3. Finding Parts of a Page.

Contents

[1. Introduction 1](#_Toc30777)

[2. Introduction to Selectors 1](#_Toc21253)

[3. Selectors in a Web Page 1](#_Toc4234)

[4. CSS3 Selector Review 2](#_Toc25072)

[5. Demo: CSS3 Selectors 2](#_Toc13519)

[6. Demo: HTML Literals 2](#_Toc30764)

[7. Demo: DOM Elements 2](#_Toc1692)

[8. Demo: Find 2](#_Toc27749)

[9. Demo: Parents 2](#_Toc30632)

[10. Demo: Children 3](#_Toc4149)

[11. Summary 3](#_Toc1793)

# 1. Introduction

=>slides: Pg. 1

Presenter

2015-05-14 09:34:39

--------------------------------------------

One of jQuery’s greatest strengths is it’s ability to find parts of the page in an easy and reliable way. Modern web pages are often complex with hundreds of elements on a single page, so being able to easily find just what you are looking for is crucial – and jQuery makes this a snap.

In this module, we'll discuss many different ways to find parts of a page. All right, let's go ahead and get started.

# Introduction to Selectors

=>slides: Pg. 2

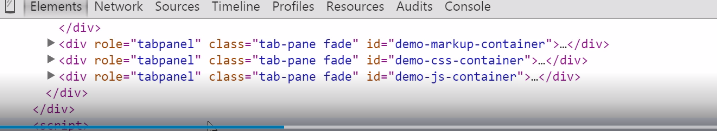
Coming back to our example, the Coded Homes page has the logo in the upper left of the page [click], while the logo is one of the first things you see on the page, it appears there as a result of a bit of a deep hierarchy on the page. If we just look at the skeleton of this page for a moment.

=>slides: Pg. 3

you'll see how the page begins with the root html element and then proceeds down through the body element, inside a number of containing divs, and some are named, others are not, through the anchor element that makes the image a link, and then, finally, to the image itself. HTML is, by nature, hierarchical. So how do we find what we're looking for in light of the sometimes complex structure of a web page? Well, you use the right selector. So what is a selector?

# Selectors in a Web Page

=>slides: Pg. 4

If we take a look at the running version of our example page, we go over to the image. I'll right‑click and then say Inspect Element. Now let's rearrange things here for just a moment so that we can take a look at the selector that is built up in order to find this element on the page. So let's take a look down here at the bottom of the screen. So you'll notice first that it's looking at HTML. Inside of that is the body element and then a div element. And then here we have another div element, but it has the id of demo‑content‑main‑container. Now this is the same hierarchy that you saw on the slide a moment ago, but here you can see how it ties together, specifically within the web page. So then we have a few other divs and then another div that has an id to it, so demo‑container. And then it also has a class applied to it, which is .tab‑pane and then .active. So those two classes are applied to this one div. And then inside that, we have a div of house‑detail. And then finally, we get down to the logo. Inside that div is an 

anchor tag, and that anchor tab wraps the image tag itself so that it can make a link for the image. So this entire string, this entire hierarchy that you see down here at the bottom of the screen, this makes up basically the full selector that you could use in order to find this one image element on the page. Now obviously, if you were trying to narrow down to just that image, you wouldn't need to use the entire selector that you see here. But what you can see is how different elements can be identified in different ways. For instance, here we have a div, which we could find if we were looking for all the divs on the page. But we could also specifically look for the id of demo‑content‑container or look for elements that have the tab‑pane or the active classes all applied to them. So the selector gives us a number of different ways in order to find elements on the page.

# CSS3 Selector Review

=>slides: Pg. 5

jQuery is largely based on CSS3 selectors. As you just saw – the selectors are strings that identify elements on the page. Now, an intimate knowledge of CSS3 is NOT a requirement for this course, but just to refresh your memory let’s review a few principles of CSS selectors.

=>slides: Pg. 6

One of the most commonly used attributes of HTML elements the ID [click]. IDs are special because the values are meant to be unique across a page. This is helpful when working with selectors because you can easily zero in on an element on the page by using the ID selector. [click] here you can see how the pound sign is being used to tell jQuery that the ‘content-container’ is an ID value.

=>slides: Pg. 7

In contrast to IDs classes are meant to be applied to one or many elements on the page. [click] This makes it easy to locate collections (or sets) of elements that either require styling (in the context of CSS) or manipulation (in the context of jQuery). [click] When you are targeting a class, the selector is prefixed with a dot – as shown here.

=>slides: Pg. 8

The last item I’d like to mention is the concept of a pseudo class. Pseudo classes reflect a specific state of an element. [click] In the code snippet the pseudo class is being used to apply style to the hover state of a link. [click] And in jQuery the pseudo class is used to find the first element inside a DIV. Pseudo classes are easy to spot because they make use of the colon after the element name. Now this is just a \*small\* sampling of the type of selectors you can create…

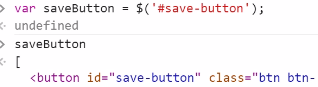
# Demo: CSS3 Selectors

=>slides: Pg. 9

So let's start by taking a look at how jQuery uses CSS3 selectors, and we'll start off with some real basics here.



First, let's look at finding all the buttons on the page. So by executing this statement and passing in the selector of just button into the jQuery function, it finds all of the buttons that are found on this page and sets them equal to an array here called buttons. So you'll notice that the first one here is a collapse button, which you can't see right now because this is the Bootstrap button that collapses the navigation when it's working within a smaller view port. The next two buttons you can see on the page. So you'll notice that down here the Cancel button shows up and the Save button for working within the form for the Coded Homes page. And the last button you can't see. It shows up on the JavaScript tab, and that's just a little alert message telling you to open up the console window if you want to work with the code. But the value here for you is to say that all I had to do was pass in a known HTML element. So this is button. I passed in the name of the HTML tag of button, and that found all of the elements within the page and placed it into an array for me. So that will work with any HTML element that you have on the page. Now let's look at using a selector that's an id selector. So now instead of looking for all buttons, I'll look specifically for the Save button.



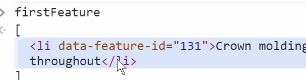
And so when I execute that statement, all that's returned to me now is the Save button. And remember, this follows the principle found in CSS, and that is to say that any element with an id is expected to be unique on the page. So by going specifically for save‑button on the id of this element, I pretty much know that I'll get back just that one item. So ids can be very, very useful. Now if I want to look for more than one item at a time, of course, I can use the HTML element. But if I want to look for something more specific, I can look at a class. So let's look at what that would look like next. Now if you remember, a class is found in a selector by prefixing what you're looking for with a dot. So let's take a look at just finding buttons that have the btn class applied to them.



So here I'll do class btn looking for the class of btn on the elements. I'm going to execute this statement. Now I can take a look at what's returned. And you see all of the different items that have the btn class applied to them. And you'll notice because of the way Bootstrap works, you can apply the btn class to anchor elements as well. So these are just regular links that are being styled as a button. So here I find everything that has the btn class applied to it. Now those are some of the most basic selectors. Let's try something that's a little more robust, something that we might expect to find within a CSS3 selector. What I'd like to do with this next one is combine an id with a pseudo class. So let's find the first feature. So the item that I'm looking for here is trying to find this list item here that's the first features found in the special features.



So to do that, we'll create a variable. And then first, we'll look at the id of the containing div that has those list items in it. Now I'll add a space, and I'll say that what I'm looking for is that container div and then something inside of it. So that space then will say now I'm looking for a list item, but what I'd also like to do is find the first‑child.



And so when I execute this statement, when I look at firstFeature, that returns to me that list item here of crown molding throughout. So as you can see, you can get very specific with the selectors that you use that you pass into jQuery in order to find elements on the page very quickly. The last one that I'd like to show you is how to find links that are only external links, so not relative links to the page. And to do this, we'll use a CSS3 selector that uses the startsWith operator.

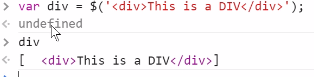


So let's start off by creating a variable. And here I'll look for any anchors on the page that have an attribute of href that start with the value of http. And when I execute this statement and we take a look at what it returns, you'll notice that each one of the anchors that are returned are only items that are linking to something outside the page. So in other words, there are no relative links returned in the result set. So here's an anchor with the href of http://codedhomes.com or api.jquery.com on and on. So the CSS3 selector of looking for the anchor tag with the href value equal to http, but by using this caret, I'm saying that it starts with whatever string I'm passing it into. So the more you learn about CSS3 selectors, the more effective you'll be working with jQuery in order to find elements on the page. So next, let's take a look at what it's like to work with HTML literals with jQuery.

# Demo: HTML Literals

=>slides: Pg. 10

Now from time to time, you may want to create HTML literals and select those with jQuery in order to create new elements on the page. From there, you can change the way they look or behave, but first you need to know how to create them using a literal selector.



So let's create a div, a brand new div, just in memory. So here my selector, instead of being a CSS3 selector, will actually just be a string of HTML. And so now as I execute that and take a look at the result, you'll see that what's returned here is a element that's been created in memory based off the string that I passed into jQuery that created that div in the result set. So now once I have this here, I can change the way it looks. I can change the way it behaves. I can add it into the page, and there's a number of different ways of doing that. But what I want you to understand right now is just that by passing in a fragment of HTML, you can create new elements in memory using jQuery. Now you can also do the same thing with arrays. So if I want to create more than one element, I can do that as well.



So here's what I'm doing is I'm passing in an array to the jQuery function, and then I can add in multiple literals. So here I'll add in a span and then add in another. So let's take a look at the result there. And as you can guess, now what I have is a selected set of new elements that are created in memory, one of the span one and the other one of span two. So by using HTML literals, you can create new elements in memory and then work with them on your page, all using jQuery. Now there may be times when you're working with raw DOM elements, and what you can do is you can pass those into jQuery as well in order to get a wrap set. So let's look at doing that next.

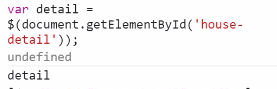
# Demo: DOM Elements

=>slides: Pg. 11

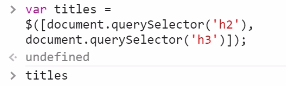
Now sometimes when you're working with a page, you'll have an opportunity to have access to raw DOM elements. So, for instance, if we want to find the div on the page for house detail, we might do something like this.



So we'll look for the element by its id and then pass in the id value of house‑detail. And so this returns the actual DOM element itself, the raw DOM element of house‑detail. So what we can do is you can take that same DOM element and pass it into jQuery and then work with a wrap set. Now the value for this will become more apparent as we continue on in the course because once you have elements selected with jQuery, there's a lot that you can do with it. So let's take a look at how we might work with a raw DOM element directly in jQuery. So let's do the same thing that we've done in the past.



Let's create a variable. And then we'll pass in a selector to that DOM element directly into the jQuery function. Now you'll notice here I'm not putting a pound sign in front of the id because that's something specific to jQuery because that's a CSS3 selector. Here, document.getElementById simply takes the id value. So as that's executed and I take a look at the value for detail, you'll notice that it's wrapped in brackets as an array. And this is an indication that this is a wrap set from a jQuery selector. And just as we did with HTML literals by passing in an array in, as well as individual items, you can do the same thing with raw DOM elements.



So let's take a look at trying to find the titles on the page. You'll notice I do my open and close brackets and open and close parentheses all at the same time. That way, I make sure they don't leave anything off at the end. Now document.querySelector is the native browser implementation for doing selection inside the browser. So what this says is it will take in a query and it will look for any element that matches the h2 selector. So we'll use that to find h2s and h3s. So I can pass all of these into jQuery. And then when we take a look at titles, you can see that it finds the h2 and the h3 all on the page. So working with DOM elements works just as well when you're using jQuery, as well as using selectors. Now once you've found a specific element, you may want to find something that it's in relation to. So we'll look at using the find function in the next clip.

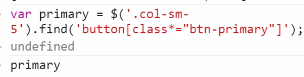
# Demo: Find

=>slides: Pg. 12

Now sometimes you'll locate an element on the page, but what you want to do is find something within the hierarchy of that element, so something that's hierarchically underneath the element that you found. So one of the things that you can do is use the find function. So in this case, what we'd like to do is find the primary buttons on a page, but only do that within a specific div element. Now there's a lot of different ways that you can do this, but I'm using this example in order to show you how to use the find function.



So the first thing that we'll do is take a look at finding just the column on the left‑hand side of the page. So we'll create a variable for the primary button. And the first selector that we'll use here is for the class of col‑sm‑5, and that's a Bootstrap class that says it's a column that has 5 units of width to it. The details of that aren't important. You can watch the Bootstrap course if you'd like to find out more about that. But for right now, we're just looking at a specific class on the page. The interesting part is to say that once we found that item, and you can see that it's selected here. So this is the column or the division that we found on the page, and we want to find that button. It's the Save button that we're ultimately looking for.



And so what we'll do is add to our item here and say that now that we found that, we want to run the find function. And here I can pass in another selector. Now I'll make this a very specific selector. And of course, you have the power and capability of CSS3 selectors available to you. So it doesn't necessarily have to be this complex, but I'll show you what's possible here. So we'll say that we're looking for a button. Now there's two buttons within that column there, so we want to find something specific. So we want to find a button that has a value within the class attribute. That's anything that equals btn‑primary. And so now, instead of looking with the selector, trying to find everything on the page, what it will do first is select the left‑hand column first, and then as it runs the find function, it will only look for items specifically within the context of what was initially found within the first selector. So as we look through the left‑hand column, it will only look for items within that. So let's execute this and then take a look at the results. And so you can see, it found that element on the page by narrowing down its scope to just what's in that left‑hand column. Now another really useful capability is to be able to find items in relation to elements that you've found. So let's look at trying to find parents and children of an element.

# Demo: Parents

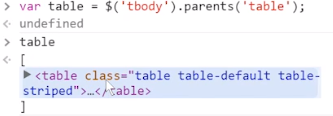
=>slides: Pg. 13

Now the next function I'd like to show you how to use is the parents function. So let's take a look at the HTML for a moment.

  
What we're doing is we're going to look at selecting the tbody of this table.



So if you need to find elements in relation to something that you've selected on the page using jQuery, you can do that using parents, but that gives you the entire hierarchy chain. But what if you just want to find the immediate parent or a parent of a specific element type? Well, let's take a look at doing that.



So here, what we're going to do is we'll select the tbody, but we just want to find the parent table itself. So in this case, we're starting off with the exact same selectors we had before, and we'll call parents once again, but in this case, we'll say that we're looking for parents that have the element of table in them. And in this case, there's only one match. And so you can see here table maps to the actual table that is found as the parent element for tbody. So, depending on how you need to find the parents of the elements that you've selected, the parents function can be very powerful in finding those items within context. All right, now let's look at working with children.

# Demo: Children

=>slides: Pg. 14

Now, just as we worked with the parents for the tbody, we can also do that with children.



So let's go back to our friend tbody, and let's get the children and use the very same selector. So we're just looking for the tbody and asking for the children of this item. And when we inspect the results, you'll notice that it returns each one of the rows that's a child to that tbody. And that's the full element graph of that item. So as I expand this, you can see that it has the header and then the data item within that row. So if you need to find children of an item, you can just call the children function after you've run a selection.

# Summary

=>slides: Pg. 15

In this module you’ve learned how to find parts of a page using jQuery….

=>slides: Pg. 16

You’ve seen how to use CSS3 selectors with jQuery

=>slides: Pg. 17

You’ve also learned how to find elements by relation by locating parents and children from an element.

=>slides: Pg. 17

In the next module you learn to start changing how elements work by working with style sheets and classes.

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