There's no particular hierarchy, and so on.

Now, at the other end of the spectrum, carousels are great for explicit user guided interactivity. Now, we've seen this before in the last lesson. This is from apple.com/ipad. They've got this nifty little carousel up at the top. It shows a bunch of products. And then since they couldn't fit them all on one screen, they've got a whole other section. If you click on the other section, it swoops over. The next stuff swoops in like this, and you can jump back and forth.



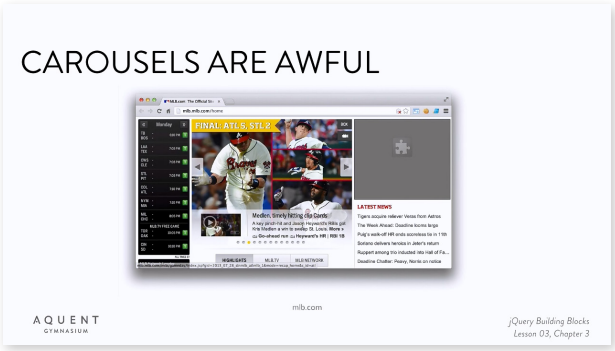


Now, this is great because the user is fully in control and it's very clear how to get to where the user wants to go. So carousels are great. But with all the cheerleading out of the way, here's a word to the wise. It is really, really easy to misuse carousels. It's so easy that there's actually been a bit of a backlash in the user experience community. And it's well founded. So I want to walk you through some of that.

Now, this example is from bbc.com. There's lots of detailed text. There's some unintuitive navigation. It just doesn't feel right. Which is too bad, because BBC has got some great carousels on other parts of their sites. Another common problem is transience. This example is from mlb.com, and it's slow. I actually sped up the video for illustrative purposes. So it's not too bad. But if

I see something that I want to read, and I go back to it like that, and then I'm reading it and I look away for a sec because someone walks into the room and says hi, and when I look back, it's gone.

It's moved on. I have to go and reverse again. And this can happen right in the middle of interaction. And this transience is something that it turns out really, really negatively impacts users. And studies have confirmed that. They've shown that carousels are really fragile. If you have a carousel that's just a little bit wrong, it absolutely craters in terms of user interaction.

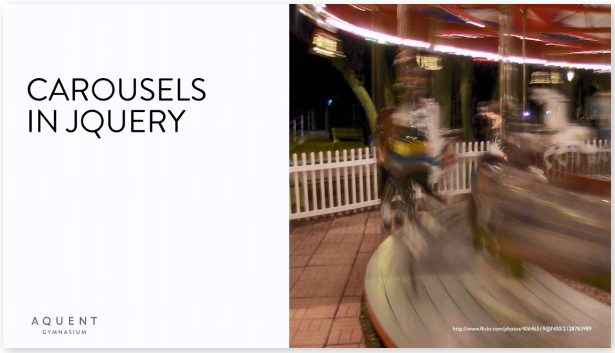


So it's incredibly easy to convince a user to take their mouse somewhere else when you're dealing with these things. So they're great, but they're easy to do really badly. The lesson materials have got some further reading for avoiding common faux pas with carousels, and I recommend that you take a look at them before you implement this in your own project.

Regardless of whether they're awesome or awful, they are everywhere. Anytime you've got a photo, anything. Anything on corporate brand sites, they're going to show up. On e-commerce sites, they're going to be all over. So at some point in your career, you are going to run into a project requirement for a carousel and the important point that I want you to come away from today's lesson with is that a carousel is a 10 minute requirement. It's not a 10 hour requirement. jQuery and jQuery plug-ins make it super, super easy.

CHAPTER 4: CAROUSELS IN JQUERY

With that, we're back around to jQuery. We are going to talk through the process of choosing and finding and downloading a jQuery plug-in. So let's go.



Last lesson, we played the role of someone who's boss had come up with the genius idea of imitating Apple's quite nice shelf experience on their website and adding it to our poster store. So we went and made that happen. This week our final product is going to be a carousel.

We're taking that same shelf and turning it into a carousel. Before we get that far, we're also going to put together a quick slide show. We're going to turn the portfolio from lesson one into a slide show like this.

So with that in mind, the first things first: go find a carousel. This is a situation actually where if you go out on your own and try and find one, you're going to be confronted rapidly with the fact that there are just absolutely tons of jQuery carousels. We've got 15 of them in this guy's blog post, 10 of them here, 45 best ones going on here.

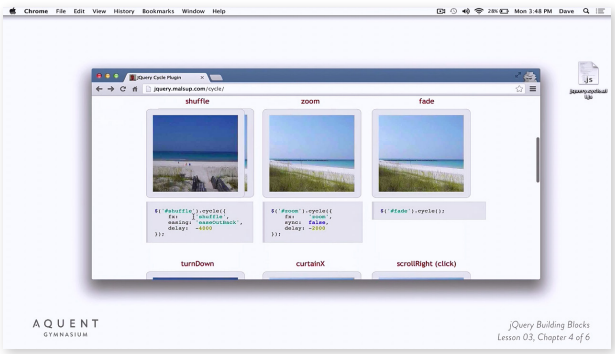
So you know, experiment. Find the one you want. Or talk to your good friends and figure out what they like. Since I'm your good friend, we're going to use jQuery cycle. I've used it for a while. It works well. I like a lot, and it gets her done. So today we'll be using jQuery cycle.

The thing that it does: its got a series of effects, different ways you can make transitions happen. There is also built in support for this kind of on action activity, which is what we're going to want for our shelf work later on.

First up, let's download this. There's a nice big download link right here. If you actually click on this, you'll just see all the code. That's now what we're looking for. So right click and "Save link as." Then drop that on the desktop for easy access. So there it is. We've successfully downloaded jQuery cycle, let's hold onto that for a bit.

Let's go back now and take a look at this page. They've got some code examples here, and that's nice. There's two other things that we're going to want to have available as we move forward with this.

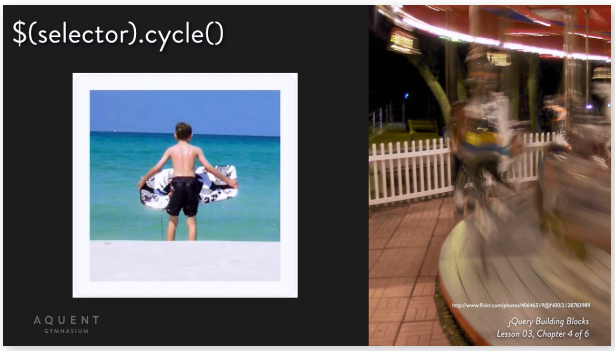
The first is the effects browser. So pop that open in a separate window. This is just going to show us all the different ways we can transition. The second thing that you want to have open is the reference page. Now any complicated plug-in worth its salt is going to have a nice set of documentation for you to go take a look at and this one is no exception. It's actually got quite a lot of options. We're going to touch on a few of these, and it's got some command strings in comparison as well.



So let's take a look just very briefly at the cycle API. Now cycle is fantastic because it's incredibly simple to use, but it has an incredible depth of options. So that's the best kind of abstraction you can ask for, something that's really simple at the surface, and lets you dive as deep as you want to go. We'll be diving pretty deep today.

There's three ways that you can use cycle, and they all start with this method `call, cycle`. It's just one method that cycle as a plug-in adds. The first way is you just call it like this. Theres a scene default that just puts together a slide show for you out of all of the child elements of whatever element the selector selects and turns it into a nice cross fading slide show.

Now the second thing you can do is pass in options. That's going to be a JavaScript object full of options and we'll go into those. This is going to allow you to specify details of how the slide show should build itself, the transitions, timing, all kinds of stuff about it. Thirdly, once you've got a cycle slide show going, you can pass in commands. And those are commands like pause, resume, destroy, and so on. So with that in mind, let's go into chapter five and actually have a go at building these things.

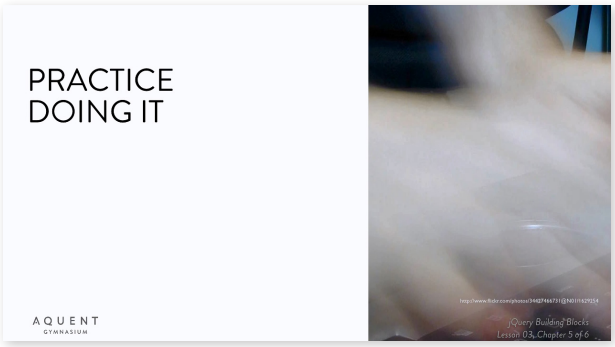


CHAPTER 5: PRACTICE DOING IT

In Chapter 5, we're going to actually do it. You could open your lesson materials and go into the Portfolio folder. And we're going to play around in here for a little while. Now our goal in this section is to get jQuery Cycle loaded into a page and get it to kick off a simple standard photo slideshow.

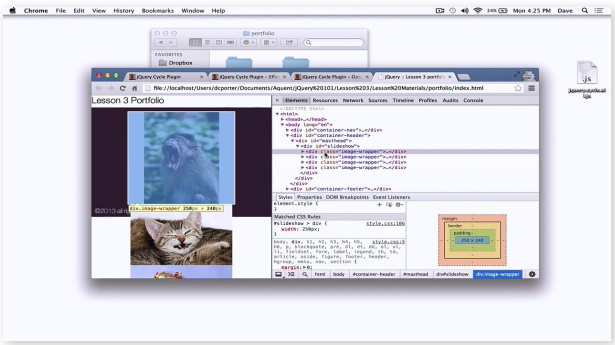
So first, let's take a look at where we're at now. Go ahead and open index.html in your favorite web browser. So this is from Lesson 1. I've updated some things, but this is from Lesson 1. I've

got some throwback pictures in there. So these aren't a slideshow yet. These are just kind of stacked, haphazardly at that, on top of each other.



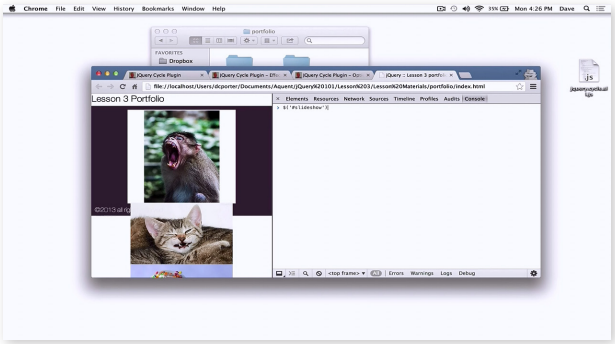
So first, let's see where we're at. The way that you figure that out is to take a look at the Chrome Inspector. Now I really want to lean really heavily on the idea that this Chrome Inspector is the best thing for your productivity. The ability to look at your code as it runs is a bit difficult to explain the benefit of, but in situations where I can do that, I work twice to three times faster and better than in situations where I have to save and go back again.

So in order to get into the Chrome Inspector, right click and Inspect Element. Now this brings up a whole bunch of options, but one it's going to bring up on top is a live look at your DOM, live look at your HTML. And this is all completely live.



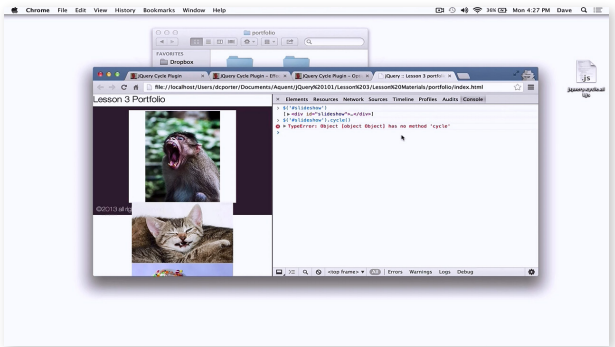
You can go and you can delete a node but we don't want to do that. So let's reload. You can take a look at all the IDs, all the classes are all there, along down here with all the styles that are applied as well. But we're not going to be getting into that right now. But I want you know that it's there. So that's fantastic.

Here is the slideshow div. This is the one that we want to turn into a slideshow. Inside of it, we've got these image wrappers. These are very simple image wrappers for centering CSS. Now there's a whole bunch of things up here. Elements is really useful and then console is the other one that's extremely useful. Console is where jQuery lives. This is a look into your page's JavaScript inner world and into its inner mind.



We can verify that we already have jQuery in here. Now let's go ahead and select that slideshow element using the syntax that we learned previously. There it is. It's properly selected. So we should be golden.

Now Cycle isn't part of the core jQuery. So right now, if we try to kick off a slideshow, we're going to get an error. This is the same error that we saw earlier in the day when we tried to raptorize things without having raptors available.



So our first goal is to get the jQuery Cycle plug-in, which we've downloaded over here, into our page. Go down to our Index.HTML code and right click on it and open with Sublime Text. Everyone's favorite IDE, or whatever you're using.

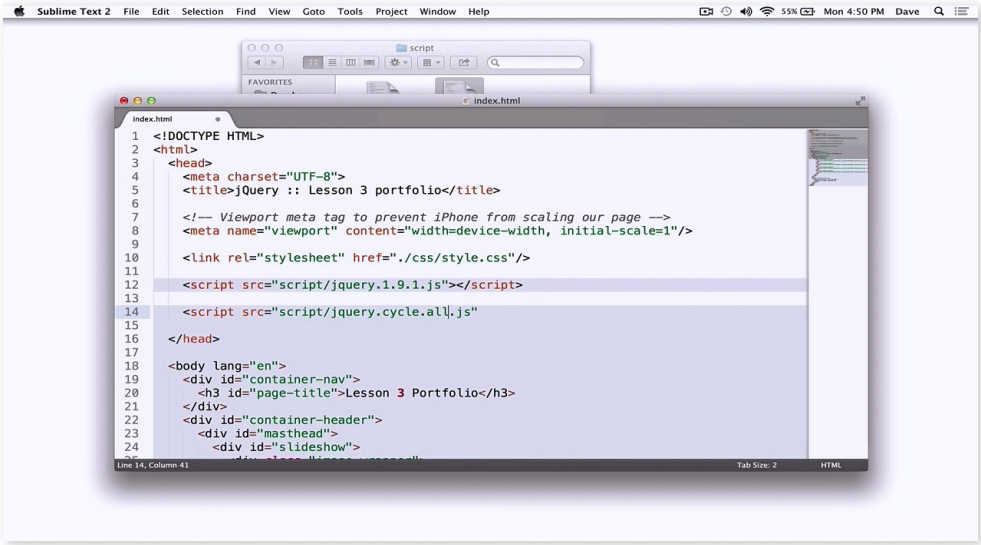
Now let's jump in here. Take a look. We are successfully importing jQuery, this particular version. That's nice to know. And it's coming from a folder called script.

Now just by looking at this, also by virtue of the fact that I wrote this, I assume that that means

that there's a folder here called script and inside it we're going to find our jQuery code. And there it is. And boy, is there a lot of it.

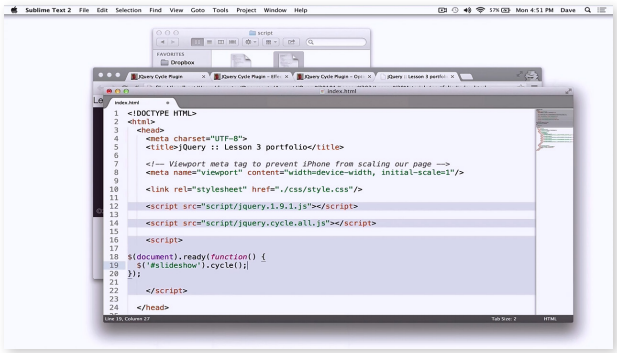
So with that there, what we want to do is get this guy over here. Super simple, drag it over. Piece of cake.

Now our Cycle plug-in code is ready. It's accessible to the page. It's ready to come in, but it's not in the page yet. For that, we need to give it its own <script> tag. And that's going to be `jquery.cycle.all.js`. This is a fairly standard jQuery plug-in nomenclature for the file names. You've got jQuery itself, name of the plug-in, some modifier. `All` in this case refers to the fact that you can get a slimmed down version of this if you prefer. Sometimes you'll see this would say `min` and that's going to be an indicator that it's been minified so it's smaller. It's hell to read at that point, but it's smaller and it will go over the wire faster.



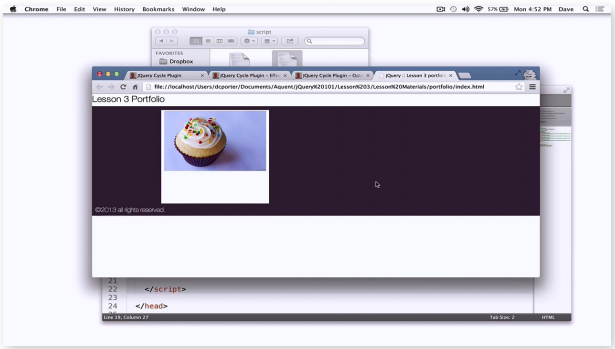
So let's go ahead and close this guy off. Give it a save and reload. And here we are, same place as before, nothing has changed except that when we do this, make that method call and hit Enter, this time it works. So our plug-in has successfully loaded into the page. And over here in the console, we now have a successful slideshow set up.

Of course, you don't want the user to have to jump into the console and tweak your page, so let's go ahead and move this code into the page itself. We'll give ourselves a nice inline script tag. Make it nice and big so that our code is nice and visible at all times because we're good developers to our friends. Like last time, let's go ahead and put this into a `document.ready` call. If you don't remember why this is happening, give it a pause and jump over to last week's quiz and brush up.

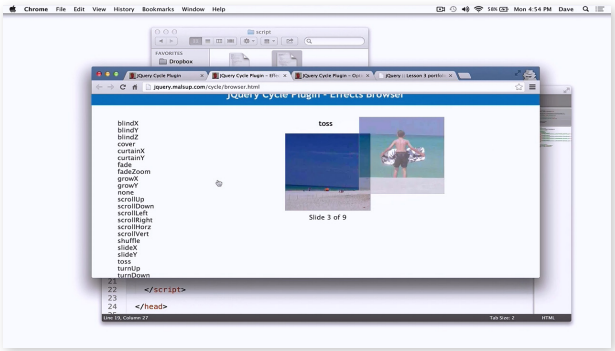


So within this, let's go ahead and do what we were doing before, which is select the slideshow and then Cycle it. Give that a Save. Give it a Reload. And without us doing anything over here in the console—in fact, let's just shut that guy down—we now have a nice happy slideshow. Nice little cross fade effects going on.

Now of course, that doesn't mean that we're done with the console. So let's pop that guy back open. This is a pretty nice slideshow, but Cycle can do so much more. Let's take a look back over at the main Cycle page. And we've got this set of effects going on here. And these are pretty, but there's a bunch more. And so if we go over here to the Plug-in Directory, the list of effects, which, again, if you've lost that, you can find it from this page under Browse Effects.



Let's take a look at some of these. We've got Blind X, that's pretty cool. Blind Y, predictably goes in the up and down direction. Curtains. I guess that kind of looks like a curtain. That's fun. And we've got Fade, right? Fade is the normal one. It just cross fades between the two.



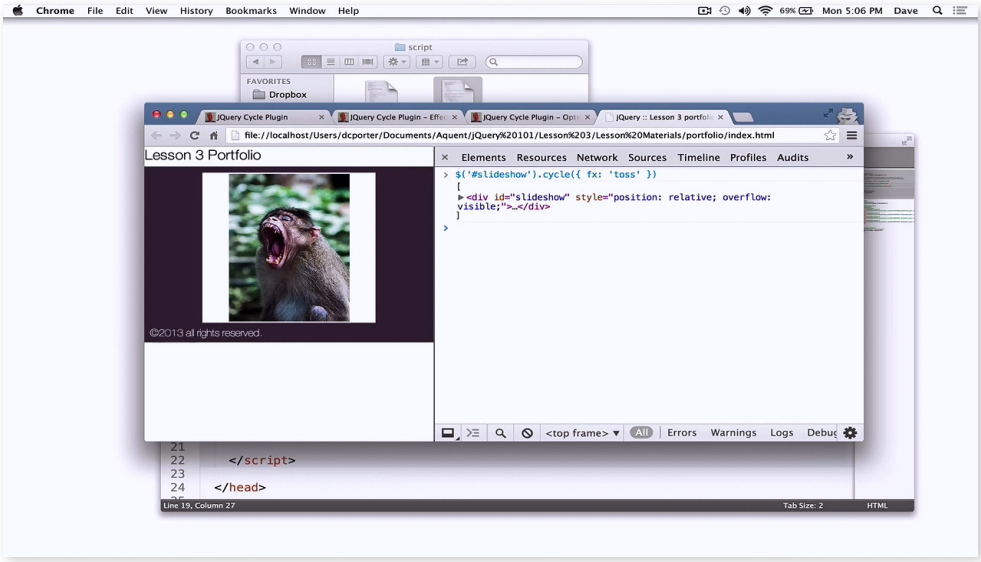
And then we've got something called Fade Zoom. And this is a lot of fun. I like this guy a lot. Unfortunately, it doesn't work very well with pictures with different heights which is what we've got. So we're going to move quickly on past Fade Zoom.

We've got None. This one is super boring. Then we've got Scroll Left, which is a real nice kind of scan as you go effect. Then two down from here we've got something called Scroll Horz, for one presumes horizontal. And that looks to be doing exactly the same thing. We'll come back to this later because it's not.

Finally, let's give Toss a try. This sounds like it might be fun. And it is. That's boat loads of fun. So we're going to take this bit of whimsy with us and we are going to tell Cycle that we want to have it use the Toss effect.

How do we do that? First, let's get our selector going. Then get our method call going, but don't hit Enter yet. This time, I'd like you to add an open and closed curly bracket. And that's on your keyboard above the square brackets over on the right.

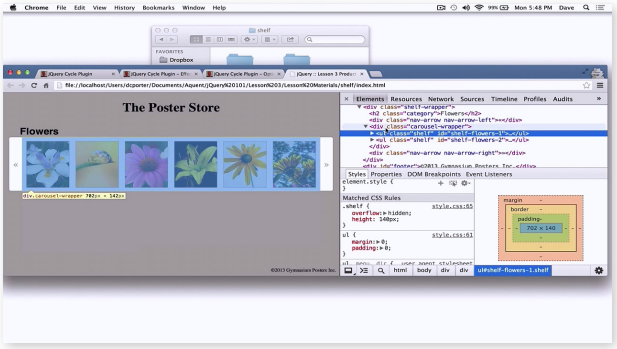
Now this, the thing we've just created is a JavaScript object. It's a data type. It's called a hash or a dictionary in other languages. It's all a fancy name for a bunch of key value pairs. It's a bunch of keys, which are just names for the thing, and the values, which is whatever that thing's value is.



So let's say in this case the key is going to FX, the type of effect that we want Cycle to use. And the value is going to be Toss. Let's hit Enter. We notice that slideshow resets itself. And there it goes. So that's fun. So that's it for the Portfolio page and for our introduction to Cycle.

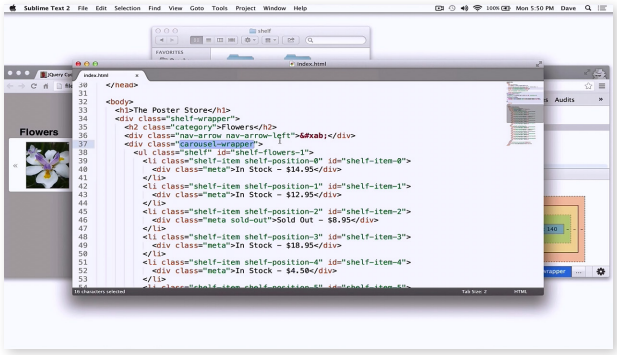
Again, just to summarize, we've added the plug-in directly into our page and we added it into our folder as well. And this is going to be the same process for any plug-in you want to drop in. And we've gone ahead and made some use of it.

Now we're going to turn our attention back to the shelf that we built last week. We're going to augment it with the ability to be a carousel and cycle between multiple shelves. Our goal is to get last week's shelf to cycle in a full user controlled way. So that's what it's going to look like. Let's take a look at what it looks like right now.



Go back to your lesson materials, and let's go into the Shelf folder. Pop this guy open. It actually looks about the same, but it's not doing anything. Let's take a quick look. You can just watch along for now. I'm going to inspect this guy. And take a look here. Now I've got two shells here. And one of them is hidden and hanging out below the other one. That's not where it's going to be eventually, but for now, that's where it's hanging out.

And just to prove it to you, let me notch up the height of this guy. And you can see, there's all our brand new pictures, some very nice ones. That's what this guy looks like on the surface right now. Let's take a look at the code. Right click on this. Open with your favorite code editor. And I've made some changes since last time. I'll walk you through them real briefly, but I took some of the grunt work out of this week's lesson.



First of all, I went ahead and dropped the Cycle plug-in right in. I, of course, added a whole new shelf with a bunch of new pictures that you just saw. I wrapped those shelves in a carousel wrapper element which wasn't there before, didn't need it before because there's only one shelf. Now there's multiple ones. And we need a Div to turn into our carousel.

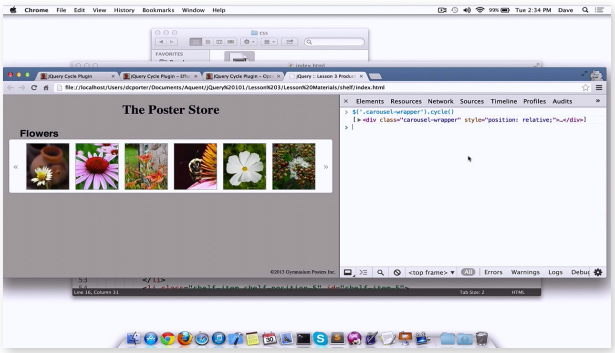
We won't actually be making any changes over in the CSS file today. But let me briefly run you through the changes I made there as well. Let me put in some simple styling for the nav arrow. I like using text here whenever I can. It makes for quicker and simpler designs. Put in a hover state, which is just fine. It's 2013, but that's all fine.

I'm positioning everything with some simple floats. It's quick and it's easy and it works as long as those width of things doesn't change too dramatically. There's the carousel wrapper from before. And I've separated out shelf item positioning code from the background image themselves.

That's just a good practice because we're going to have a number of items taking shelf position five, right? That's anything that's going to be over here. But each one of those items is going yet a different background. So that's going to be shelf item five itself, shelf item 11, and so on. That's going to be reused a lot. So we've split those out.

Long story short, we've got a bunch of new items with background images. Here are their original locations. Since we're good licensees, go ahead and jump on Flickr and tell these people how darn pretty their pictures are. And that's it. That's all the changes that we've made so far to date.

Now moving forward, our first goal is going to be to create a slideshow here, a cycle and then to control it from the command line, from the console. We aren't going to hook up the buttons yet. First, we want to make sure that we've got the correct commands. So let's jump over to the Console. Get our selector. And kick off a cycle. Let's see if this works first. And success. Great.

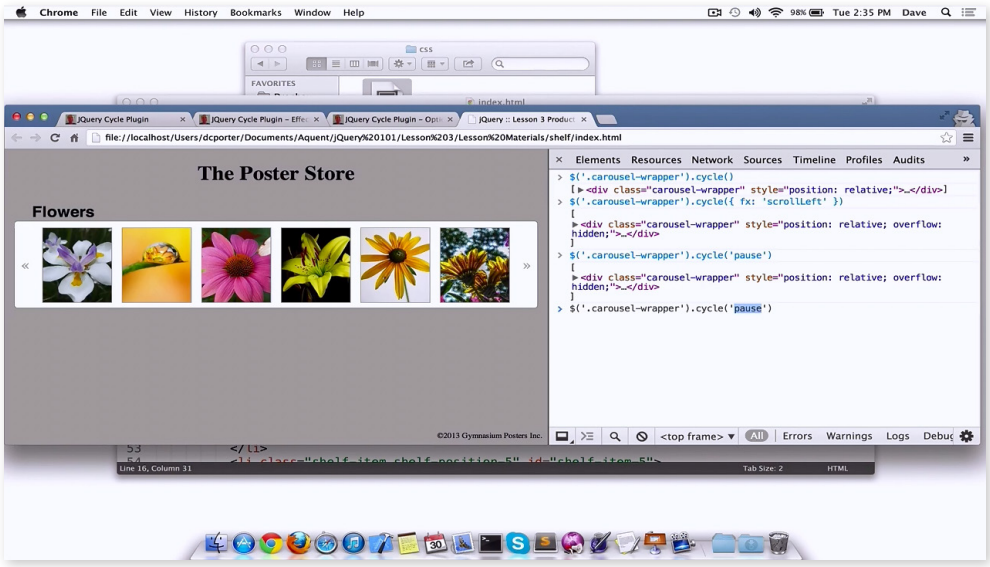


Now this cross fade thing isn't super useful. So next, let's go ahead and change the effect. And we do this, again, by passing in a value for the FX option. We saw before scroll left looks like it's got what we want. In a second, we'll see if that worked. And that works great. That's awesome.

Now earlier on I mentioned that there are three ways to use the Cycle method in this plug-in. We've used two of them. Right? We've got them both right here. We've created a default slideshow. And we've created a slideshow with options passed in.

Now we're going to use the third way and remember, up and down in the Console will get you to previous commands to save you some typing. And to save me some typing.

The third way to use this is to pass in a command. And you pass in commands by strings. So let's go ahead and pass the Pause command. Now this should pause the slideshow mid-run. And that's great. That's exactly what happened.

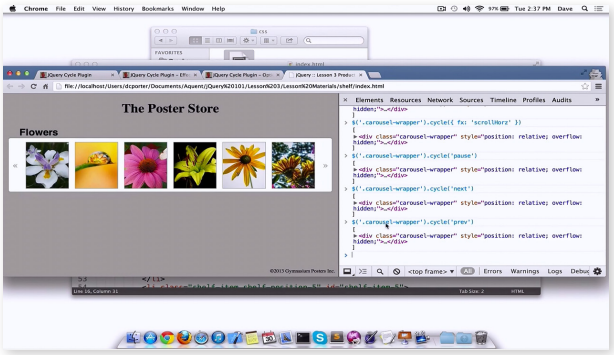


Now we can declaratively tell the slideshow to advance. Awesome. We can tell it to go backward as well. You might get tripped up here by thinking it's previous. It's not. Make sure that you check your reference from time to time. It's P-R-E-V. So we're going to go with that.

Also, something doesn't quite look right there. Let's go Next again. And then go Previous. And it's going to the correct shelf. There's only two of them, but it's going in the wrong direction. It's advancing to the left both times which actually makes sense because if you recall, the effect setting that we passed in was Scroll Left.

Now I spent some time looking at this Cycle plug-in page. And I ended up spending some time on the internet as well. It turns out that the Scroll Left isn't what we want. It's Scroll H-O-R-Z, presumably for horizontal. They built this to handle the situation where you want to go different directions based on whether you're advancing or regressing.

So let's do this again. I'm going to hit the Up button several times until I get back to this guy. And replace Scroll Left with Scroll H-O-R-Z. Since it's a new effect, the Pause command from before has been wiped out. So let's go ahead and Pause it again.



Now let's go Next. That looks good. And let's go Previous. And there we go. That's exactly the behavior we're looking for. Wonderful.

So let's move this out of the Console and move it into our code to make a little more permanent. From last week, we had a document dot ready method all set up. So that's ready to go. And let's just add to it.

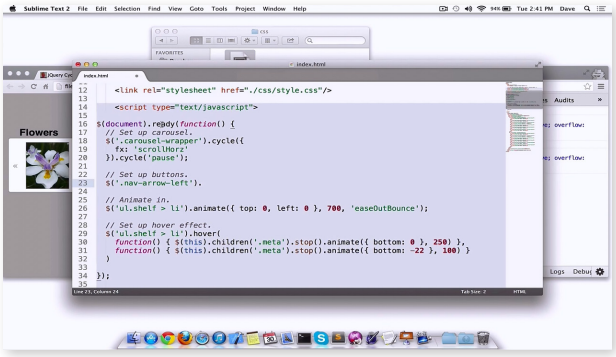
Set up carousel. Always document your code. Let's get our selector. So a nice auto-complete there. Let's create a cycle passing an FX option. And we're going with Scroll H-O-R-Z.

Now again, remember, we don't want this guy to automatically advance without the user's input. We've decided that it's a smarter design to not have that happen. So at the very beginning here, we can do some jQuery chaining and pass in the Pause command. Now this ought to prevent it from moving forward. And in a second, we'll see that it does. Wonderful.

Let's just verify that this can still advance and regress for us from the command line. Carousel-wrapper.Cycle. And we're passing in the next command. There it goes. Wonderful and the previous command. Beautiful. So there we have, at least from the Console, a functional cycle behaving the way we want to.

Now our next goal is to take these commands that we've just demonstrated from the console and actually hook them up to the button so that the user can do them. So let's jump back into the code. Remember, always comment your code. Set up a button. So now there's two buttons. If you look down in the HTML, we've got this guy and the correct selector is going to be like class, NAV arrow left.

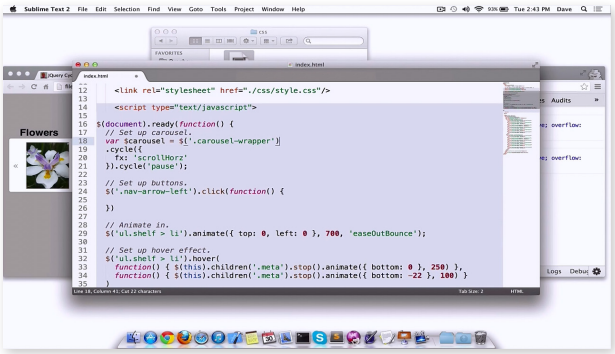
So let's go ahead and get that in there. Now the way that you set up the Event Handler up here to handle the ready event of the document is going to be the same exact concept as the way that you hook up the click event. We've got the element. We're going to tell it what event we want to handle. Just call `click`.



There's a number of these. These are effectively helper methods to make it easier for you to set up common event handling. There's `ready`. There's `click`. There's `mousedown`, etc. and then there's some more generic ones that if you need to handle an event that is done. Otherwise that doesn't have a helper, you can use that. We'll be going more into that in a future lesson.

Now inside here, when this element gets clicked, this function will be run. Now all we want to do is have it get this guy and send in a command to cycle one way or the other. Now we

could just Copy/Paste this in, but Copy/Paste is bad. So in order to give us a little more efficiency, we're going to create a new variable out here and we're going to move this guy into here. Again, the goal here is just to have a persistent variable so that we can access it from in here. Since we know we're going to be getting this guy multiple times, no reason to not go ahead and create a variable. Now by general agreement, you can start a jQuery variable with a dollar sign. That's just a nice convention.



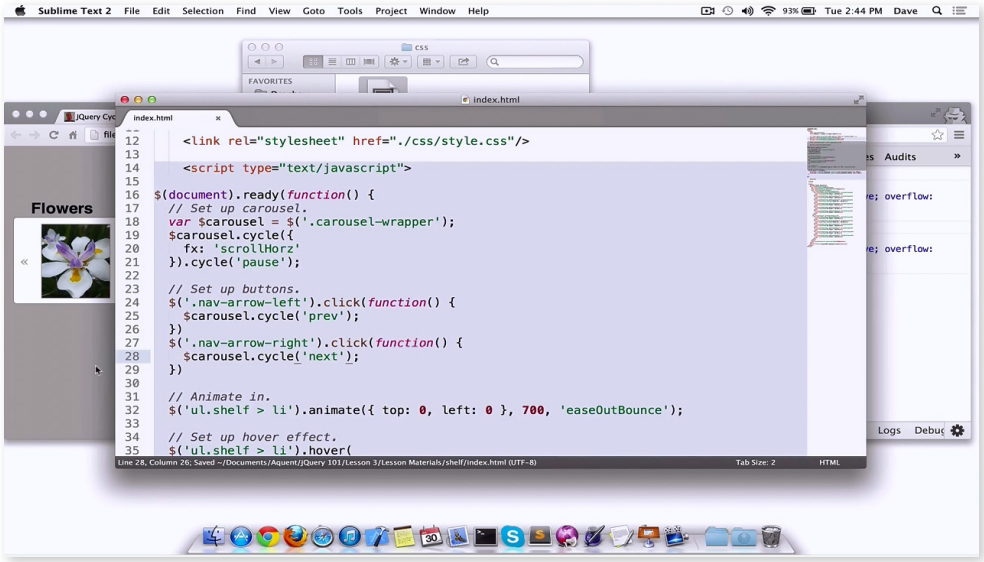
So now that we've got this, let's go ahead and replace that call with a new variable. And then from in here, we can do exactly the same thing `carousel.cycle`. Let's see, this is the left button so that's the previous direction. And now Copy/Paste is your friend. We are going to set up the right button to go next.

Let's take a look at how that works out. We've got this button going this way and we've got this button going this way. That is stupendous. Before we move on, I want to show you one little thing that I noticed go by quickly. I'm going to reload the page. I'm going to click the left button first.

Watch very closely over here on the right. Watch what happens to these guys when I click the left button. They kind of disappear and that's not really what we're going for. But if you look at any other time, and in fact, if you look after the first time we click it here, everything just eases in and out which is exactly what we're looking for.

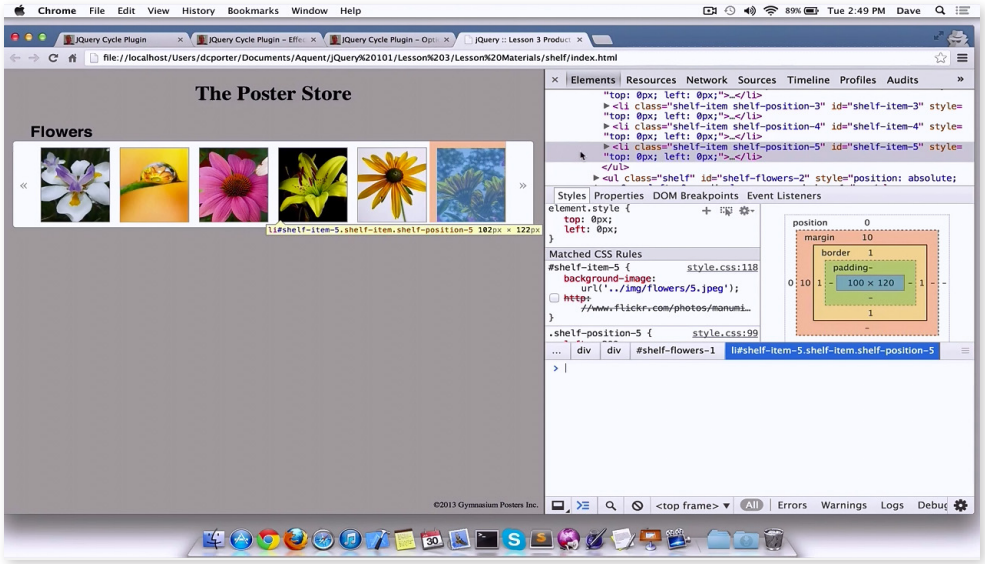
Let's take a look at that again. I'm going to refresh the page. And I'll hit Back. Now those are disappearing. That's not good. That's not what we're looking for. You don't want your boss to see that. He'll think it's sloppy. You don't want users to see it once it's in production because they'll think you're sloppy. It's bad news all around.

So let's do some quick debugging on it. Refresh again. No matter what your debugging, it's an advanced skill set. There's a lot of instinct that goes into it. There's a lot of knowing all the different places that you can look. So we're going to do a very abbreviated debug. Now I'm going to expand this guy a little bit to give us some breathing room later on.



Then I'm going to take another close look at this. As these guys ease out, they disappear. And that looks to me like a positioning thing. That looks like the sort of thing that happens in HTML when an element is too narrow. Its child element sort of get knocked down to the next row.

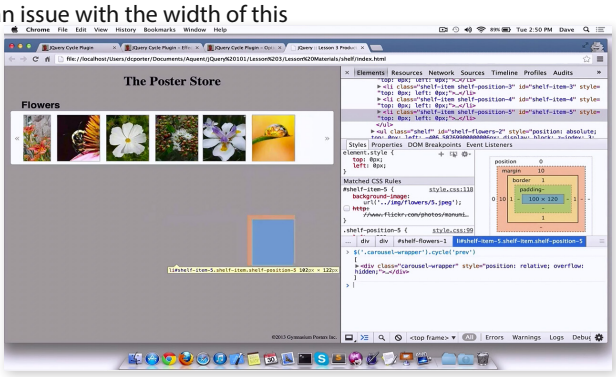
Now if that's the case, we ought to be able to see that happening in here. So let's right click Inspect Element. We're going to need to be able to do two things at once. We're going to need to be able to hold our mouse over it so that Chrome gives us this great highlight which works no matter whether the thing is visible on the screen or not. If it's there, it will highlight it for you. And we need to regress the cycle once.



So I'm going to go down to the Console down here. This guy, by the way, appears by hitting escape. You can get both a Console and an inspector at the same time, which in cases like this is crucial.

Now I'm going to hit the Up button down here. That's going to give me the command that I need to execute. Then I'm going to mouse over up here. And now I'm going to hit Enter. And watch carefully what happens to that orange box. You see that? It kind of went off and then it ended up down here. We can do that again.

First I'm going to go down here and mouse over this guy. Then I'm going to hit Enter. See how that goes away? So this is definitely a spacing thing. This is definitely an issue with the width of this guy not being set the first time.



Now if I check it out in the Inspector, I see that there is no width or height or anything set on this. But if I do this again, suddenly I see that the width and height have popped in. So something's going on in Cycle. And this could be a bug, in which case there are bugs in any code which means that there's going to be bugs in code that you use. And that might be something we have to work around. Or it could be a feature that we're using wrong.

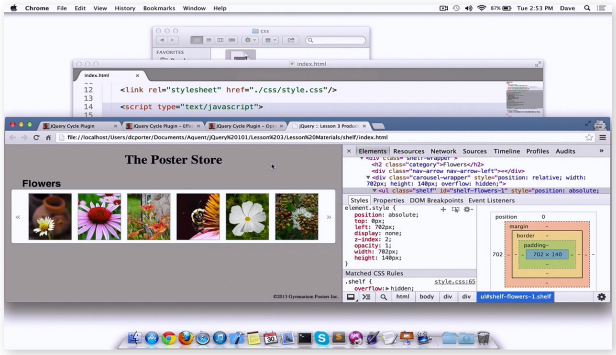
So let's take a look at the references. Now I'm going to scan down here. And looking down this list, I notice something. We've got height here. Now my guess is we've got width down here. Let's have a go at setting the height and width the first time around. So we'll jump into our code and along with FX, let's pass in

some new options—height and width.

Now for the height, we've got our `carousel` variable already. So let's go ahead and get the height out of it. The reason we're not hard coding this is because the number of places that any one value in your code are declared, you want those to be as few as possible. So if it changes, someone has to remember to go in and change it here too.

If instead we use this thing where we're going to go ahead and calculate it every single time if we go ahead and do that every single time, then any time the size of the carousel changes because there's been a design change or something, nobody even has to know this exists. It just works and that's what we're looking for.

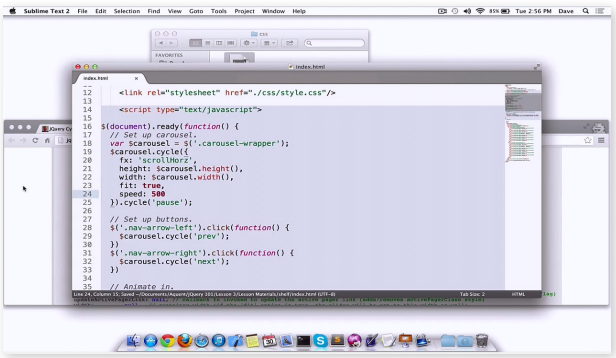
Now let's go back here and let's do a refresh and let's see if we fixed it. No, we haven't. But I bet we're on the right track. Let's go back to the Options list and scanning through it again, I notice that there's this odd thing. It says Fit. Force slides to fit container. Now that's exactly what we're looking for it to do.



So let's add one last option to our code. Let's turn Fit to True. And if we go back here and we Refresh, beautiful. I'll show you that one more time. Refresh. And now it slides off exactly the way it's supposed to.

So a quick introduction to debugging. It's really hard. It's really weird. You never know what piece of thing you're going to have to know ahead of time. But a quick introduction to how to debug a positional issue in HTML.

Now there's just one more thing I want to show you today. When we click these around, they come into view, but they roll in pretty slowly. Let me refresh this. That takes a while. That's I think over half a second when those roll in the first time. Slow transitions are fine when the user hasn't kicked it off. If they're saying, I want to see the next set, you should get it to them quickly. If the user has to sit around and wait, that's an unhappy user.



The user starts to notice that they're waiting around about a tenth of a second. And they're OK with that for a little while, maybe up to, well, probably half a second at the most. If they're super entertained by the transition, you may be able to get them around longer. But you want to try and keep that as slow as possible.

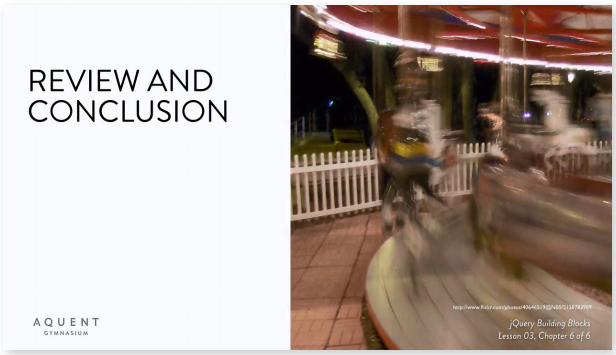
So we want to try and speed up this transition, taking a look back at our options reference, I see something here called speed. I bet that's what we're looking for. So I'm going to add one last option here, let's set the speed to 500 and see how it looks. If I refresh, that's much faster.

We might even be able to speed that up. Lets drop that down to 400. Keep in mind these are milliseconds. Almost everywhere in programming anytime you have a duration that's just a number, it's going to milliseconds. All right. So down to 400 milliseconds, 4/10 of a second. Let's see how it looks. That's just right on the balance of looking good but still being fast enough that you're not annoying users.

With that, we've got our code. We've successfully set up our carousel. And we've hooked up the buttons to go backwards and forwards whenever the user tells them to.

CHAPTER 6: REVIEW AND CONCLUSION

So let's review :starting off with the cycle commands. Passing in no command will simply create a default slide show. You can pass commands such as `next`, which goes to the next slide and `prev`, which goes to the previous one. You can pause, and you can resume at will.



Cycle options are passed in via your curly brackets. Those include `fx`, which determine which effect is used to transition between slides and speed, where you can control how quickly the transition goes. Remember, if the user is kicking it off, faster is better. If the user is not kicking it off, you can take your time.

`Height`, `width`, and `fit` get you to some bug fixes, and are also useful in other situations. Don't forget there's a whole bunch more. You've got to look at the full list. Next time you're developing a carousel, read through them and see what else is possible.

Here was the final cycle slide show command that we came up with. `ScrollHorz` gets you going forward and backward as needed. `Fit`, `height`, and `width` work around that little cycle that we discovered and `speed` speeds it up to a speed where your user isn't going to be annoyed.

So in conclusion, today we talked about the vast ecosystem of jQuery plug-ins, the idea that no matter what it is you need to do, there's a very good chance that jQuery is going to be able to help you out or that someone has created a plug-in. We talk about what plug-ins do. Generally speaking they give you new methods to call on the jQuery selected object.

And we talked specifically about carousels, how to do them well, how to do them badly. Don't forget doing them badly is very easy. We talked about cycle, the jQuery plug-in for doing great easy declarative carousels. And we put one together ourselves. There is some homework. Feel free to use the pause button throughout. Firstly, please complete the short quiz that you find in the lesson materials. Secondly, I'd like you to go back to the slide show that we created towards the middle of today's class, and I'd like you to combine last week's jQuery hover events with this week's cycle to pause and resume our photo slide show when the user hovers and then leaves the slide show. Bonus points by the way if you can figure out how to do that directly within cycle without using jQuery's hover mode at all.

If you run into any problems, have any questions, or just want to say hi, please hop on the forum, and we'll try to help you out. Otherwise, we will see you next lesson. This has been Lesson 3, the jQuery Carousel(s).