4. Adding Style and Effects with a Little Class

Contents

[1. Course Overview 1](#_Toc18888)

### 1. Introduction

=>slides: Pg. 1

Often one of the most common actions developers want to do after elements are selected is to change how elements look. This module demonstrates how to manipulate the styles applied to selected elements and how to do it all via jQuery.

=>slides: Pg. 2

CSS, or Cascading Style Sheet are the way that web pages get a look and feel applied to them. By way of a short review I’d like to quickly go over some of the basic rules of Cascading Style Sheets just to make sure you have the appropriate context for the coming modules. So let’s take a look at how styles behave on a web page.

=>slides: Pg. 3

So first, let's look down at the bottom of the screen here, and you can see the style sheet that's applied to this page. So the body rules there just give the font a little nice look to it, and then I've added some padding to the page.

=>slides: Pg. 4

What we're more interested in here is the values of the font size. So for the h1, the font size is set at 55 pixels, and that's a pretty large font size. And then the title class. When the title class is applied to the element, that takes the font size down to 35 pixels. But if you notice up in the right‑hand side, I'm cascading instyle!, that's not really quite as big as 55 or 35 pixels. The reason for that is because there's an inline style being applied to this element, setting the font size at 25 pixels. So even though the rule for the h1 of 55 pixels is being applied to this h1 element, that's ultimately being overridden by the style, the inline style, setting the font size at 25 pixels. So let's see this in action.

=>slides: Pg. 5

If I were to come in here and remove the style, the inline style, from this element, you'll notice now that the font got much larger, and this now is rendering at 35 pixels for the font size. And that's because the title class is still being applied to the h1. So if I were to remove the title class, now you can see that it's being rendered at 55 pixels. So this demonstrates the cascading nature of style sheets. The first styles that are applied to the page are the ones that are linked to from a linked style sheet like you see here in the link element at the top of the page. Then the next precedence is any styles that are applied to an element itself. So down here you can see the h1, so then the font size where the h1 is applied. But that's overridden by the rule given for the class of title, and ultimately, if I put these back in, you can see that the inline style, the one that's the most specific, is the inline style setting the font size at 25 pixels, is ultimately what takes the highest precedence. So when we talk about Cascading Style Sheets, we're talking about the nature of cascading rules that go from general to specific. So now that you've had a chance to get a quick primer on how CSS works in the browser, let's turn our attention to how you can use jQuery to change the style of selected elements

=>slides: Pg. 6

# Demo: attr

The first example that I'd like to show you is how to use a function of jQuery that gives you the chance to manipulate just about any attribute of an element, and so that's the attr function.



So we'll start off by using the same selector that we've been using in the past, and that's by looking for the h2 on the page. Now with this selected, we can use attr and then tell it what attribute of the element we want to affect. So in this case, it's the style attribute. And as the second argument, all I need to do is pass in a style sheet rule. So here, let's change the background color of this element. So here you can see that the background color is now changed. So this is the brute force way of changing an element's style. So you can go directly against the style rule and change it how you want, but there are easier ways to interact with an element's styles. So let's take a look at using the CSS method itself in the next clip.

# Demo: css

=>slides: Pg. 7

Now the css function cleans up your code quite a bit in working with an element's style sheets. So previously, we had a selector that looked like this, and that works, but there's a way to clean this up quite a bit. Instead of using the attribute function, here we'll use css, and then style becomes unnecessary at this point. So now all we need to do is work with individual style properties. Now if you recall, getter and setter functions allow you to get values out, as well as pass values in.

  
So in this case, what we can do is, we can interrogate the style sheet and find out what the current value of the background color is, so we'll set this equal to a variable. So when we take a look at the result of bg, you can see that the current background value is set as white, or rgba(0, 0, 0, 0) So if we want to change that value, we can use a similar type of syntax. So here we'll use the h2 selector and then, instead of passing in a string for background‑color, we'll pass in a hash that has the style sheet rules that we want applied to the element. Now, since this is a JavaScript object, normally background color in CSS rules is the word background‑color, like you see at the top here. But what we're passing into the css function is a JavaScript object, so backgroundColor here will be camel cased, and then we can pass in a value for it. So this sets the background color of the element to the value that I passed in from calling the css function. And the camel casing works with the getter as well.

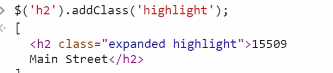


So if I were to ask for the backgroundColor of the element by executing this statement, you can see that I get the color code for the orange that's shaded for that element. So the css function is quite flexible in allowing you to set individual style properties for each one of the elements that you have selected, and it allows you to do it in a very clean way. Now you can enjoy even more flexibility in working with elements on the page by using classes rather than having to use specific style sheet rules. So next, let's look at adding and removing class from an element.

# Demo: addClass and removeClass

=>slides: Pg. 8

Now working with classes is often a big advantage to you as you're dealing with styling elements on your web page, because with classes, you can group together different styles at one time, rather than working with individual style rules.



Now, you may remember a demo like this in the introduction to the course. We're going to show you this with a little bit of a twist on it here now. So to add the highlight class to the h2 element, we'll use the addClass function. So this, of course, adds the highlight class to the element, and you can see the result here on the screen.



And of course, we can remove that class just as easily by calling removeClass. And you can see from the result that comes back into the console window here that there's still another class applied to this h2 element of expanded. So by calling removeClass we're only removing the one class that we've passed into the function. But there's other ways that you can use this function as well. Rather than passing in a string, you can pass to addClass or removeClass, a function which allows you to decide which class you're manipulating. So if you have some logic that you need to run before you decide which class you'll add or remove, you can do that quite easily.



This is what that would look like. So here I'm just returning a string, but as you can imagine, you may have some small bits of logic that you want to run to decide whether or not you pass bordered back or some other class name. So when we run this, it adds the bordered class to the h2 element. And of course,



we can do the same thing with removeClass that we did with addClass by allowing it to run this function and to return bordered for the class that needs to be removed. So there is a lot of power available to you just in these few functions that allow you to change individual style sheet rules, as well as add and remove classes to elements as you're working with your pages. In the next clip, I'll show you how to use some effects that will allow you to take control of whether or not elements show up or are hidden on the page.

# Demo: show, hide, and toggle

=>slides: Pg. 9

Quite often, one of the first things that you want to do as you're working on the UI of your page is to show or hide different elements based off of certain logic that happens in the background. So by using addClass, removeClass, you can achieve those types of interactions, but it gets even easier when you use some of the specialized functions that are available within jQuery. So first, let's take a look at how easy it is to hide an element on the page. So let's hide our address here.



You simply call the selector and then use the hide function. So watch carefully as a street address disappears from the page. Now, of course, making it reappear is just as easy.



By using the same selector, we can call the show function, and now it appears back on the page. Now you can imagine there's probably certain areas of logic where you may want to hide or show an element, depending on certain settings that are happening within your logic. So in order to accommodate that, in order to make it easy, there's a function that you can use called toggle, and toggle does just that.



if it's shown, it will hide it, so right now it's appearing on the page, so it hides it. If I call it again, it'll show it back up on the page. And in fact, if I explicitly say that I want to hide this item, it knows that it's hidden so that if I were to call toggle again, it'll make it appear back on the page. So by using show, hide, and toggle in the appropriate places, it makes it very easy for you to make certain elements of your page appear and disappear. And just like you have the option when using add or removeClass, you can also pass a function into toggle as well. And it's actually the same for show, hide, and toggle, but I'll show you it with toggle.

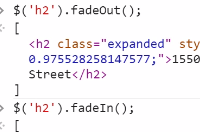


So here, by calling toggle and then passing in a function, I can just write out to the console that this element has been toggled in its visibility. So you see now it disappeared, and then in the console, I got the text written out based off of the command that I gave it in the anonymous function. So this is really powerful. It really allows you to take control of your UI. But there's another command that I'd like to show you next, which gives it a little bit more of a refined look.

# Demo: fade

=>slides: Pg. 10

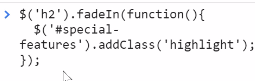
Now showing and hiding an element get the job done, but by using fading effects, you can make it look even more refined.



So watch the address here as I tell it to fade out. That was a nice little transition in order to fade it out on the page. And of course, we can do the very same thing with fadeIn. Now you do have control over the speed in which this effect is applied. So here I can say that I want to fadeOut, but I wanted to do it quickly, so I tell it to execute that effect fast.



So that was a lot faster than the time before. And, of course, in the very same way you also have control over being able to execute a function after the effect is done executing. So here let's fade in and then run some code once it's done fading in.



Just to mix things up a little bit, let's add a class to another element on the page once this one fades in. So here you can see that after that address fades in, then we'll add the highlight class to special features. And if you were watching quickly, you saw that the class was applied after the element was done fading in. So by having just these few tools available to you gives you enough power to make a truly interactive UI for each one of your applications, all using jQuery. So now the next thing we want to turn our attention to is working with some of the events on the page so that we can apply these interactions at just the right time.

# Summary

=>slides: Pg. 11

In this module you’ve learned about how you can use jQuery to manipulate style and add effects to selected elements.

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