jQuery JavaScript Library

by Sophia

WHAT'S COVERED

In this lesson, you will learn about the JavaScript framework called jQuery and how it can simplify JavaScript code. You will learn some of the basics of jQuery syntax. Finally, you will learn about the options object, which can be used to customize how some of the jQuery features operate and behave.

Specifically, this lesson will cover the following:

BEFORE YOU START

You may notice the version number in the bolded URL below and that it comes from ajax.googleapis.com. This may be different from the version you see because of the different versions and CDN providers. Some developers use the googleapi.com CDN, while others go right to the source and use code.jquery.com instead. While the CDN provider does not make much of a difference, the version number could. You want to make sure that the version used matches the version that was used when the code was developed.

1. Introduction to JQuery

JQuery is a JavaScript library designed to simplify JavaScript syntax and provide convenient shortcuts to commonly used commands. Since jQuery runs right on top of JavaScript, there is no special software needed on the client's system or on the server. The only requirements are an understanding of JavaScript, CSS selectors, and the jQuery syntax. As for implementing the jQuery framework, it is as simple as using a <script> tag to link to the jquery-3.7.1.js file. As for where the file is located, there are two options:

1. You can download the JavaScript file, place it in your website file, and link to it there.

```
<script src="scripts/jquery-3.7.1.js"></script>
```

2. You can link to the file located on a **content delivery network (CDN)**.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
```

The benefits of linking to the file through a CDN is that CDNs provide worldwide distribution of the same file. Additionally, users who may have already downloaded the jQuery file will not need to redownload it. As for where to write jQuery, nothing changes. You can write jQuery commands in the same locations as you can JavaScript.

A JavaScript library is a set of pre-built JavaScript code, functions, and objects that are designed to simplify or enhance the existing JavaScript language and library, usually designed with a specific purpose in mind. The jQuery library enhances the JavaScript code by providing simplified functions that handle tedious and complex operations.

KEY CONCEPT

Many software and programming languages have multiple versions. Older versions are eventually deprecated and not supported. At the time of this writing, jQuery Core 3.7.1 is the current version of jQuery. You can find information about jQuery versions and updated version availability at releases.jquery.com/jquery. Regardless of the version used, the instructions above can still be followed.

TERMS TO KNOW

jQuery
A programming library that simplifies the syntax of JavaScript and provides numerous conveniences for front-end developers.

Content Delivery Network (CDN)
A site or network of sites that host resources online for distribution and delivery across the world.

2. jQuery Syntax

Since the purpose of jQuery is to simplify JavaScript code and syntax, the syntax of jQuery is extremely simple. To begin, you will select the HTML element to manipulate or update in some way. The jQuery syntax is $(selector).action().

- The $ is required syntax to let the browser know that jQuery is being used.
- The (selector) is used to find the specific HTML element. This can be in the form of HTML tags, HTML ids, HTML classes, and so on.
- The action() is performed on the element.

An example of jQuery being used is $(".names").hide() to hide all the elements that have the class "names." The arguments make use of CSS selectors in order to point to specific elements or objects. For example, if we want to select an element and insert new HTML content, we can do this using an appropriate attribute as the selector and then calling its .html() function.

EXAMPLE Using jQuery to add an event listener using the `on` property

```
<body>
  <main>
  </main>
  <button id="btn">Add Content</button>
  <script>
    $("button").on( "click", ()=>addContent() );
    function addContent() {
      $("main").html("<h2>New Content</h2>");
    }
  </script>
</body>
```

Let us take a closer look at what is happening in the code sample.

EXAMPLE We have a main section with no content and a button.

```
  <main>
  </main>
  <button id="btn">Add Content</button>
```

EXAMPLE Next is a script element that contains a jQuery command and a JavaScript function containing a single jQuery command.

```
  <script>
    $("button").on( "click", ()=>addContent() );
    function addContent() {
      $("main").html("<h2>New Content</h2>");
    }
  </script>
```

Since the first command is outside of any function, it will get called immediately. This command gets a handle to the button using $("button"). The command then applies an event listener using the .on() function and providing the event as a string and a callback function to be executed when the event is fired: `$("button").on( "click", ()=>addContent() );`

Since the on() function expects a callback as the second argument, we can provide an arrow function that makes a call to the desired function, addContent().

The addContent function gets a handle to the main element in the page and updates its inner HTML attribute using its html() method. html() allows you to update the element's content. In the example, we updated main with an h2 heading: "<h2>New Content</h2>."

Without jQuery, the command might look like this:

```
document.getElementsByTagName("main")[0].innerHTML = "<h2>New Content</h2>";
```

Although not significantly longer than jQuery, it is less error prone as the jQuery names and functions are simple and easy to remember and help reduce human error by requiring less typing.

Additionally, many of the common style properties can be easily accessed through a simple function call. For example, if we need to hide something from view, instead of needing to access the element's style property and setting either visibility to "hidden" or display to "none," we can simply select the element and make a call to the .hide() function, which will not only change the element's display property to none but also keep track of the object's previous display state. This way, you can make the element reappear by simply calling the .show() function, which will restore the element's previous display value.

Are .hide() and .show() not enough? How about .toggle(), which when called will toggle the value of display from the original value, such as "inline," to "none." A subsequent call to .toggle() on the same element will toggle the value from "none" back to the original value.

Still want more? CSS properties can also be modified in jQuery by using the .css(property, newValue) function and giving it the property name and desired value as two arguments. Here is an example:

EXAMPLE

```
$("main").css("display", "none");
```

And wait . . . there's still more! Inline CSS styles can also be added using the .attr() function, which takes the attribute name and value as the two arguments and can be used to add an HTML attribute to an element. In this case, will provide the attribute name of "style" and a value of "display: none."

EXAMPLE

```
$("main").attr("style", "display: none;");
```

The one caveat about using the .css() and .attr() functions is that jQuery will not keep track of the element's previous value; however, .toggle(), .show(), and .hide() will keep track using jQuery's internal code.

Another effective method that allows you to toggle entire sets of style properties is to use the .addClass(), .removeClass(), and .toggleClass() functions. This option eliminates the reliance on jQuery to keep track of previous states.

Finally, jQuery provides a convenient method of attaching a handler function to the event related to the page being ready. This simply means that you can trigger jQuery code only after the webpage has finished loading and rendering to the page. To do this, simply start with a basic jQuery function "$()" and place an anonymous or arrow function within the parentheses. jQuery automatically interprets this as "wait until the document is ready and then execute this command." This is helpful when you have code waiting on resources to be loaded and helps avoid issues of a code block attempting to add an event listener to something that has not yet been rendered.

---

3. Options

One of the additional benefits of using jQuery to perform style changes to elements is that we can define and pass a JavaScript object in order to affect different options within the function. For example, if we want to hide an element and we want the process to be animated (as opposed to blinking out of existence), we can create an options JavaScript object with the key of "duration" and a value of how long we want the animation to take place in milliseconds. So, if we wanted the element to disappear over the course of 2 s, then you would configure the options objects with a duration of 2000.

EXAMPLE Adding an animation to a visibility toggle method using the options object

```
let options = {          duration: 2000,
                         easing: "swing", };
function toggleHeadings()
{
     $("h2").toggle(options);
}
```

Each jQuery function has its own set of options that can be configured using the options object. Furthermore, we can even provide callback functions that can be executed when the animation begins (such as hiding or showing an element) and once the animation completes. This way, we can execute additional operations after or while an element is being toggled. In the following example, whenever the animation of an element is finished, an alert will open with the message "Animation complete":

EXAMPLE Setting up the callback function when the animation is finished using the options object

```
let options = {          duration: 2000,
                         easing: "swing",
                complete: ()=>{alert("Animation complete");}
};
```

LEARN MORE

While jQuery is not required to be a web developer, many organizations and development teams may utilize jQuery. As such, if you were going to be considered for a position, you would want to be familiar with jQuery. You can find a list of all functions on the jquery.com documentation website: api.jquery.com/

Furthermore, jQuery also has different documentation on other sites, such as everything related to interactions, effects, methods, and themes for the user interface. This can be found on the api.jqueryui.com site.

There is a lot more to jQuery, just as there is a lot left to learn about JavaScript. Utilizing tutorial and resource sites like w3schools.com and api.jquery.com can help you learn more about how jQuery can simplify your JavaScript code and improve your coding efficiency.

MAKE THE CONNECTION

You will use what you learned in this Challenge to complete Touchstone Task 3.1: Implementing Dynamic Features With JavaScript.

SUMMARY

In this lesson, you learned about the **jQuery framework for JavaScript** and about its benefits to you as a web developer. You learned about the simplistic **jQuery syntax** and how you can configure the behavior of various jQuery features by setting up an **options** object.

Source: This Tutorial has been adapted from "The Missing Link: An Introduction to Web Development and Programming " by Michael Mendez. Access for free at https://open.umn.edu/opentextbooks/textbooks/the-missing-link-an-introduction-to-web-development-and-programming. License: Creative Commons attribution: CC BY-NC-SA.

TERMS TO KNOW

Content Delivery Network (CDN)
A site or network of sites that host resources online for distribution and delivery across the world.

jQuery
A programming library that simplifies the syntax of JavaScript and provides numerous conveniences for front-end developers.