Intro to Front End - Intro to JQuery

Set up the Live Stream

Front End Fundamentals

How's Javascript going?

Intro to JQuery

Today’s Objectives:

- Create and manipulate elements on a page using jQuery (using show and hide methods)
- Collect user Input (HTML Forms)
- Use more of Javascript’s built-in functions

Starter Code

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="./index.css" type="text/css" />
    <link
      href="https://fonts.googleapis.com/css?family=Montserrat:400,700"
      rel="stylesheet"
    />
  </head>

  <body>
    <h1>jQuery Practice</h1>

    <main id="“content-box”">
      <div class="first-square"><span> 1 </span></div>

      <div class="second-square"><span> 2 </span></div>
    </main>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
    <script type="text/javascript" src="./index.js"></script>
  </body>
</html>
```

```
html {
  font-family: "Montserrat", sans-serif;
  background-color: #0299cc;
}

body {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}

main {
  display: flex;
  justify-content: center;
  align-items: center;
}

.first-square {
  width: 100px;
  border: 1px solid black;
  padding: 3px;
}

.second-square {
  width: 100px;
  border: 1px solid black;
  padding: 3px;
}
```

Javascript Review

Console Log a helper function that prints out to the console;

Data Types: the values we use in javascript are all of a type and these are:

- Numbers: we can add, subtract, multiply and divide numbers in JS;

```
console.log(1 + 1);
console.log(10 - 5);
```

- Strings: often represent names, email addresses in code;

```
        console.log(“123”) // not a number!
```

- Booleans: values that are true or false;

```
console.log(true);
console.log(false);
```

Variables: store values for reuse;

```
var name = “Lighthouse Labs”

console.log(name)
```

Functions: help us store instructions and actions for reuse;

```
function isOdd(num) {
  return num % 2 !== 0;
}

console.log(isOdd(10));
console.log(isOdd(11));
```

If statements: allow our code to do different things based on conditions

```
var age = 15

if (age >= 13) {
    console.log(“You should be in high school!)
}
```

Objects: organized sets or related information stored as key-value pairs. We can use dot or bracket notation to retrieve values;

```
var person = {
    name: “Sherlock Holmes”,
    age: 60,
    occupation: “detective”,
    connections: [“Watson”, “Detective Lestrade”, “Moriarty”, “Irene
Adler”]
}

console.log(person)
console.log(person.name)
```

Arrays: lists or collections of stored values. Bracket notation is used to retrieve values;

Loops: allow us to repeat the same action or code to each item in an array or object;

Write a for loop to print the array - then replace all even numbers with the word 'even'

```
var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

for (var i = 0; i < numbers.length; i++) {
  console.log(numbers[i]);
}

for (var i = 0; i < numbers.length; i++) {
  if (numbers[i] % 2 === 0) {
    console.log("even!");
  } else {
    console.log(numbers[i]);
  }
}
```

The DOM

So now that we not how to write HTML and how to write Javascript... How do we put them together?

When a browser loads an HTML page, it builds a structure from the HTML called a Document Object Model or the DOM. The DOM is like a tree! When we change a pages style or html in dev tools, we are changing the DOM.

Javascript is very good at accessing that data structure and changing it for us!

What is JQuery?

jQuery is a very popular javascript library for making webpages interactive. It provides developers with many “helper methods” that perform tasks like hiding elements when they are clicked.

For review, a library is just a file we can put in our apps that contain a bunch of helpful functions that make common tasks easier to execute.

Add JavaScript

Before we can begin to use jQuery in our projects, we need to create a new JavaScript file and connect it to the HTML page.

Similar to CSS, we connect the files together using special tags

Live Code

Create a new javascript file in Atom

The Script Tag

To connect a javascript file to HTML, we use the script tag.

To use a separate JS file, add the following attributes:

- src: the path to the .JS file being used.
- type: indicates the type of script being connected. In our projects, we are using 'text/javascript';

Placement: In the `<head>` of the HTML page:

- The browser will not load any images or stylesheets until after the scripts are fully loaded when the tag is placed here which can make your sites load slower if the script files are large;

At the bottom of the `<body>`

- Allows the browser to load images and styles, making the webpage appear fully loaded even if the scripts are still downloading

Live Code

```
<script type="text/javascript" src="./index.js"></script>
```

Add JQuery

To add jQuery, the process is very similar. A CDN, or Content Delivery Network, link is added to the HTML page at the bottom of the body as this file can be quite large.

Live Code

Go to https://code.jquery.com/

Get link and paste at bottom of body tag

Getting Started with JQuery

Using jQuery in our projects requires us to select the elements we want to act on and then choose an action to perform on that element.

In jQuery the actions we perform are helper functions that we now have access to because we added the CDN link to the HTML page.

Before we start adding interactive actions, we'll go over how to start a jQuery file, and the ways we can select elements.

Ready?

For our jQuery functions to behave the way we want them to, we have to make sure that the entire webpage has been loaded first.

jQuery has a special helper function that will only run when the page is fully loaded, and therefore 'ready'.

The first 3 lines of code are the full version of this helper.

- $: a short form for query itself. We use it at the beginning of all the methods we write in jquery;
- document: refers to the DOM;
- .ready(): this is the function that will run your jquery when the document is fully loaded;

The second block is the short form of the same function and we will be using it. This short form exists because the document ready function is so common.

Live Code

Add both versions:

```
$document.ready(function() {
  //code goes here
});
```

```
$(function() {
  //code goes here
})
```

Selecting Elements

To select the elements we want to act on, we use methods similar to selecting elements in CSS.

- tag name
- class name
- id name

Selecting in jQuery

To select elements in jQuery, we start with the $, the shorthand for calling on jquery.

Inside brackets and surrounded by quotation marks, we select elements using the same syntax we used in CSS.

In this example, we're selecting by:

- Tag Names
- Nested Tag Names
- Classes: use a " ." to select;
- Ids: use a " # " to select

Live Code

```
$("h1");
```

```
$(".first-square");
```

```
$("#content-box");
```

---

jQuery Animations

Previously we have seen how CSS can animate elements on the page. We can do the same things with jQuery!

Each of the helper methods we use will take an argument which represents the amount of time our animations will run, similar to the ‘transition-duration’ property in CSS.

Hide and Show

Show and hide will either display the selected element or hide it. The value between parentheses is the duration of the animation and as always, we start with the ‘ $ ‘ jQuery shortcut.

Live Code

```
        $("main").hide(1000);
```

```
        $("main").show(3000);
```

Slide Up and Down

These helpers also show & hide elements by sliding them in or out from the top of the screen.

Live Code

```
        $("main").slideUp(1000);
```

```
        $("main").slideDown(3000);
```

Delay

All of the animations we have seen so far happen as soon as we refresh the page. They delay helper takes a duration argument and waits to start the queued animations for that amount of time.

Live Code

```
        $("main").delay(2000).slideUp(1000);
```

```
        $("main").hide(1000).delay(2000).slideDown(3000);
```

Event Listeners

So far, all the animations we've used have happened as soon as we refresh the page. Usually, we want animations or interactions to run when an 'event' happens.

Events are form submissions, button clicks, hovering over elements and many more.

We can use jQuery to listen for these events and then run some code when they occur.

Click Events

To attach a click event to an element, use the .click() helper which accepts a function as an argument.

That function will contain the action you want performed after a click event has occurred.

Live Code

```
$('main').click( function() {
  $('.second-square').slideUp(2000);
 })
```

Mouse Events

Another common event to attach actions to are those caused by the mouse entering or leaving the space an element takes up.
These are 'mouseenter' and 'mouseleave'.

These events are similar to the CSS ' :hover' state, we are now using javascript to override the browser's default styling!

To demonstrate these events, we will be using a condensed version of the jQuery event helper function which uses the .on() helper function.

It takes an event name as a string and then another function will describes the actions to be performed.

Live Code

```
$(".first-square").on("mouseenter", function() {
  $(".first-square").hide(1000);
});
```

```
$(".second-square").on("mouseenter", function() {
  $(".second-square")
    .hide(1000)
    .show(2000);
});
```

Forms

In our projects, we will be using a small form to submit a query and the external data sources will return data that matches.

When we submit a form, all the data the user has entered is gathered up and submitted to the backend.

There are a million ways to handle form submissions but we will be using jQuery.

Forms Contd

To create a form, HTML has a semantic `<form>` tag!

Inputs are the tags which will actually collect the user’s input. We change the type attribute to change the type of input we are using. We will be using ‘text’, the most common.

Live Code

```
<form><input type="text" /> <button type="submit">Submit</button></form>
```

Form Submission

We are going to use jQuery to handle form submission and turn off some of the browser’s default behaviour.

jQuery, of course, has a helper function for this called, .submit() which accepts a function containing the actions to be performed. On submit, the browser will refresh the page so we add event.preventDefault() to prevent the default behaviour of the submit event.

In the code example, we use another helper function, .val(), which collects the content the user inputted.

Live Code

```
$("form").submit(function() {
  event.preventDefault();
  console.log($("input").val());
});
```

Resources

MDN “First HTML Form” tutorial

jQuery Documentation

Target Elements, jQuery Practice

Use jQuery to modify page

Reminders

- Record Attendance
- Upload lecture
- Send out Notes
- Log hours on Clocktower