6. Chaining Makes It Easier

Contents

[1. Introduction 1](#_Toc22521)

[2. Demo: Verbose Code Without Chaining 1](#_Toc15464)

[3. Demo: Basic Chaining 1](#_Toc5686)

[4. Demo: Complex Chaining 1](#_Toc30766)

[5. Demo: Single Match Chaining 2](#_Toc13836)

[6. Summary 2](#_Toc31572)

# 1. Introduction

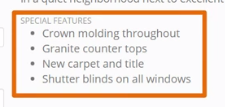
=>slides: Pg. 1

In this short module I introduce you to a technique called chaining that will make your code easier to read and more efficient

# Demo: Verbose Code Without Chaining

=>slides: Pg. 2

Now perhaps the best way to see the value of chaining is to take a look at an example that doesn't use chaining. This way you get an opportunity to see what benefit it gives you.



This example will use the special features, and I'll apply a number of different commands to the list items for the special features.



So let's begin by adding a few classes to the list items for special features. So with this statement, we'll add the highlight and bordered class to each one of the list items for the special features. I forgot an item here, so we'll go ahead and add there. So each one of the list items has both of those classes applied to the element. Now what we'll do is change the height and width,





and we'll do the width as well. So now you can see, we've changed the look of each one of those list items quite a bit through each one of the different statements that we've applied to the list items, but the problem with this is that if we take a look at each one of the statements that we've run, each time we've selected the special‑features list item and applied either the new class to it or changed the height or changed the width. Now this is really inefficient. What's happening is that each time one of these statements is evaluated, jQuery has to search through the Document Object Model or the DOM, and find each one of those items in order to apply the width, or to add the classes, or change the height. And so what you really want to do is change the syntax in order to use chaining so that the selection against the DOM is only happening one time. Let's take a look at doing that in the next

# Demo: Basic Chaining

=>slides: Pg. 3



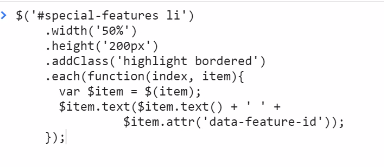
Now we left off at the last clip with code like what you see here in the console window. And so this is changing elements on the page using jQuery, but doing selection over and over again. Now what we want to do is take this code statement, and change it so that it uses chaining instead of re‑selection.



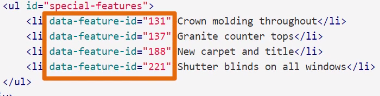
So the first thing that I'll do is remove each one of the semicolons in the middle here, and we'll just keep the last one at the end of the statement. Then what we can do is simply get rid of these selectors. And you see me lining everything up with white space after the selector, because by doing this and setting the dots all in the same place, it makes it much easier to read the statement. So now when I execute this statement, this all runs together. The selection is happening against the DOM one time, and then first it's applying the width, then it applies the hype, and then it adds each one of the classes of highlight and bordered to each one of the list items found inside the special‑features div. So visually and programmatically, we end up with the same result. It's just that the syntax that we're using is easier to read, but more importantly, it's more efficient, because we're not having to re‑select the set of items each time we're applying a command or running a function against the set. In the next clip, I'll show you something that's a little bit more involved, and you can get a better idea for how chaining makes it easier for you to read jQuery code.

# Demo: Complex Chaining

=>slides: Pg. 4

Now for this next example, I'd like to show you an example of chaining which is a little bit more complex, 

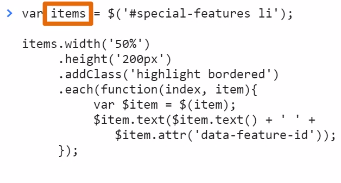
so I'll just paste in the code here and then we can talk about what's happening. So just like as before, we start off with a selector that goes to the div and gets each one of the list items. The width is set to 50%, the height to 200 pixels, and then we add the class of highlight and bordered. Now, each is a function that you can use, which works really well with chaining, because often what you want to do is create a selection and then do something for each one of the items in the resulting set. So in this case, when each executes, an anonymous function is provided, and the arguments for that function are the index of the item that's being iterated over, and the actual item itself. Now, if you look in the body of this function, what we need to do is take that item. The item that's passed in as the argument for the function is the raw DOM element. So we use jQuery. We wrap that into jQuery selector instance of the item. So then from there, we can go to $item and change the text. Now we'll set the text and update it to the current value of the text. Plus we'll go and get an attribute value out of the list items.



Now, if we take a look at the HTML for each one of the list items, you'll notice that data‑feature‑id has a value for it. So the first one's 131 and on down the line.

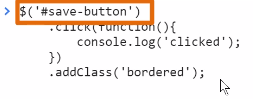


So the code over here will take that attribute out of each one of the list items and update the text with it. So this is kind of a contrived example of how you would use this, but it gives you an opportunity to see how you'd use the each function and use it with the chaining syntax in order to work with each one of the items. So once this has executed, you can see that each one of the style changes is made to the items, and then the text is updated with the original text, plus extracting out the ID from the element and updating the text with that value. So when you look at code that uses the chaining syntax or the chaining approach, it makes it much easier for you to be able to see what's going on because you can see that based off the selector, a number of different actions are being taken against that set. Now, there's one other thing that you can do in order to improve the performance of this code. So here I've brought back the original statement.



Now, instead of executing everything against the raw selector itself, what we could do is have the same code available, but notice what I've done at the top. I've had the selector run and set that aside in a variable. So now that the items variable holds the matching set, or the selected set of items from jQuery, now I can chain off of that variable and do as much as I need to. What's important here, though, is that now I can reuse items over and over again, and I don't have to reselect the page or re‑query the DOM in order to select those items. So if you're only going to work with elements once, you don't need to worry about setting up a variable, but if you need it over and over again in your page, it makes a lot of sense to create that variable. And if you're doing multiple actions against a set, you can chain each one of those commands and make it much easier for you to read and maintain in the future. Now chaining isn't just appropriate for sets. You can also use it for individual items as well, and so we'll look at this last example next.

# Demo: Single Match Chaining



Now you might think that based off the examples that I've shown you, that chaining is only appropriate for a set of items, but in fact it's not. So if we take a look at this example, you can see that what I'm selecting is the individual save‑button on the page, but then I'm chaining together the commands of providing an implementation for when the click event happens, and also adding a class to the item. So as I evaluate this code, you can see that the button is now bordered, and when I click on it, it shows up as clicked in the console. So there you have it. Chaining makes it easier.

# Summary

=>slides: Pg. 5

In this module I’ve shown you how you can use chaining…

=>slides: Pg. 6

To write clear code that reads somewhat like a sentence

=>slides: Pg. 7

And write efficient code that reduces the number of times jQuery needs to search through the DOM.

=>slides: Pg. 8

You’re doing great! With only two modules left - in the next module I introduce you to some strategies for working with user input.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*