**Your first jQuery-enabled page**

- In this chapter, I'm going to give you a quick introduction to the main features of jQuery, which we'll then cover in more detail throughout the rest of the course. By the time we reach the end of this chapter, you will have seen how jQuery manipulates page content, builds animations, handles events, and even uses AJAX. Now that we've set up our development environment, and learned a little bit about jQuery at a high level, it's time to build our first jQuery-enabled webpage. For our first example, we're going to keep things pretty simple. Here is what our first jQuery page will do.

We'll include a link to the jQuery library that we downloaded earlier.

We'll set up an event listener to trigger when the DOM of the page is loaded.

And inside the event handler, we'll use jQuery to insert some content into the webpage,

indicating that jQuery has properly loaded and is working correctly.

Let's fire up our code editor and see how all this is done. All right, here I am in Adobe Brackets. And what you'll want to do is click this little menu up here and choose Open Folder, and then choose the ExerciseFiles folder, which will open the folder here in Brackets. Now again, this is if you're using Brackets. If you're using a different code editor, you can open the code however you see fit. So, once we have the folder open, I'm going to open the FirstjQueryPage\_start.

Now, you can see there are two HTML comments here that indicate where we need to place certain things.

So first, I'm going to include the script reference to jQuery. So, I'll just add that, script source, and I'll just point that at jQuery 3.0.

<!DOCTYPE html>

<html>

<head>

    <title>First jQuery-Enabled Page</title>

    <link rel="stylesheet" href="../style.css" />

    <!-- insert the script to reference jQuery here -->

    <script type="text/javascript" src="../jquery-3.0.0.js"></script>

    <!-- insert the script to display the Hello World message here -->

    <script type="text/javascript">

Now, I want to point out here, I'm loading jQuery 3.0.0 because that's what I downloaded. If when you download jQuery, they've bumped the version number to 0.1, or 0.2, whatever it is, make sure that you are editing the version number here, so that you're loading the proper version. The examples that I've provided all use jQuery 3.0. So, if you download a different version number, make sure you take care of that. All right, so I've included the script tag, and this script tag references jQuery in our webpage. Now, in this case, I'm using the uncompressed development version of jQuery. And again, in your production webpages, you would usually replace the development version with the compressed production version in order to save space and improve download time.

The next step is to write the actual code that will listen for the page load event, and then insert some content into the page. So, let's do that.

And I'm gonna write $ document .ready, and then I'll put a function in here. And my function is going to use the jQuery function again. I'll say #content .append, and actually, have to put a script tag around all of this as well.

        // Inserting content using jQuery

        $("document").ready(function() {

            $("#content").append("<p>The page just loaded.</p>");

        });

So, let's just do that. Here we go. And I'm going append a string of text. All right, let me explain what's going on.

So, in this case, I'm calling the jQuery library,

which is referenced by this dollar sign right here.

And I'm using the "**document**" keyword, **which indicates that we're about to perform an operation on the page itself.**

The **ready** function sets up an event listener for when the DOM structure of the page is fully parsed by the browser, and is ready to be operated on.

Now, this is different from the onload event, which you might be used to using, because onload fires only after all of the page content, including images, have been loaded.

Now obviously, depending on how many images you have, and how big they are, that can take a while. The **ready event allows us to operate on the page much earlier than the load event does**. So, I pass a **callback function** to the ready function, and this callback function is the **event handler** that will be fired when the ready event is triggered. So, the code once again calls the jQuery function, which is this dollar sign function, right here. But this time, it passes in the keyword "content" as the argument,

***indicating to jQuery that we want to perform an operation on the element in the page that has the id of "content", which if you look down into the page structure, is this div, right here. And this is an example of jQuery using CSS style syntax to operate on page elements. In this case, the code uses the append function. The append function appends a piece of content to the tag that we passed to the jQuery function, again, in this case, that div tag. The content being appended is this paragraph tag, which just simply says, "The page just loaded."***

// Inserting content using jQuery

        $("document").ready(function() {

            $("#content").append("<p>The page just loaded.</p>");

        });

All right, so let's go ahead and run this in the browser. So, I'm going to save this. And here in Brackets, I'm gonna click on this little lightning bolt icon, which Live Preview, which will bring it up my browser. Now, if you're not using Brackets, you can just double-click this file in your operating system, and that should just launch the browser. And you can see that when the page comes up in the browser, the text is inserted into the webpage. Now, all this might not seem like such a big deal. But for comparison, let's take a look at the code I would have to write using the browser DOM if I didn't have jQuery. So, in Brackets, open up First QueryPage\_finished.

And what I'm going to do is take a look at this code that I have down here, below the jQuery part, just for comparison.

  // Inserting content the old way using the DOM

        window.addEventListener("DOMContentLoaded", function(evt) {

            var elem = document.getElementById("content");

            var para = document.createElement("p");

            var text = document.createTextNode("The page just loaded.");

            para.appendChild(text);

            elem.appendChild(para);

        });

    </script>

</head>

<body>

    <h1>First jQuery Page</h1>

    <div id="content">

        <!-- we start with an empty document -->

    </div>

</body>

</html>

So, in this case, I'm using the window objects addEventListener method to listen for an event called DOMContentLoaded. And that's the corresponding standard DOM event for when the DOM structure of the page has been loaded. And once again, I pass a callback function. But in this case, I have to have three separate lines of code just to get the body tag and then create the new paragraph, and I have to create a new TextNode for the text string, using standard DOM functions for each. So, that's three lines of code just to create the new piece of content. And now, I have two lines of code to insert the content into the document. So, I have to append the TextNode inside the paragraph, and then I have to append the paragraph inside the body element. So, in total, that's five lines of code to accomplish what jQuery does with, essentially, just one line of code. So, just to make sure this runs, let's open this in the browser. And you can see that in both cases, the content has been inserted. So, even though this is a basic example, it should give you some idea of the power of jQuery. But that's just the beginning, and we'll see a lot more of this power as we move through the course.