JavaScript – theory and tasks

The DOM

We can link a JavaScript file to HTML by including it as the src of a <script> tag inside of an HTML file, like this:

<script src='js/main.js'></script>

This line of code will link the file located at js/main.js. You can find this file in the file navigator by clicking the file button located at the top left of the code editor. Within the navigator, there's a folder named js, and within that folder is the main.js file.

By linking js/main.js in the index.html file, we are asking the browser to run our JavaScript code each time index.html loads.

We've provided you with a sample website (and the corresponding HTML and CSS code). Our goal: use JavaScript to make this page more dynamic. We will add interactive features to it as we go through the lesson.

In the code editor, we've loaded the files for a static HTML and CSS website. If you've never seen HTML before, don't worry, we'll walk through how JavaScript is added to an existing HTML and CSS project in this lesson. This lesson won't require you to greatly modify the HTML and CSS code itself.

***Instructions***

1. Let's start by writing some JavaScript that we will soon link to our HTML document.

Click the folder icon attached to the code editor, and navigate inside the js folder and open *main.js*.

Inside *main.js*, write:

alert('Hello JavaScript!');

***Note:*** alert is a JavaScript function that will create a pop-up window with text inside it. When we link the *main.js* file to the HTML file, we will see a pop-up window that was generated by this code. alert is not used by JavaScript developers in practice, however it is useful here to demonstrate linking these two files. We will use it over the next few exercises for demonstration purposes.

1. Navigate back to *index.html*. Before the closing </body> tag, create a link to the *main.js* file using a <script> tag.
2. We just connected JavaScript to an HTML file. When the HTML document loaded, it ran the code inside *main.js*, which created the alert pop-up.

The *Document Object Model*, commonly referred to as the *DOM'*, is the term for elements in an HTML file. Elements are any HTML code denoted by HTML tags, like <div>, <a>, or <p>. Let's use JavaScript to interact with the DOM.

We can select an HTML element with JavaScript by selecting its class attribute, like this:

var header = document.getElementsByClassName('example-class-name');

This would find an element like this in the HTML:

<div class='example-class-name'> ... </div>

***Instructions***

1. Let's see how JavaScript can select an HTML element.

Inside *index.html*, notice there's an <div> element with a class of skillset on line 10. Let's select this element with JavaScript.

Click on the folder icon to navigate to *js/main.js*.

Inside *main.js*, delete the alert you wrote in the last exercise. Then, create a variable named skillset and set it equal to the HTML element with the class skillset.

1. Under the skillset variable, use an alert to make the skillset variable's value into a pop-up.
2. The pop up showed that you selected an [object HTMLCollection].

This confirms that JavaScript can select HTML elements. Since we can select HTML elements, we can manipulate them.

Selecting and modifying HTML elements with plain JavaScript can be tricky. Luckily, there's an easier way. Click 'Next' to learn more.

We've just covered how to select HTML elements using the syntax: document.getElementsByClassName. This is verbose and clunky, however. If we were to select a lot of elements this way, our code would get dense and difficult to read.

Wouldn't it be nice if there was a simpler way to select DOM elements? As you might have guessed, there is!

To better interact with DOM elements, we can use a library. A library is a set of code that contains useful pre-written functions that help with certain tasks.

A great library for interacting with the DOM is *jQuery*.

jQuery is a library written in JavaScript. The syntax and functions it contains will help us interact with DOM efficiently. We'll walk through a few examples in the following exercises.

In order to use jQuery, we need to:

1. Include jQuery in our project. jQuery is a library, which means it is a set of code in a file, therefore we will need to link that file in our HTML in order to access it.

Once we link it in our HTML file, we can use its functions and syntax in our *js/main.js* file.

1. Once linked, we'll need to make sure our HTML is loaded before we run our jQuery and JavaScript code.

This will prevent our jQuery and JavaScript code from running before the elements they select are rendered.

***Instructions***

1. Since jQuery is a library of code, we need to include a link to it in our index.html file before we can use it. Before the closing </body> tag, right above your current <script> tag, include this code:

<script src='https://code.jquery.com/jquery-3.1.0.min.js'></script>

The link to jQuery needs to be above the link to the *js/main.js* file, which will give *main.js* access to the jQuery library.

1. Now that we've included jQuery, let's get it ready to run.

Navigate to *js/main.js*.

Delete or comment out your existing code in *js/main.js*. Then, write a function named main. The function should take no parameters and should have an empty block.

1. jQuery has a built in function to check if the page is ready before it will run our code. After the main function, write this code:

$(document).ready(main);

Notice that we put main inside the parentheses of ready. main here is a callback, which means that our code will wait until the document (in other words, the DOM) is loaded, or ready. When it is, then it will execute the main function. jQuery calls back to the main function, therefore it's a callback.

In the event that our HTML and CSS took 5 minutes to load, this code would wait until it loaded completely before running

With plain JavaScript we selected an HTML element with this code:

document.getElementsByClassName('skillset');

With jQuery we can select the same element with:

$('.skillset');

1. We can wrap any CSS selector, like class, id, or tag, with $('.example-class') to select it with jQuery.
2. The selectors jQuery uses are the exact same as CSS selectors. For instance, if there's an element with a class of supporting-text, you could select it with $('.supporting-text'). Another example, if an element had an id of 'header', you could select it with $('#header').

***Instructions***

1. Let's select the element with the class of skillset, but this time with jQuery.

In *js/main.js*, inside the main function, write a variable named $skillset. Set $skillset equal to a jQuery selector for the skillset class.

***Note:*** It is a common convention to name variables that hold jQuery selectors with a dollar sign $.

1. On the line below the jQuery selector for skillset, write an alert on $skillset.
2. The alert shows [object Object]. This is right and means that we successfully selected the same HTML element as before, but this time using jQuery.

Now that we can select an element with jQuery, we can use built-in jQuery functions to modify it. From here on, we will start building features into our skills website.

First off, it would be nice to make the page fade in when loaded.

To make a page fade in, it must first be hidden. We can hide elements with jQuery with a function named hide.

We can hide elements with jQuery, like this:

$('.my-selector').hide();

1. We attached the hide function directly to the jQuery selector.
2. The hide function will add the CSS property display: none to the DOM element from the page, which will hide it.

***Instructions***

1. We want all of our skills to fade in, so we need to hide the skills first. In *index.html*, the element around all of our skills has the class skillset.

Inside the main function, delete the $skillset variable and the alert you wrote in the last exercise. Then use jQuery to hide the skillset element. Do this by writing a selector for skillset, then attaching hide() to it with a period.

1. Notice that the skills have been hidden. hide added a CSS property of display: none to .skillset element, which hid the elements.

In order to fade in the skillset element, we can use a the jQuery function named fadeIn.

True to its name, fadeIn will fade an element in over a period of time in milliseconds. It looks like this:

$('.example-class').fadeIn(400);

1. We must start with an element that is not currently displayed on the page.
2. Just like before, we can attach fadeIn directly to a jQuery selector.
3. Within fadeIn's parentheses, we can specify how long we want the fade to last in milliseconds. 400 is the default.
4. The example code will fade in the '.example-class' element over 0.4 seconds.

***Instructions***

1. Under the line that hides the skillset element, write another jQuery selector for skillset.

Then, attach fadeIn to it. Your fadeIn should last for 1 second/1000 milliseconds.

The next feature we'd like to build is making the 'Recent Projects' clickable. When clicked, the button should show the individual projects, and when clicked again, it should hide the projects.

In order to make an element clickable, we need to write jQuery that listens to an element for a click event. jQuery can do this with an event listener function named on('click').

This function will wait for a click event, and when one occurs, it will execute a provided function. The syntax looks like this:

$('.example-class').on('click', function() {

// execute the code here when .example-class is clicked.

});

1. $('.example-class') selects an HTML element with the class example-class.
2. .on('click', function() { ... }) adds a click listener to the selector. When it's clicked the function will execute the code within its block.

***Instructions***

1. Our goal is to show our HTML projects when the 'Recent Projects' button in each section is clicked, and to hide them when we click it again.

Let's start by hiding the projects that are currently there.

Under your existing code in the main function, use jQuery to hide the elements with a class of projects.

1. Our elements are now hidden. The next step is to make the 'Recent Projects' button clickable.

Under the hide you just wrote, write a jQuery selector for the 'Recent Projects' button. Its class is .projects-button.

Then, attach on('click'), and provide it an empty function as the second parameter to on.

1. Now that the click listener is set up on the 'Recent Projects' buttons, let's make the projects appear when we click them.

To make our projects visible when the 'Recent Projects' button is clicked, jQuery provides a function named show, which is the opposite of hide.

To show an element, the syntax looks as such:

$('.example-class').show();

1. show is attached directly to the jQuery selector.
2. show will change the CSS attribute display: none to a visible display property, therefore showing the element.

***Instructions***

1. Inside the click function you wrote in the last exercise, write a selector for the projects class.

Then, call the show function on it.

1. Click on the *'Recent Projects'* buttons, and notice that the projects now show when clicked.

Notice that there are some problems though:

1. The projects don't hide again when you click the button again.
2. When you click one button, all the projects show in each section.
3. It would also be nice if the button we clicked looked selected, or changed appearance when clicked.

We will solve these problems in the upcoming exercises.

When we click on a 'Recent Projects' button, the projects show. Next, let's hide the projects if we click the 'Recent Projects' button again.

jQuery provides a function named toggle that is helpful in this situation. toggle will hide or show an element, each time it is triggered. The syntax looks like this:

$('example-class').toggle();

1. toggle can be called directly on an jQuery selector.
2. When toggle is executed, it will hide or show the element that the selector points to. If the element is currently hidden, toggle will show the element, and vice versa.

***Instructions***

1. Inside the click function, we wrote a selector for the projects class and we called show on it.

Replace the show function with toggle.

Then 'Run' your code and click on the 'Recent Projects' buttons multiple times.

Let's add one more feature: when we click the 'Recent Projects' button the background color and text color should change.

We can toggle a CSS class with a jQuery function named toggleClass. The syntax looks like this:

$('.example-class').toggleClass('active')

1. .toggleClass is a function that will toggle a CSS class on the jQuery selector it's connected to. If the element has the class applied to it, toggleClass will remove it, and if the element does not have the class, it will add it.
2. 'active' is the class that we will toggle on and off. Notice that toggleClass does not require us to include the period before 'active' since it's already expecting a CSS class.

***Instructions***

1. In *css/styles.css*, there is this class:

.active {

background-color: #333333;

color: whitesmoke;

}

Inside the click function, toggle this class on the elements with the projects-button class.

The .active class will make the projects-button's background dark and its text light.

1. Click on the 'Recent Projects' buttons. Now they change color when clicked, and change back to their original state when clicked again.

There's still one big issue: we only want the element we clicked on to toggle its projects and class.

In the last exercise, we were toggling every 'Recent Projects' button instead of only the one we clicked on.

We can select the specific element we clicked on with the jQuery selector $(this).

The syntax looks like this:

$('.example-class').on('click', function() {

$(this).toggleClass('active');

});

1. $(this) selects the clicked element. If there are multiple elements with a class of .example-class, this will only toggle the class of the one that is clicked on.
2. Notice that $(this) does not require quotes around it, since it is not a CSS class. Instead, this is a JavaScript keyword.
3. $(this) behaves just like our other selectors. We can attach toggleClass or toggle to it in the same way.

***Instructions***

1. Let's begin by only changing the class of the element we clicked.

Modify the toggleClass we wrote in the last exercise to use $(this) as its selector.

1. Now click on the 'Recent Project' buttons within each section and see that only the button you click on will toggle its class.

Next up, let's toggle the projects in the section we clicked on, instead of toggling them all.

In order to toggle the projects in each section, we will need to use $(this) to select the button we clicked on. The issue is that $(this) refers to the projects-button in our current code, and not the projects themselves.

We need a way to select the projects elements next to the button that we clicked.

Luckily, jQuery can select elements logically. In *index.html*, notice that the projects-button element is directly followed by the projects list. Therefore the projects element is next after it.

jQuery has a function named next to help us select elements that are next to another element. If we have this in our HTML:

<div class='item-one'> ... </div>

<div class='item-two'> ... </div>

If we wanted to hide item-two, we could write:

$('.item-one').next().hide();

1. We can attach next to any jQuery selector to select the next direct element.
2. Then, we can attach any jQuery function to next(). In this case, we attached hide, which would hide the next element after the $('.item-one') element.

***Instructions***

1. Inside the click function, let's modify this line:

$('.projects').toggle();

Instead of selecting all the projects elements, use $(this) and next to select the projects, then attach toggle on the end to toggle the projects on the page.

1. When you click on the 'Recent Projects' buttons now, only that section's buttons projects toggle.

Since we have a few areas to click on, it may be helpful to show users which sections they have viewed by changing the text inside the 'Recent Projects' buttons.

When a user clicks on a button, we will permanently change the text of the button to 'Projects Viewed'.

We can change the text of an element with the jQuery function text. It's syntax looks like this:

$('.my-selector').text('Hello world!');

1. text attaches directly to a jQuery selector.
2. Inside of text's parentheses, we can provide text that will become the text of our DOM element. The text we supply will replace any existing text, and if the element has no pre-existing text, text will add it.

***Instructions***

1. Within the projects-button click function, under the toggleClass line, write jQuery to change the text of the button that was clicked to say 'Projects Viewed'.
2. Now click on the 'Recent Project' buttons and notice that they now turn to 'Projects Viewed' after each click. Very nice!

The last feature we'd like to add is an aesthetic one. Right now when we click the 'Recent Projects' buttons, the projects appear instantly.

Let's instead make the projects slide onto the page when we click the 'Recent Projects' button and then slide off the page when we click the button again.

jQuery provides a method named slideToggle that can animate an element's entrance and exit. The syntax looks like this:

$('.example-class').slideToggle(400);

1. slideToggle can be called directly on a jQuery selector.
2. slideToggle also takes a parameter of milliseconds that the animation should last. The default is 400 milliseconds, or 0.4 seconds.

***Instructions***

1. Let's make our projects slide in and out when we click on the 'Recent Projects' button.

Inside the click function, delete or comment out this line in your code:

$(this).next().toggle();

This line can prevent the slideToggle from working properly, since it is also affecting the showing and hiding behavior of the projects element.

Then, select the projects element of the button that is clicked. Use slideToggle on the selector to animate its appearance and exit on the page. The animation should last 400 milliseconds.

1. Now the projects slide in and out of the page when we click the 'Recent Projects' buttons.

By using jQuery, we've made this page much better by adding interactive elements.

jQuery is a complete library, and we've only covered the basics.

In this lesson we learned:

* How to link a JavaScript file to an HTML file using a <script> tag.
* jQuery is a library to help JavaScript interact with HTML elements.
* We can make sure our page is ready to go with $(document).ready(). Then, we can pass in a function to ready that will execute when the page is loaded.
* jQuery uses the same selector names as CSS.
* We can hide elements with hide, and show them with show.
* We can make elements appear with fadeIn.
* on('click') functions allow us to make HTML elements clickable. When an element is clicked, the click function will execute the function we provide. It's full sytnax looks like:

$('.example-class').on('click', function() {

// Execute when .example-class is clicked

});

* toggle will toggle an element on and off the page.
* $(this) will select the specific element that was clicked if placed inside a click function.
* toggleClass can toggle a class on and off.
* We can select elements next to each other with next.
* text will replace a DOM element's text with text we specify.
* slideToggle will make an element slide into and out of the page with an animation.

Impressive work on completing Learn JavaScript!

The next Javascript course, Intermediate JavaScript, is coming soon! In the course you'll learn how to write full-fledged programs in JavaScript.