Capstone Project - Phase 3 of 6 (jQuery)

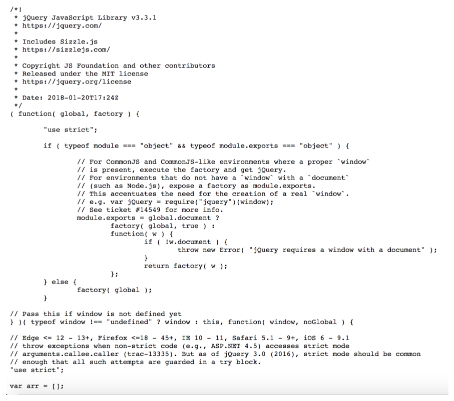
Getting Started

Recall that you already have both a local and a remote repository for your Capstone Library project. In Phase 1 of the Capstone Project, you built the engine that will drive your application using JavaScript and in Phase 2, you built the User Interface for your digital library application. **Before you begin Phase 3, please merge your final copy of Phase 2 into your develop branch if you have not already done so.** As we enter Phase 3, we will create a new feature branch within our feature folder, titled **phase-three**. Your task is to hook your Phase 1 methods to your Phase 2 buttons, inputs, displays, and table to create dynamic interaction between the two. To do this, you will need to focus in on the init/bind structure we've talked through. We will use what we have learned in JavaScript thus far to initialize our application upon page load (many of you already have this, as a document.addEventListener statement) and bind the selectors that have been initialized to their event-handlers using a bindEvents() method. At this point, Kyle has walked you through doing some of this, and moving forward, you should be able to continue accessing various elements using jQuery selectors and adding to their functionality through your event handlers in the same manner.

Set Up

As you get ready to update your file structure, you will first need to download jQuery (https://jquery.com/download/). The uncompressed, development version of jQuery 3.3.1 is recommended: Download the uncompressed, development jQuery 3.3.1

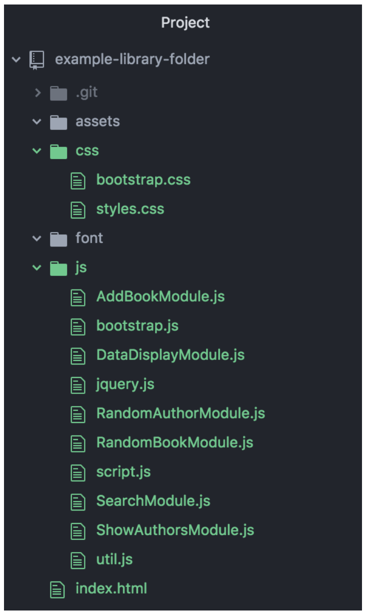
Unlike Bootstrap, which gives you a zip file to pull source code from, jQuery will simply give you the file in a browser like this:



```
/*!
 * jQuery JavaScript Library v3.3.1
 * https://jquery.com/
 *
 * Includes Sizzle.js
 * https://sizzlejs.com/
 *
 * Copyright JS Foundation and other contributors
 * Released under the MIT license
 * https://jquery.org/license
 *
 * Date: 2018-01-20T17:24Z
 */
( function( global, factory ) {

	"use strict";

	if ( typeof module === "object" && typeof module.exports === "object" ) {

		// For CommonJS and CommonJS-like environments where a proper `window`
		// is present, execute the factory and get jQuery.
		// For environments that do not have a `window` with a `document`
		// (such as Node.js), expose a factory as module.exports.
		// This accentuates the need for the creation of a real `window`.
		// e.g. var jQuery = require("jquery")(window);
		// See ticket #14549 for more info.
		module.exports = global.document ?
			factory( global, true ) :
			function( w ) {
				if ( !w.document ) {
					throw new Error( "jQuery requires a window with a document" );
				}
				return factory( w );
			};
	} else {
		factory( global );
	}

// Pass this if window is not defined yet
} )( typeof window !== "undefined" ? window : this, function( window, noGlobal ) {

// Edge <= 12 - 13+, Firefox <=18 - 45+, IE 10 - 11, Safari 5.1 - 9+, iOS 6 - 9.1
// throw exceptions when non-strict code (e.g., ASP.NET 4.5) accesses strict mode
// arguments.callee.caller (trac-13335). But as of jQuery 3.0 (2016), strict mode should be common
// enough that all such attempts are guarded in a try block.
"use strict";

var arr = [];
```

Simply use Command + A to select all of the text. Then use Command + C to copy it all. Create a new file within your js folder for jquery.js and paste this code into it directly using Command + V.

Your file structure at this point should look like this: index.html at the root, a css folder with bootstrap.css and styles.css, a js folder with bootstrap.js (or bootstrap.bundle.js), script.js, and jquery.js, an assets folder for later images, and a font folder with icons/fonts for later, etc. For further containerization, you will add a js file for each module so you file structure may look something like this (we may need to add on or amend this later, so use it as a guideline only):



```
Project
example-library-folder
  .git
  assets
  css
    bootstrap.css
    styles.css
  font
  js
    AddBookModule.js
    bootstrap.js
    DataDisplayModule.js
    jquery.js
    RandomAuthorModule.js
    RandomBookModule.js
    script.js
    SearchModule.js
    ShowAuthorsModule.js
    util.js
  index.html
```

Remember to link your new file to your index.html file by writing a <script></script> tag that includes the name of the external file where that code lives now. The order also matters. Within the script tags above the end of your body, jQuery should run first, then bootstrap.js, and lastly, your own script.js file.

- Must use: bootstrap.css, bootstrap.js, jquery.js

Requirements

- In the module that will allow users to add book objects, you will utilize your **addBook(book)**, and possibly your **addBooks(books)**, method from Phase 1

- Ensure that the 5 input fields of your module (cover as a base64-encoded string, title, author, number of pages, publish date) are linked with your Add Book button's click event
    - For further information on base64 encoding:
    - I would suggest locating the cover artwork for your book instances, saving their images as .jpeg or .png files, and adding them to your assets folder so they are stored down locally
    - Then, you can use the second link below to convert them to base64 encoded images for use in your Add Book input field
    - https://blogs.oracle.com/rammenon/base64-explained
    - https://www.base64-image.de/
- To make your Add Book module dynamic, your display area should show the option to add one book or multiple books at a time, by displaying a new form or form-type element or by having a counter/queue set-up for tracking how many books will be added.
    - You will most likely need to leverage your **addBooks(books)** method here

- Include a table that displays each book's information
    - You will need to recreate the hard-coded table you have now by identifying the DOM elements that serve as placeholders for the data you want in the table, and are passing in through your inputs, and replacing them with new values using jQuery (Template Strings would be a good thing to look up for this section. Template Literals are also helpful, but you may not use them until we refactor into ES6)
    - Each row will still have information for a single book object, but the data will be populated dynamically using jQuery
        - A delete (x) button in the rightmost column of the table should remove the entire row of data from the table and from your bookshelf array
        - The delete functionality should leverage the **removeBookByTitle(title)** method
        - **Bonus items:**
            - The ability to edit a book's title
            - The ability to edit a book's author
            - You may need an edit button to accomplish this

- Link your display area to show all of the authors available within the library to your button for doing so

    - Leverage the **getAuthors()** method here
- Link your module with an input field/button that will allow you to remove books by a single, unique author to your button for doing so
    - Leverage the **removeBookByAuthor(authorName)** method here
- Link these buttons to the appropriate data
    - Return a random book object by leveraging the **getRandomBook()** method here
    - Return a random author from your library by leveraging the **getRandomAuthorName()** method here

- Link the input field and button for searching through your library to your data using jQuery
    - Leverage the **getBookByTitle(title)** method here
    - Leverage the **getBooksByAuthor(authorName)** method here

- Should be mobile-responsive and have the appropriate metadata information included in the head
- Should have scripts that run just before the </body> tag to match industry standards

What to Do

1. Review the requirements
2. Plan out how you think you can best accomplish these targets and determine various approaches you can try, research and read JavaScript and JQuery documentation from Helpful Materials below
3. Start writing your code
4. Discuss issues with your peers and ask for help if needed
5. Commit and push every time a new module is completed and functional

Helpful Materials

-https://www.w3schools.com/jS/js_htmldom.asp
-https://www.w3schools.com/jS/js_htmldom_eventlistener.asp
-https://www.w3schools.com/jquery/default.asp
-https://api.jquery.com/
-https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals

Testing

- Compare your code to the rubric and clean it up for code review

Submitting

- Add Kyle and Shambre as collaborators (kbrothis and ShambreSW) on GitHub
- Project due at **9:00 am on Wednesday, July 18th, 2018**
    - If you do not think you can finish by this time, Kyle and I must receive a Slack message from you saying so by **9:00 pm** on Tuesday, July 17th, 2018 (it's okay to tell us you won't finish and then discover that you are able to, but we want to know if this pace is too strenuous and adjust if necessary)