2. Anatomy of a jQuery Command

Contents

[1. Introduction 1](#_Toc31278)

[2. Hello World 1](#_Toc26722)

[3. Thinking in Sets 2](#_Toc10289)

[4. Working with Non-matches 3](#_Toc22387)

[5. Getter and Setter Functions 4](#_Toc5071)

[6. Summary 4](#_Toc32200)

# 1. Introduction

=>slides: Pg. 1

Code can be cryptic at times, almost like we're speaking in, well, code. Well, this module is all about demystifying the jQuery syntax so that as you continue to learn more about jQuery, even beyond this introductory course, you'll have all you need to decipher unfamiliar code. So let's start by taking a look at a few of the simplest jQuery statements possible. Ah, the venerated hello world demo. As you can see, when executed in the browser, the code prints hello world in the console.

=>slides: Pg. 2-3

But there might be some things you consider strange going on here, so let's take a look at this code piece by piece.

# Hello World

=>slides: Pg. 4

So first there's this dollar sign, and you're probably wondering what the significance of that is.

=>slides: Pg. 5

And then there's this function declaration, and you might be saying to yourself, well, it seems a bit strange. It's sort of just sitting there.

=>slides: Pg. 6

Well, then there's console.log. Now this might be the only part that seems familiar to you from your previous work with JavaScript. And then we've got this curly brace and parentheses and semicolon all on their own here. Well, trust me, everything here is for a reason, so let's begin with the dollar sign.

=>slides: Pg. 8

The dollar sign in jQuery is just a shortcut variable to jQuery itself. Think of this as the front door. Everything you do in jQuery starts with the dollar sign. The jQuery library is actually declared in a variable called jQuery,

=>slides: Pg. 9

so you could write the statement like this, but having to write and read that word jQuery over and over again, and trust me, you'll be writing statements like this a lot, is just a bit cumbersome. So the shortcut of the dollar sign makes things much easier, and

=>slides: Pg. 10

it points to jQuery itself. In fact, I can even show you.

=>slides: Pg. 11

If you start to poke around in the jQuery source code, you can see how the jQuery variable is defined, and you'll see how we use those parameters of selector and context in coming demonstrations. But as you scroll down the file a little bit further,

=>slides: Pg. 12

the global dollar sign variable and the global jQuery variable is set equal to the local jQuery variable. =>slides: Pg. 13

And all that is to say is that the dollar sign variable is equal to jQuery itself. Now, let's go back to that code snippet we started with and take a look at it piece by piece.

=>slides: Pg. 14

The dollar sign, as you now know, is the shortcut for jQuery. This variable doesn't just hold a static value, but it's actually a function.

=>slides: Pg. 15

So a jQuery statement runs the function. This adds the open and closed parentheses and is terminated by a semicolon. The closed parentheses and the semicolon end up wrapping around on a few lines below. So when you see them just hanging there, they're there on purpose to complete the statements. Now, this by itself doesn't do anything, so an anonymous function is needed.

=>slides: Pg. 16

This function is called anonymous because there's no name assigned to the function. It's just there and poised to run after the jQuery statement executes.

=>slides: Pg. 17

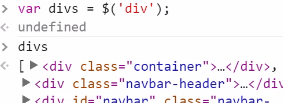
When the function is run, whatever statements that are inside the function, those run. So here the message hello world is logged to the console. While writing to the console isn't really all that interesting, what you're looking at is really one of the most famous jQuery commands, the jQuery ready function. =>slides: Pg. 18

This function is implemented on nearly every page that uses jQuery because the function runs after the jQuery script itself is loaded and the web browser's DOM, or Document Object Model, or all the elements of the page, are loaded and ready for you to manipulate and interact with via jQuery. So, like I said, this is one of the most basic commands available. But most of the time you'll be trying to select something on the page first. So now let's take a look how we can access all the divs on a page.

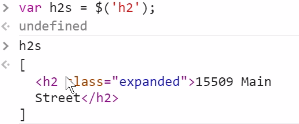
# Thinking in Sets

=>slides: Pg. 19

So this is the example page that we'll be using for many of the demos found in this course. So let's just take a look around for a quick moment, and then we can start working with jQuery. So if we go in and take a look at the HTML associated with this page, you'll notice that there's a number of different elements available. So you'll see that we have a few divs defined in the page here, and then down here, you'll notice that there's an h2 defined for the address, and that has expanded class on it, and that's the street address for our home here with encoded homes. So there's a number of different elements available, but specifically we'll be looking at the divs and the h2 for this example. So let's return now over to the visual representation of the page. And what I'd like to do is use a jQuery command to find all the divs that are on a page,



and so I'll do that by first declaring a variable, and then I'll use my jQuery command of dollar sign and then pass in the element that I'm looking for. So in this case, it's a div. And so when I run this, what it will do is return to me a list of all the divs on the page. So you'll see here, this is quite an extensive array of divs. Now, one of the first things that you'll notice by the return value is this open bracket right here, and what this tells you is that this is an array of elements. And so what it's doing when jQuery selects something on the page, it returns a set or returns an array of those items. So you'll notice I have divs here on the entire page, everything from the navigation bar of the page itself, all the way down into divs individual for the layout of the page. So even though you may be attempting to select a single item on the page, when you select something, you are going to get back a set. And so if we take a look at the selector that I used at the top here, again, you'll notice the dollar sign here. This is the gateway, or the front door, to jQuery, and then I passed in a very simple selector saying that all I want are the div elements on the page. And so the result of that, again, is this set of all the divs that are available on the page.

  
So one of the things that you can do, let's clear this for a moment, to see whether or not you have something returned from the selector that you provided, is take a look at your results. So in this case, I have the divs variable and I can inspect its length. So here you can see that there are 66 divs on the page. So that's one of the first techniques that you can use as you run a selector. If you want to find out if you've returned a result, you can take a look at the length property. Now, what if there's only one item that matches your selector? Well, that'll work just as well also, so I'll clear this once again, and now let's 

try to find the h2s. So once again, I have a result here, but I just have an array with one item because there's only one h2 defined on the page. And, of course, if I take a look at the length, I'll get back 1 for that.

  
But if I run another jQuery function against my wrapped set, and that's what this is called is a wrapped set of jQuery objects, against either h2s or my divs variable, they'll behave in a similar way. So since I'm only returning one item in the h2s, if I decide to change some of the CSS properties or change some of the text in that item, and I'll show you how to do that in coming demos, since I only have one item in the wrapped set, it'll only execute against that one item.

  
But if I want to do it against all 66 of the items in my divs variable, it'll execute against all of those items. So the takeaway for you here is that most often with just about any jQuery statement that you build, the first part of it will be the selector, and I'll show you a number of different types of selectors. In this case, all I'm doing is selecting defined HTML elements, either div or h2, and there's more complex ways that you can do your selectors. But the first thing that you'll want to do is tell jQuery what it is you're looking for on the page, and you do that with a selector. So that works great. But what happens if you try to select something that doesn't exist? Well, we'll take a look at that next.

# Working with Non-matches

=>slides: Pg. 20



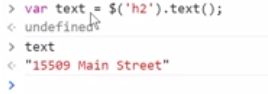
Now jQuery is also very forgiving. If you try to pass something into a selector that doesn't match or doesn't even exist, you might think it would throw an error, but, in fact, it does not. So let's try this for a moment. Let's try to put in a selector for an element that does not exist anywhere in HTML. So here we can take a look at the result of that selection, and that's simply an empty array.



So, again, we could take a look at nomatch.length, and that will return 0. So as you're using the results of a selection, you can take a look at the length property to find out if you've found what you're looking for as you're interrogating the page. And even if you don't find something, or even if you pass in something that doesn't exist within the page, instead of returning an error, what you get is an empty array. So that makes it very easy for you to write code that doesn't throw errors when you don't want it to, by making sure that you're looking for things like length against your selections. Okay, so now that we've figured out how to select something on the page, let's take a look at what we do once we have something selected.

# Getter and Setter Functions

=>slides: Pg. 21



Now once you have a set, a selected set of elements on the page, there are a number of different functions that you can use in order to interact with the elements that have been selected. So let's turn our attention again to the h2 on the page, but this time what I'd like to be able to do is get the text out of that element. So this is the selection that you saw before, and so now what I'll do is run the text function against that selection. And so now when I take a look at text, you can see that's the value that's inside that h2. If we take a look at the markup and scroll down here, that's the text inside the h2 itself. Now, we can use that function in a little bit of a different way. Let's say I want to change the text that shows up on the page here. I can use the very same function, but instead of simply calling the function, I can pass in a value as the first argument to that function, and that will change the text within that item. So here, let's take a look at doing that a little bit differently.



So let's select the h2 once again and call text, but this time I'll change the street address. So notice over on the page, 1234 Main Street is now what shows up on the page, and I did that by calling text and passing in the value that I want. So many functions that are available off of a selected set of a jQuery selection are both getter and setter functions. So when I called just text up here, that was a getter function. And so by calling text, it returned me the text of the element. But then I could set the value of that element by calling text and passing in a value. And again, we're working in sets, so I could just as easily work with a different element and set all of the text within that same element at the same time. In fact, let's try that with an unordered list.



So here I can select all the list items on the page, and this will be more than just elements within my example, but then I can go and set the text to list item. And so there's a number of different items that are available on the page, and so you see down here as well as up at the top and even the tabs, all of the list items have now been updated to the text of list item because it's working against that entire set that matches the selector.

# Summary

=>slides: Pg. 22

So, to wrap up this module, you've learned about the significance of the dollar sign variable in jQuery. You also learned a little bit about anonymous functions and how they can be used with jQuery and how to think in sets. JQuery inherently works with sets as you select parts of the page and work with them as elements in the DOM. So in the next module, we'll be talking about finding parts of a page, and I'll show you a number of different techniques for you to be able to find different elements within a web page. I'll see you there.

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