AQUENT

GYMNASIUM

JQUERY BUILDING BLOCKS:

THAT CERTAIN SPARKLE

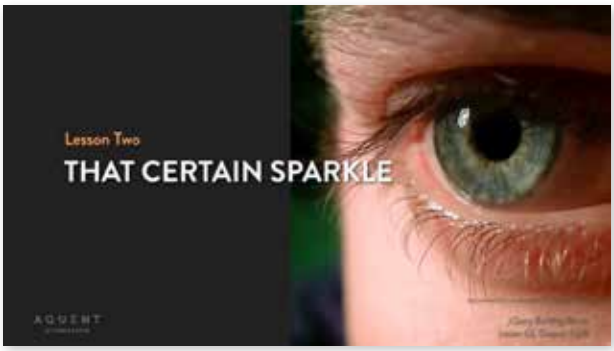
LESSON 2

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: THAT CERTAIN SPARKLE



Welcome to jQuery Building Blocks: Five Techniques to Cut Your Web Development Time in Half. Lesson Two—That Certain Sparkle.

My name is Dave Porter. That's me in the foreground. In Chapter 1 I will be introducing you to, among other things, Chapter 1. In Chapter 2, we'll briefly discuss what benefits animation brings to the table. Why you'd bother with it. In Chapter 3, we'll take a look at some examples of animation and software, with a closer look at animation on the web in Chapter 4. In Chapter 5, I'll show you how to add some animation to a simple e-commerce page. And then in Chapter 6, I'll wrap it up with a quiz, some assignments, and a solemn goodbye.

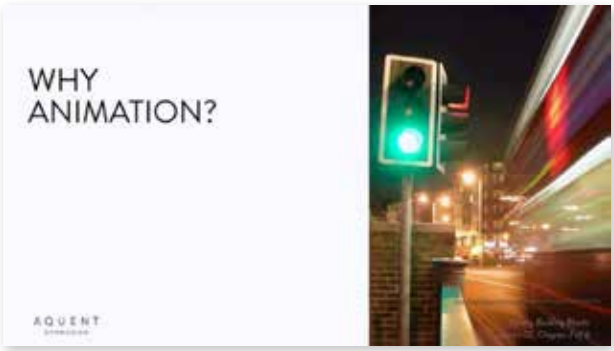
Don't forget, as we're going through always feel free to use the pause button to go take a drink or whatever. While we're in the middle of coding you should use it to catch up anytime you like.

I'd like to briefly review software requirements for this course. First of all, you will need a browser with a JavaScript debugger that you're familiar with. If you're not familiar with any particular one then please make sure you have Google Chrome. It comes with a great JavaScript debugger built in and I'll be showing you how to use it.

And starting this lesson, and for all subsequent lessons, please make sure that you have an IDE that you're familiar with. Again, that's an integrated development environment. It's like a fancy text editor for your code. I will be using and showing you how to use Sublime Text 2. I highly recommend it. It's speedy. It's really nice. Everyone I know uses it right now. It's the new hotness. If you have something you're more familiar with, feel free to use it.

And so without further ado, Lesson Two—That Certain Sparkle.

CHAPTER 2: BENEFITS OF ANIMATION



What does animation do for you that static transitions, blinking between states and an application, doesn't do?

jQuery is going to make animation super easy. But even so, why are you going to go to this effort? What benefit does it bring to you?

I'm going to start off the lesson by taking a quick detour into the entirely unrelated topic of film theory. The eye light is a name for a lighting

technique that is going to show up in every film with any kind of budget. You're never going to be able to not see it again now that I've shown it to you.

So along with standard lighting in a scene, any lighting technician that's worth his weight in sawdust is going to set up this additional light, whose only job is to reflect off the eyeballs of the actors. Now through some hurried Photoshop I've removed the eye light from this lesson's title photo. And here it is back.

The eye light, it's a simple but completely indispensable technique that lends human eyes a sense of depth and vitality really. Without it, actors look flat and weirdly lifeless. And with it, they look alive. Works for cats too.

Now this is important in photographs, but it's absolutely crucial in film. And like I said, you're going to see this in every single film that you go and watch after this. If you've got half a second now, go watch a trailer at trailers.apple.com or something and have a look for it there.

So let's take a look at somewhere where this animation adds a life light to an interface. And Pinterest, luckily for me making this video, has two active different interfaces going right now. The first one is pretty straightforward. If you click the Pin It button a model pops up. And then if you click x it goes away. This makes sense. It's straightforward. Appears and disappears. It relies on the user to reorient themselves, which isn't that bad because this is a pretty straightforward orientation question. And it feels a bit stark. Whatever.

Here is their other user interface, which I believe is their new user interface. I certainly hope so because it's much nicer. If you click that Pin It button, the whole photograph spins up into place. It maintains the user's orientation throughout the process. And it feels alive.

So animation brings software to life. What else does it do? Affordance is a fancy name for how things—and that includes imaginary things in software—how they communicate their function to us. For example, cup handles are for picking up. And door handles are for pulling, right? And no matter how classic or oddly postmodern a door is, it tells you very clearly how to open it. In fact, affordance's promise of clear communication is so ingrained in our expectations of the world that we have a special fascination with doors that break it.

Affordance can be trained too, right? Buttons on elevators, at least to my eye, they look pushable because we've pushed them a million times. There's nothing inherently pushable about them. But we know about them from experience.

So when the iPhone came out and it needed to speak an existing language so anyone had any idea how to do anything, it just made its buttons look pushable. Of course, having trained us effectively, it's now going in a completely different direction with a much more economical interface. But that's a talk for another lesson.

CHAPTER 3: SOFTWARE IN MOTION

We'll take a look at some examples of animation and software. Using animation makes software come alive and communicate its purpose. This is Maps. Here's what it does. You zoom in. You zoom in more. And then you drop a pin.

Now, it's straightforward. We understand how this works. But functionally here's all that's happening. Zoom in. Zoom in again. And pin goes down.

Now, here it is again with the animation. Just watch this. Watch how much nicer this is. Even with the loading and the shifting of stuff, the zoom and then the drop, they make so much more sense. The animated transitions afford this map a sense of continuity. A sense of space. A sense of sort of "thingness." And the pin's animation, again, draws your attention to its interactivity.

Here's the same product category. Different company. Very different animations. When I click it pings a little ping on the map, just so that I know that I've succeeded in action. And if you look in the upper left-hand corner, the navigation bar animates into place, very eye-catchingly. So you know that there's a thing there that you can go do now.

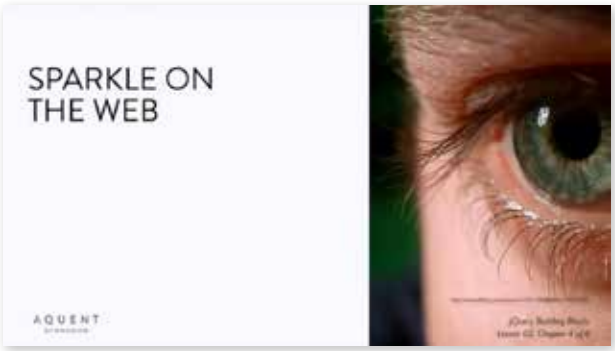
And then finally, from an entirely different vertical, this is Apple's take on uses of animation. Now this was on their store. Let me show that to you again. This is from their iPod site. And it uses animation to do two things.

It draws your attention to the things on the page that it thinks you ought to be spending the most time on. That's, of course, the things that it wants to sell you.

And it does it in such a way that it lets you know that these things are interactables, right? These are things that you can go and click on and learn more, and so on. So one more time there.

That's Apple's predictably tasteful animation.

CHAPTER 4: WEB ANIMATION



Now in theory, if you've got a little bit of JavaScript, then you know how to do this, right? You know how to put a thing in a place, you know how to kick off timers rapidly, right? Maybe set a timer for once every 60th of a second so that it can move it quickly. And then in theory this math is doable math that you can know where the stuff should go at every second. But you don't feel like doing that. I've done that in the past and it's no fun.

So instead, of course this being a jQuery lesson, we're going to use jQuery.animate to do all of our work for us. First I want to show you a couple more examples of this on the web. You can use Animate to pop in some neat notification messages, slide them in off the top as a kind of mental place in the user's mind for them to hide. And they go away again.

I like this effect. I'm not necessarily a huge fan of how it's being used here, but we're going to use it later on so I want to highlight it. When you mouse over this thing, this extra information pops up and that information waits until the user has sort of declared the intent to look deeper. And then at that point, it shows the extra information. Now I like that. If you've got limited space, you can hide information away. Or if you want kind of a pure presentation with more information available on request.

Now with great power comes great responsibility, right? There was an entire decade dedicated to finding all of the horrible, horrible ways to use animation on the web. And we did. We absolutely went down all those paths. We found all the really ugly horrible ways that you can over use animation. So you have to be careful. Use it sparingly. Use it where it's useful. And make sure that you're using it with a purpose.

Again, here's Apples take on the subject. And I think they've done a really good job. Now, they haven't done a subtle job. Good animation doesn't necessarily have to be subtle. They've done a fairly eye-catching job of drawing your attention to these things that need doing. And then letting you know that it's something that you can go do.

For the purposes of this lesson, let's say that you work or contract with a company called The Poster Store. Now they sell posters, as you'd expect, on the Internet. And they've got a pretty bland website. And it's not catching anyone's eyes. They

got maybe some nice transparency effects on the overlay information and I really like that red. I think that was well chosen, but it just sits there. And I will ignore the fact that the background is gray right now. That may or may not be a good thing. And maybe the fonts are a little boring, but today, we're going to focus on using animation to draw visitors into this site.

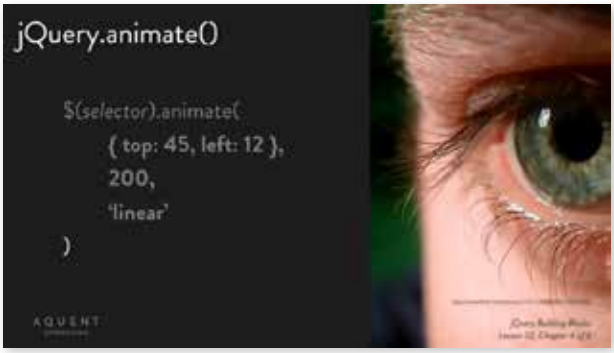
Now your boss or client has just been to Apple.com. And they've seen this. And they've had the very brilliant idea, of course, maybe it was your idea and you sort of snuck it to them, let them think it was their idea. But at any rate, they're now heavily sold on using this effect on their own website.

So we're going to take this static site which already exists in your lesson materials, and we are going to turn it into this. It's got a jump in effect just like Apple.com. And it's got this nifty mouseover effect as well, where this shelf really showcases the product. I mean, these are the things that you're here to buy. And if the user wants more information about them, they can mouse over them and find out about it then.

Again, we're changing this into this. And we're doing it with jQuery.animate. Now I'm going to run through jQuery.animate very briefly. And then, of course, we're going to go into more depth as we start to actually use it.

First thing you do is you call the jQuery function. You pass in a selector string. We went over this in the first lesson. And we'll be reviewing it again in this lesson. So for now, we're going to assume that you successfully selected your elements, the elements that you want to animate.

Once you've got that jQuery element back, you just call animate on it. Now the arguments that you pass in, there's three of them that we're interested in today. There's a whole bunch more that you can use to get very fancy with things. But for now, we're going to look at these three.



The first is the Properties. Now these are the things that you want to animate. Let's say that you want to animate whatever it is from wherever it is to top 45, left 12. And those are presumably pixels. You want to have it happen over 200 milliseconds. Now that's quick. If the user is actively interacting with something, quick interactions are better. And I'll go a little more into that theory later on. If they're not interacting with it yet, if you're drawing attention to it, then slow animations can be OK, not too slow, but not bad. But if the user is trying to see something or get somewhere, you want to be quick about it.

And finally, this third one is the timing function. You set that to linear if you like. That's the sort of ugly stiff animations. The default is something called swing, which is a nice easy kind of information. And we'll be looking at some other examples today as well.

And with that declarative command, you've now told jQuery what you want to have happen. And behind the scenes, jQuery's going to take care of a whole bunch of stuff that you just don't have to worry about.

CHAPTER 5: PRACTICE DOING IT

In Chapter 5, we're going to actually do it. Please go to your lesson materials and go into the Shelf folder. Now, since this website, lucky for you, already exists, there's a lot of resources already in here. So let's pop 'em open.

First thing, let's open it up in Chrome. Here is the website as it currently exists. And these are very pretty flowers. But again, they're all just sitting there. Now, let's take a look through these folders.

First, let's go into the Image folder. That's going to have the images themselves. These are lovely and sized for Retina displays. No big deal. Those will be downsized thank to our CSS on normal displays. And they are future proof—well, at least for the foreseeable future.

In the Script folder are a couple of files, just files we won't be working with, but we're going to need to include. The first is jQuery itself. Now, this is the readable version, which is much larger, and which you can read. Of course, you won't have to, but I wanted to include it to contrast it to this plug-in we're going to be using called easing, which is going to give us a few more fun transition timing functions.

This is what minified code looks like. It's horrible to read. You basically can't read it. And it's much, much, much, much smaller. So it loads faster. It goes over the wire faster, et cetera, et cetera. In your production website, you're probably going to want to use the minified version of jQuery.

But for today, we're just going to load the full one. Since this is coming right off your hard drive, we're not going to have much in the way of speed problems. Go back up a level. There's one more folder, the CSS folder, and there's only one file in there. That's where all the style is.

Now, on my machine at least, Sublime Text 2 is the default thing to open CSS files with. But let's right-click on it anyway and go to open with. And select Sublime Text 2. And here it is, welcome to Sublime Text 2. This is a great editor. It's speedy, it's lightweight, and it's free for a trial period. I've upgraded because I use it all the time, and I love it. But if you're not using it, I recommend giving it a look.

Check out this awesome little mini version over on the side. And it's not just an aesthetic thing. This really helps your workflow when you can just glance over here and know where in the document you're at.

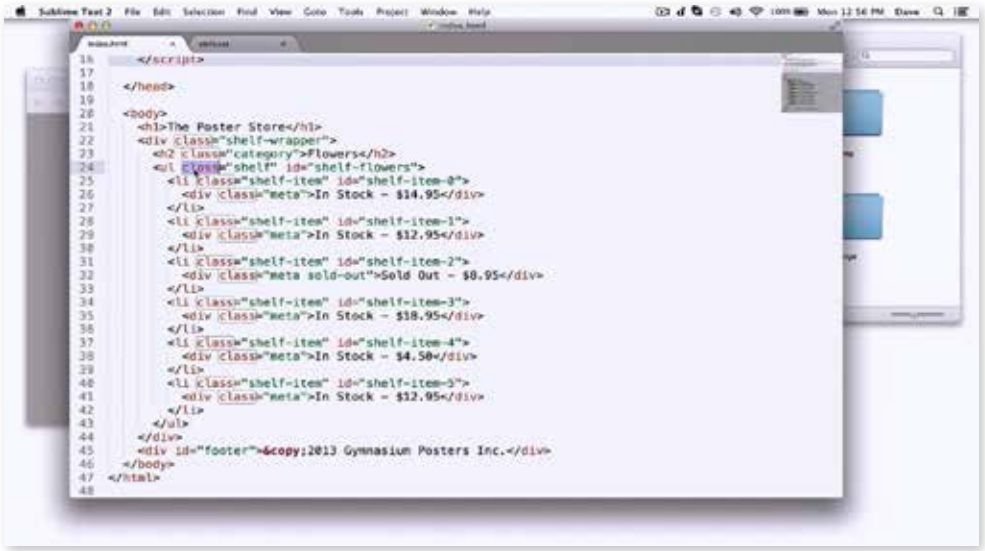
Let's go back up a level. And once again, let's open this with Sublime Text 2. This is index.html, so this is the actual thing. Now, for all the good things I have to say about Sublime Text 2, it is one of those perpetually in beta things, which is why crap like this is acceptable. But there we go. Now we've got everything in one window.

Open this guy up a little bit more. Now I would like to, first off, take a quick tour through the website as it exists now. We've got some straightforward HTML. Up in the header, we are declaring the title. We're importing our two scripts. That's jQuery itself and then the plug-in.

Quick tip here: if this doesn't start with a slash, then it's a relative path. If it does start with a slash, then it's an absolute path in terms of the server that it's on. It gets complicated. For now, since we're pulling it off your hard drive, no slashes are the way to go.

We're bringing in our style sheet, and I've sort of pre-loaded a little script area here for your code to go. And on the body itself, this is all very straightforward. We've got the header. We've got a wrapper for the shelves with a shelf header themselves, a nice little H2. And then even sideways lists ought to be done with the ul tag just to keep things semantically correct.

So we've got an unordered list with a bunch of list items, one for each shelf item, and these have all got classes and appropriate IDs on them. So we'll be able to address them in the CSS.



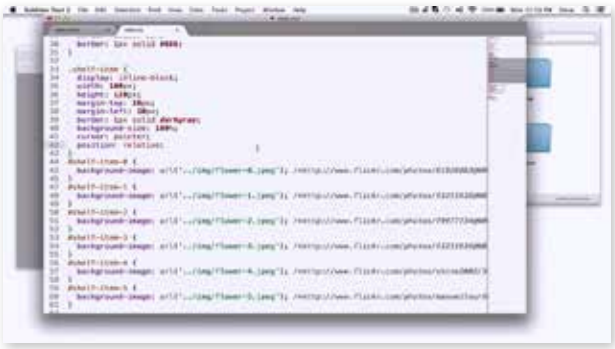
```
    </script>

  </head>

  <body>
    <h1>The Poster Store</h1>
    <div class="shelf-wrapper">
      <h2 class="category">Flowers</h2>
      <ul class="shelf" id="shelf-flowers">
        <li class="shelf-item" id="shelf-item-0">
          <div class="meta">In Stock - $14.95</div>
        </li>
        <li class="shelf-item" id="shelf-item-1">
          <div class="meta">In Stock - $12.95</div>
        </li>
        <li class="shelf-item" id="shelf-item-2">
          <div class="meta sold-out">Sold Out - $8.95</div>
        </li>
        <li class="shelf-item" id="shelf-item-3">
          <div class="meta">In Stock - $18.95</div>
        </li>
        <li class="shelf-item" id="shelf-item-4">
          <div class="meta">In Stock - $4.50</div>
        </li>
        <li class="shelf-item" id="shelf-item-5">
          <div class="meta">In Stock - $12.95</div>
        </li>
      </ul>
    </div>
    <div id="footer">&copy;2013 Gymnasium Posters Inc.</div>
  </body>
</html>
```

Here we've got our sort of metadata, the extra data about each stuff. This guy is in stock, and it costs this much. This guy is sold out. It cost this much, et cetera, and so on, and so forth. Here's the footer itself, which we will be doing absolutely nothing with. Here's a little HTML-encoded copyright symbol, anything that's not directly on

your keyword, and some stuff that is on your keyboard, you may have to do fancy like this. But again, this is some basic HTML.

Transitioning over to the CSS, we've got some pretty standard CSS here. We're wrapping the shelf. We're styling our headers. The shelf itself has got some stuff to make it in the right place and then some stuff to make it look a little prettier. But we gave it a nice, modern border radius. And if people are coming to your website with IE8, you know, they'll see some square borders. Not a big deal.

Now the shelf items themselves we're displaying inline block. That's a quick and easy way to get them to sit next to each other, the way an image would. We're wrapping our images in list items. We're wrapping them in elements, so we have to tell the browser explicitly to use that inline block display type.



You know, we're sizing it, giving it a nice border, background size so that we're stretching our images out to fill up the whole area. Giving a pointer, and then we're positioning them relative. Now this does a couple of things.

The first one is it does some fancy layout stuff, which is beyond the scope of this lesson. But the upshot is that it allows absolutely positioned elements within it to be absolutely positioned relative to this guy. And I'll show you a bit more about that later on. The other thing it does is, unbeknownst to, obviously, the person who wrote this website before you, it's going to allow you to very easily move these elements around relative to where it is that they would naturally go on the flow, which means that it makes it very easy for you to animate them.

And again, the guy who wrote this, he didn't know that. Now down here, we've got our individual shelf items. You need to drop in the background images on each one of these. Since we're not doing it in HTML, we do it in CSS instead. And since we're nice good license observers, here are the links to each one of these pictures. You should hop on Flickr and tell these people how wonderful their pictures of flowers are here.

Now the numbers start at zero, you'll notice. I did that on purpose, and not to be mean, although that's plausible. I did it because, in programming, the deeper you get into JavaScript, the more areas you're going to run into where counting starts at zero, which makes sense. If you've got a concept where there really is no emptiness, emptiness is represented some other way.

For example, in an array, being empty is represented in a different way than zero. So if the genius folks who invented computer programming back in the 1960s or whatever decided that when you're counting things where emptiness is represented some other way, you start at zero just to save a bit here and there. And here we are, in 2014, we're still doing it. So just to sort of start to get you used to this thing called zero-based counting, where counting starts at zero. The zeroth item is the first one,

something that it's a good idea to start wrapping your mind around if you haven't already.

Down a bit more, here are those absolutely positioned items we mentioned. The metadata divs contain the sort of meta information, the price and the availability. And we are positioning them absolutely relative to the relative positioned item above it. And let's take a quick look at that.

Here's this now. I'm going to dive quick into the inspector to actually inspect something for once. So I'm going to right-click on this, going to inspect element. And I'm going to change its parent's position. Remember, I said relative positioning? I'm going to change this to the default. So I'm going to remove the relative positioning.

Now you'll notice that these guys are, all of a sudden, they're down here. And the reason they are all of a sudden down here is because they are absolutely positioned relative to the first thing which doesn't have the default positioning. So by giving this position relative, we create a context for these metadata guys to be absolutely positioned. Now that's a long way around to say that by making this position relative, we can very easily drop these guys in here and have them stay where we want them.

So that's position relative and position absolute within it. Position absolute is great because you just sort of tell it where you want it to be. Over in my day job of web application development, position absolute is the way to go. It's the way everything sort of is and ought to be done. But here in the world of web documents, you're making a lot of use of a lot of very useful flows, and floats, and things. So there's nothing wrong with that, but here we are in my comfort zone.

So this is a series of stylings for the metadata. Then we do the cursor pointer just to sort of get that down another level. And then we get our nice background colors. This is the green one, RGBA. It's got a nice little 70% opacity. And here we are with the red RGBA again. Nice kind of dull red. And again, here's the Footer, which we will be ignoring entirely.

So that's our website as it exists today. It's nice. It's not bad. But there's no pop to it. And your boss, whether through his own genius, or through what he thinks is his own genius, has decided that he wants this to adopt some of the pop of the Apple website. So let's get started.

Now, the first thing that we need to do is we need to hook in and tell jQuery to do something once the document is loaded. The reason for that is that HTML documents load and parse sequentially, Line 1 on down. So if our script is running here, this stuff won't exist yet. The body won't have been parased. It won't have been run. This stuff just won't exist yet. And so anything we do to it will be fired off against an empty document, and that's bad news.

Now, I mentioned in the first lesson that there are all kinds of nasty quirks with figuring out when your document is loaded. You can move your script to the bottom if you want. That works most of the time, not always. It's not great just for semantics because there's no technical reason to not have it in the body, but it's nice to keep your code localized in the head because that's sort of where it belongs.

Now jQuery, luckily, gives us a very easy way to fire off code when the document is ready. And the way that we're going to do this is type jQuery, the dollar sign, and then we're calling it as a function. And we're passing in the actual document itself. There's no reason you couldn't pass this in as a string, but the document exists, so we're going to pass it in.

Now this generates a jQuery array, a jQuery list of elements, which, in this case, is only going to include the document. Now we're going to do document.ready. This hooks up the ready event to the document element via jQuery, via all of jQuery's subliminal machinations, so that you don't have to worry about all the things that could go wrong.

And this is an event handler. We're going to go away deeper into that later on. So you don't have to worry about it right now. But what we're doing here is we're hooking up an event handler to the document's ready event. And the thing that's going to handle it is, of course, a function.

So let's do that. And, just to keep things pretty, let's put a semicolon at the end, since all of this is, in fact, just making a command. Now just to prove that this is actually firing something off of ready, let's do—you've probably seen this before. But let's not do an alert. They are annoying.

Instead, let's do a fun thing called console.log. Now what this does is this spits out text. Let's say, hello loaded, exclamation point because we're excited about it. Semicolon because we're proper coders. And we're going to go back over here. Now this is the console. If you want something to spit out into this area from your actual code, you want console. log. Let's see if that works. There we go. Now, at load, we've successfully fired off some code. Now let's make that actually do something. Let's get rid of this guy. Now, the very first thing we're trying to do is we're trying to select these six items.

Now if we inspect them, we see that they are elements of class shelf-item, and the type of element is li, which stands, again, for list item. Now, in jQuery, remember

your selectors from last time, we are going to select these items—dollar sign, parentheses and then, in a string, since we're selecting li, we are going to say just li. And that selects all of the list item elements in the document.

Now there might be some other ones, right? This might be a larger website where there's some other li's kicking around, probably in a navigation bar at the top somewhere that we're not seeing right now. So let's narrow it down a little bit. We've seen that these have class shelf-item. So let's do .shelf-item.

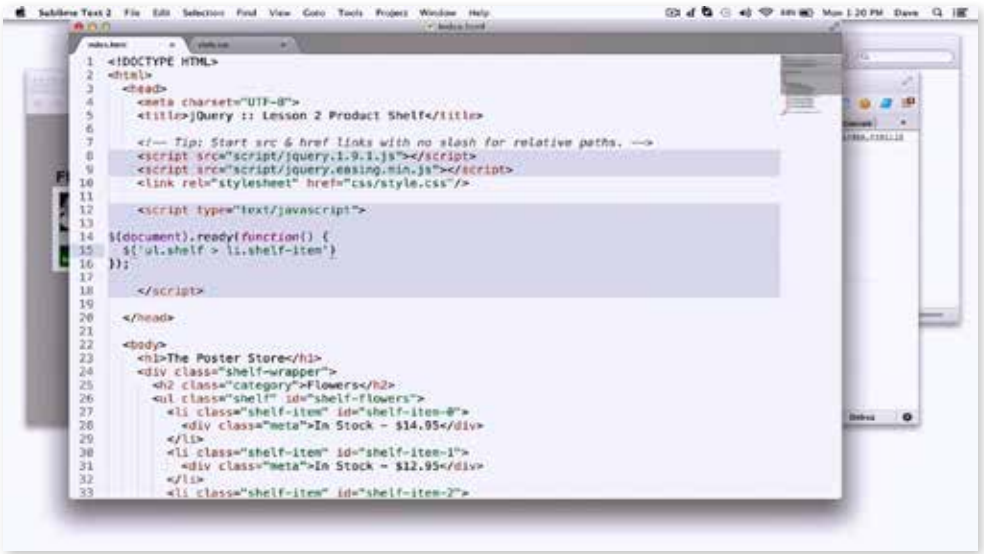
Now this guy, let's give it a look over here in the console. So let me go back here. If you don't have this guy open again, just to review briefly, again, you can always get back to it by right-clicking anywhere, clicking Inspect Element, that'll get you to this. And then the console is over here.

So here is our full list of li's. And we're going to be doing stuff to them later. Now since we're reviewing jQuery selectors, let me show you another way to get into this. Let's delete this guy back out, go back to our element map by clicking the elements tab up here, and let's get all of the children of this ul, class equals shelf.

So back to the console. Now, in order to select all of the ul's, we just type ul. Then, in order to select ul's that only have the class shelf, we do .shelf, make sure there's no spacing in here. If there's a space, then that means something else.

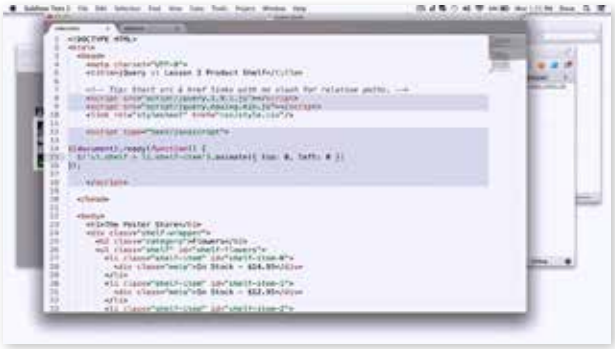
Now we don't want to select the shelf itself. We want to select all of its li children. And so, if we use CSS's really powerful hierarchy selectors, we want, in this case, to select the ul, class of shelf, and then all of its immediate descendants, which are li's of class name shelf item.

And there they are again. So that is all of the elements that we want select. Now, I'm going to go ahead and copy this guy, so I don't have to type it again, and paste it in there. So now I've selected everything.



Now what's our goal with this animation? Our goal here is to animate these items into place. Now since they are positioned relatively, place is 0 left, 0 top. Their place is kind of their origin location. So in order to do that, let's go ahead and .animate.

Make the magic call.



Now we saw this earlier. This guy takes a number of arguments. We're going to be only looking at three right now. And the first of those is the properties that we want to animate. We want the top to go to zero and the left to go to zero as well. And that's inside these curly brackets to show that it's a dictionary style, an object, a JavaScript object. You can get into way more detail on that online if you're curious. Otherwise, don't bother.

Now there's a problem with this. We're going to reload this guy, and we're going to see nothing happens. Why is that? The reason nothing happens is because these guys are already at zero, zero. So we need them to start off somewhere else. Now let's go into our style. And we're going to make our first change to our predecessor's work.

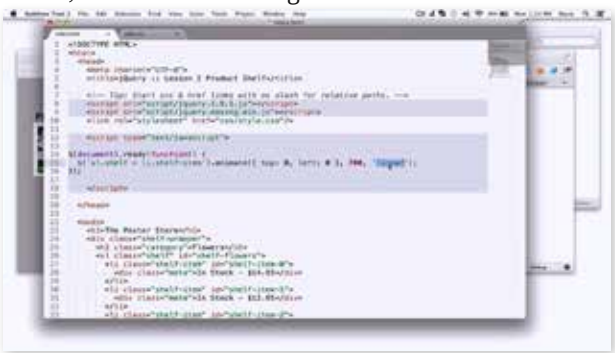
Let's find these shelf items. We've got their width, height, top, left, border, blah, blah, blah. But we don't have top. So let's say top of negative, oh, how tall is this guy, 120? Let's make this top 120px. Now minus is going to jet it up along the y axis.

So now, they're going to start out of sight, and, if we've done our job correctly, they're going to animate into place. And if you reload the page, that's exactly what you see. Now that's not awesome. It's cool that we've got something happening, but what's actually happening isn't terribly exciting.

First of all, let's say that we want this to happen a little bit slower. Now I mentioned earlier that you want fast animation when the user is paying attention to something and wants to go somewhere. You can get away with slower, flashier animations when their attention is being drawn to something, when their context is being created.

So in this case, switch back to our code, we can use that second argument to specify a much slower animation, 700. Now, this is going to look silly right now. That's OK. We're going to spice it up a little bit later on. So now we've got an animation coming in at about the speed that we want it to happen.

Now that's the second of the two. Remember, this is in milliseconds. This is 700 milliseconds, so about 7/10 of a second. For the third argument, which is the timing function (and this is where we're hiding away the most sophisticated math behind the simplest thing), now the default value (of course, there's never a point to entering the default value here, except for readability) is swing. So we do that, and that's what we get. I showed you linear before—let's try that. That looks terrible, as linear animations usually do.



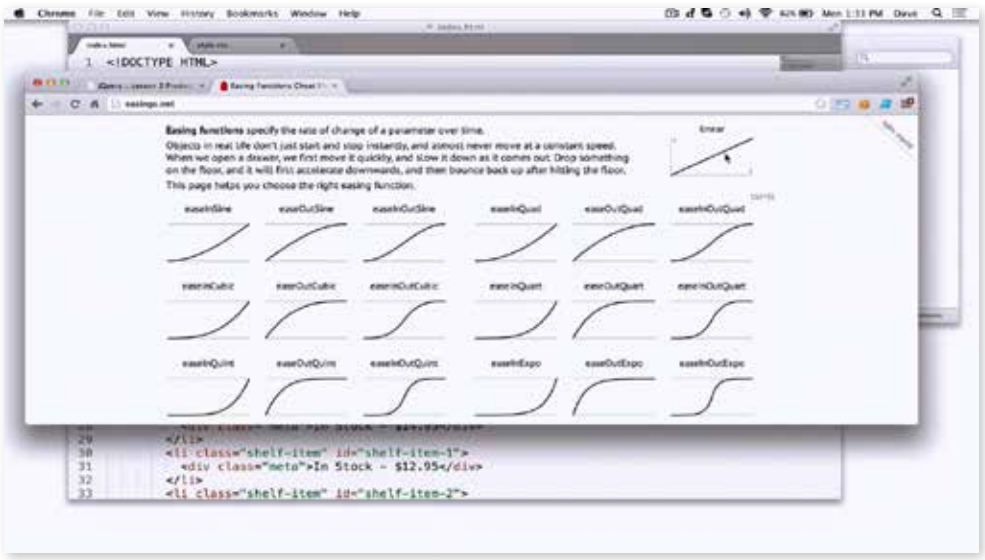
Now if you're just using jQuery, that's all you get. You get linear, and you get swing, which is this fairly decent default. Now we want a bunch of other stuff,

which is why we've included this jQuery's easing plug-in. Now, this is in your lesson materials, so you don't have to go looking for it. But if, in the future, you do want to go looking for it, just "jQuery easing plug-in" will get you right to it. Very first result, helpfully highlighted in purple because—that guy's got some problems with his blog. Let's see what's going on there—please stop hotlinking.

Yeah, fair enough. Don't use his copy on the web. Don't link directly into his copy. Instead, copy down your own, which is what we've done here. We're good citizens.

So this easing plug-in gives us a ton of other options for this timing function value. So lets go take a look at them. Now if you ever need to remember them, if you ever need to take a look at them, I highly recommended memorizing or bookmarking easings.net. It's this great, really simple resource that shows you—well, first of all, shows you the name of each one. Second of all, shows you a graph of each one, and third of all, shows you a nifty little animation for how the animation is going to ease.

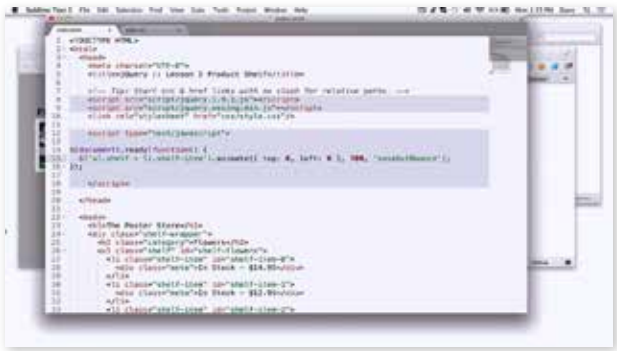
Now again, we start off with linear, which is simple and kind of not that awesome. But we've got all these other options now. We've got some pretty simple ones, basically easing in and out, starting slow, speeding up, and then ending slow. We've got these ones where it goes full speed into the beginning or into the end, depending on which the easings are on.



And then we've got some of these fun bounce ones, where it eases out and back. And that's got a really nice feel to it. That's the bounce when you get to the top or the bottom of a list on your iPhone, or your Android phone. It's that bounce that can really feel very nice.

So let's try one of those. Let's try ease out back. And I'm just going to copy it because there it is. Let's go back to our code and replace linear with ease out back, and save. Go back over here. Let's go back to our website and refresh.

Now, all right. That's not bad. Let's try speeding this up a little bit. 400, which is the default length, by the way, is 400 milliseconds. It still doesn't look quite right. I'm not a huge fan of this. I'm going to take this back to 700.



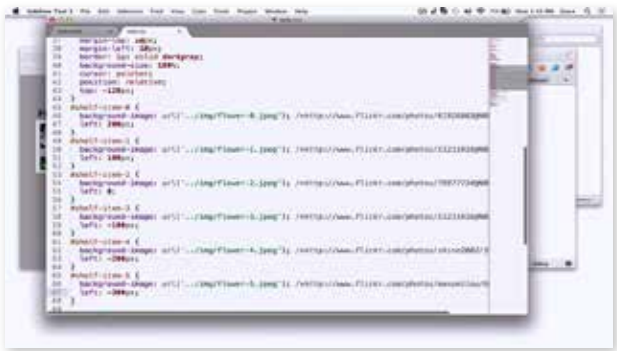
And let's see what else is available to us. We've got ease out back. All right, this elastic, maybe one these will be pretty cool? Give it a look. Drop it into our code. Save it, refresh, and that wasn't too awesome. Maybe we can slow this guy down and it'll look better. And this is fiddling, right? You're into the fiddling stage. And it wasn't bad, but it went on for a little while. And maybe that's kind of silly.

So let's go back to 700. And let's try ease out bounce. Instead of using easing elastically, where it gets to the and keeps going a little bit, it's actually going to hit a brick wall and bounce back. Now that feels right. That feels good to me. I like the timing on that. I like the bouncing effect. It's almost as though I decided on it before we started the lesson.

So that's nice. That's cool. But if we go to Apple.com/iPod, and we take a look at this guy, this is what we're going for, right? Boom. Now, they come in, it's got a little bit more elastic here. That's OK. I think we're good with where we're at.

And they fly out from the middle in this very nice, dramatic way. So let's try and get there. We're just kind of dropping in from the top. And that's fun. And that does the job of bringing your attention to it, but let's try to figure out a way—and this is a puzzle. So now we're just in a pure programming puzzle. How do we get these guys to start in a different place? Because right now they're all going to top left, 0 left. They are all returning home. They're all going to their origin. But we need to figure out a way to make them start in different places.

Now since, A, this is your first one, and, B, were kicking off from the beginning, we can cheat a little bit. There's only six of these guys. So we're not going to make it too complicated on ourselves. We're just going to cheat. We're going to tell it to start where we want it to start.



And bear in mind, the stuff that's in the CSS, that's initial. jQuery can make things happen subsequently. So let's start off, let's say, left. Now these are each 100 pixels wide. There's six of them. There's some weird math. So let's say let's start off with 200 for this guy. Let's go with 100 for this guy.

Call this guy zero—of course, you don't need to specify pixels. So let's save two bytes there. And this guy is, let's say, negative 100. This guy will be negative 200. And finally, this guy will be, of course, negative 300 pixels.

Now these are each relative to their own spot. So what are we doing here? This guy starts at zero pixels. Its origins is zero pixels. And we want him to start 200 pixels to the right. So that's going to be, over here or so. And remember, they start up a bit as well. That's going to be over here.

This next guy, we want him to start, again, on the same spot. Right about here. So we're only going to knock him over about 100 pixels. This guy, we're starting right where he is. He's good. And then this guy, we're going to have over here, which relative, to his world, is 100 pixels to the left. So that's negative 100 pixels.

We've got this all saved up. Let's give it a refresh. And there it is. We've got a nice effect. Now, I notice that we're seeing a little bit too much of that initial part of the easing. So let's maybe take the items. We're going to bump them up a little bit more. What happens if we go to 200. Let's take a look.

Maybe a little too much. Let's go with 180. And again, here we are in sit and tweak mode. I like that. That feels pretty good. So these guys are all zooming in, bouncing into place, and your boss is going to be very excited when he sees it. So everyone is super happy with how this has turned out. We've got things bouncing into place, and it's fantastic.

But let's do one more little effect before we go home. These guys are taking up a little extra space, making it just look a little bit more cluttered than it needs to. We can hide them. And then, when the user shows express interest in this particular item, by hovering, in the case, of how we're going to do it, it'll slide back into place. It's a great effect, and it makes the product really stand on its own.

So how are we going to do that? Well, we've got to go find the meta CSS. And here it is. This should be straightforward, right? That's absolute positioned, so let's just drop it 18 pixels high. There's some padding on the top. So let's call it 22 pixels from the bottom. And since it's the bottom, we have to flip the sign.

Now how does this look? Well, not great, Bob. This is off the bottom, but you can still see it. That's not what we're looking for. We need this guy's parent, which, if you check out the HTML is class shelf item, we need these guys to have no overflow. So let's say overflow hidden and refresh.

And we're back in business. That's great. We've successfully hidden them. Of course, that's only half the job. Now we want to do some more coding. We jump back into jQuery. Let's go ahead and add some comments. Whenever you're doing more than one thing in the section, you probably want to let your future self know what's going on. So this is animate in on load.

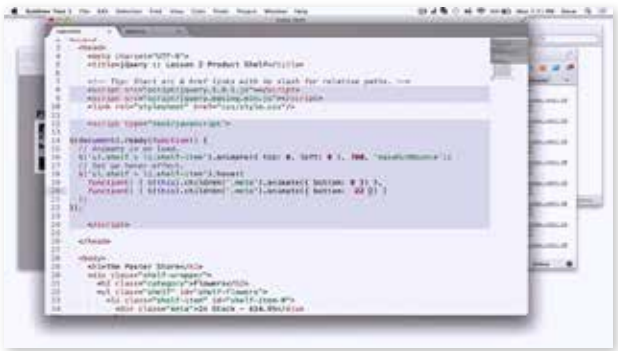
And now we're going to set up hover effects. Now, we're doing this to the same elements. So the first thing we're going to do is select them, shelf immediate descendants, li.shelf-item. And we're going to add a handler for the hover event. And again, we're going to go into more detail about events, and handling, and the complexities in a later lesson.

But for now, just follow along. Inside the over call, it takes two arguments. They're both functions. They are both things that should happen. The first one is what should happen when the element is moused over. That's the beginning of a hover. And the second one is what should happen when the element is moused out, when the mouse leaves the item. So real quick, this is what happens when it mouses over. And then the second function is what happens when it mouses back out.

So for the mouse in function, we are going to want to bring the meta element into view. And then, for the mouse out, we're going to want to leave. We're going to have it go away. So first thing you've got to know is that, in these functions, the context, the this, is the element itself. It's the element that this is happening to. So let's wrap it in an jQuery wrapper real quick. And then we want to select its children with the class meta.

So this is who it is that we're going to do stuff to. Let's just make sure that we've got this correct. So we'll wrap in a quick console.log. Let's do the same thing on mouse out via some quick copy/paste. And let's reload. Let's verify, in the console, that we are getting the things we need. And it looks like we are. Perfect.

So that's annoying. Let's go change that back. Let's get rid of the console. And once we've done that, we want to actually do something to these elements. So let's say .animate. And we're going to animate the bottom up to zero. And when we mouse back out, we're going to animate the bottom back down to negative 22.



So let's give this a look. How is it looking. Animate in, awesome. Animate out. That's great. Now, here's an interesting thing, though. If I go in and then immediately back

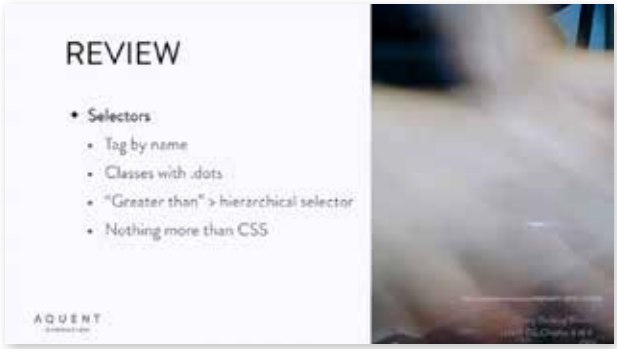
out, it does this thing where it cues it up, and it just keeps going. Now, that's not what we want. That's the default behavior in jQuery because that may be what you want in many situations. But it's not what we want here.

So let's fix this. How do we fix this? When you call .animate in jQuery, it adds it to a queue. Now instead of adding it to the end of the queue, what we want to do is we want to clear the queue, stop any ongoing animations entirely in their tracks, and start from scratch, from wherever it is. So let's call this on the jQuery object is what halts all animations. And this does what we're looking for it to do.

Now, you're going to see here, we're doing a whole bunch of chaining. We talked a little bit about chaining last time. This is a quick and easy way to just string together a series of commands. And it works great. Now let's take a look and see if this has solved our problem.

Here we go. In and out, in and out quick, in and out repeatedly. Great. Gives us a nice little DJ effect there. That is awesome. That's exactly what we were looking for. So here we've got a nice, clean poster store at the end of the day. It highlights the posters themselves. If the user wants to see more information, they can just hover over and get it. And so here we are. Day complete.

CHAPTER 6: CONCLUSION



So to review, in today's lesson we went back over selectors, selecting a tag by name, selecting classes with dots, et cetera. We also went over a new hierarchical selector, which is the greater than sign, which selects the immediate descendants of whatever you've previously selected. Bear in mind, as always, the selectors are nothing more than CSS.

Today we went over the document.ready method on jQuery, which is a great way to make sure that whatever code you're running happens after everything is ready to happen. We went over hover, which takes two arguments—mouse in and mouse out. And we went over .animate, of course the great eye light of the web.

We went over how to use stop to clear the animation queue so things don't build up weirdly. And we went over how to use children to select the children of your jQuery object.

It's possible to avoid having to use children by using a different selector. But sometimes, for example, today when we were starting with an element rather than the text selector, that's not practical.

And we went over timing functions. We went over easings.net, a great resource for being able to visualize all of the different timing functions that are available.

In conclusion, I've got some homework for you. As always, please use the pause button as you see fit as we go through it.

Assignment number one. I have a short quiz in the lesson materials.

Assignment number two. Please review jQuery. animate. And you can do that at the jQuery docs at api.jquery.com/animate. And that's a fantastic resource. There's a ton of other arguments to animate that we didn't cover today. And you can get very fancy. You can set up call backs. You can do all kinds of things. So give that a read over. At the very least, I want you to have had eyes on it. And if anything really piques your interest, then give it a go. Give it a look.

Optional assignment three. If the eye light piqued your interest, here is a very neat, very comprehensive, and a very technical blog post written by a guy who actually has the use this in his day job of lighting videos. And that's at videomaker.com.

If you have any troubles, or just have any questions, or want to hang out and chat, then we will see you in the forum. Otherwise we'll see you next session in Lesson 3, the jQuery Carousels.

This has been Lesson Two—That Certain Sparkle. I'm Dave Porter, presenting jQuery Building Blocks—Five Techniques to Cut Your Web Development Time in Half. This is an Aquent production.