Client Web App

WARNING: This assignment is a test of following directions.

This setup is required:

Install Google Chrome.
Install Visual Studio Code.
Go through the Node.js tutorial ***BUT***:
When you get to the Express part, use "web" as your application name:
```
express web --view pug
```
https://code.visualstudio.com/docs/nodejs/nodejs-tutorial

After you are done the tutorial, you should have a folder called "web" that is a functional Express Node app.
Inside "public/stylesheets" you should see a CSS file. This is where any CSS changes should go. Use this CSS file! Don't just in-line HTML your styling changes.
Inside "routes" you will see 2 JS files that handle routing. These should remain untouched.
Inside "public", create a new file "index.html".
Start simple: create an HTML file that just says "Hello world". With npm running (npm start), you should be able to refresh localhost:3000 in your browser to see the change.

Your goal for the HTML:

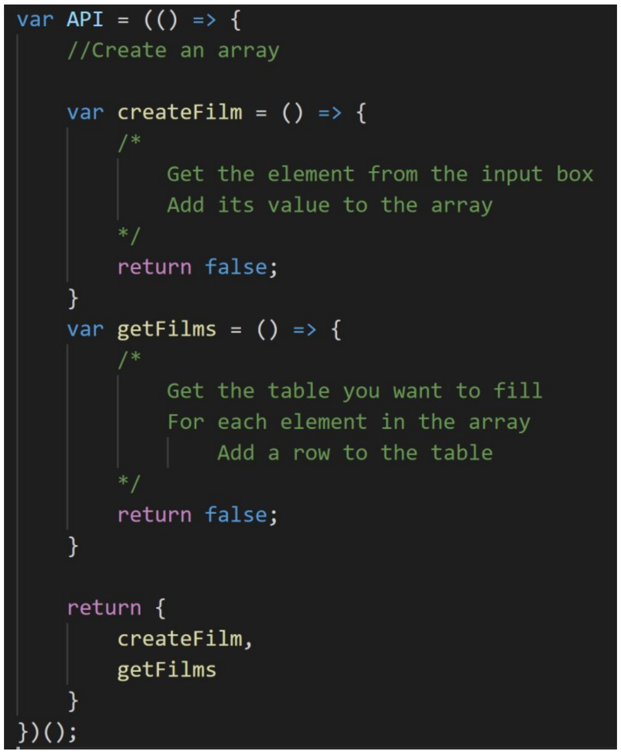
1. The user must be able to add a film. Ask the user for a film title and provide a button to submit.

2. The user must be able to get all films they have added. Provide a button that the user can click. Once clicked, present all of the films they have added, along with a rating in a table. One row for each film. For now, give all movies a perfect score. You can determine the rating scale. You can determine how it is presented. Part of your grade will be presentation, appearance, prettiness of this web page. Don't go overboard!

This is the HTML + CSS part of this assignment.

Inside “public”, create a new file “films.js”. This file will be responsible for the logic behind the buttons created in the HTML.
The button that creates a film will add the supplied title into an array. It will clear the input form to allow for a new submission. Lastly, the user will receive some sort of verification that the submission went through successfully.
The button that gets films will update the table with all films in the array. The table should be empty (invisible) prior to clicking the button, or if the array is empty.

Below you will find the core code, with comments for the lines you have to code. This is structured in such a way that each function is neatly defined. The array will have scope that is shared between both functions. Lastly, and most importantly, API can be accessed directly from the HTML. We “return false” in order to tell the button not to actually submit the form, which would refresh the page.



```
var API = (() => {
    //Create an array

    var createFilm = () => {
        /*
            Get the element from the input box
            Add its value to the array
        */
        return false;
    }
    var getFilms = () => {
        /*
            Get the table you want to fill
            For each element in the array
                Add a row to the table
        */
        return false;
    }

    return {
        createFilm,
        getFilms
    }
})();
```

In the HTML, include the JS file: <script type="text/javascript" src="./films.js"></script>
In the HTML, the form that submits to a create film should be: onsubmit:”return API.createFilm()”
In the HTML, the form that submits to get films should be: onsubmit:”return API.getFilms()”

Educator Notes:
It is important to understand that in a client-server architecture, it is not correct for the client to hold the film data. This is just a first step in a series of assignments. Because the films are stored in the client JS, they are completely non-existent to the outside world. You can’t retrieve them from anywhere else other than your browser on your computer. We will be improving on this throughout the course.

Rubric:

Functionality (meeting the requirements, following directions) – 90
Custom CSS + HTML + JS (making things pretty) – 10

Total: 100

Submission

For each assignment in this course, you will be required to create a video (with your voice, but not your face) of your finished product. The video should include:

- A quick walkthrough of your code and how it works, especially if there are any nuances or cool features added.
- A walkthrough of your finished product, demonstrating its functionality.

For this assignment, please demonstrate the following functionality:

- Getting films does nothing before you add any films
- Adding a single film and then getting films displays that single film and the maximum possible rating in a table
- Adding multiple films and getting films displays all films and the maximum possible rating in a table, 1 row for each film

Your goal with your voice is to demonstrate to me that you understand what you did in this assignment. No voice track = no grade. Doesn't have to be radio quality scripted content here, just a quick demo like one you might send to your manager demonstrating a prototype.

The video should be no longer than 10 minutes. Aim for less, like 5 minutes.

You can use Kaltura to record your video. But if you use some other tool, that's fine too. Either way, the end result must be updated to Kaltura.

https://docs.cci.drexel.edu/display/CD/Kaltura+Drexel+Streams+--+Adding+Media+to+Bb+Learn+--+Student+Instructions

Your final submission for this assignment:

- Link to Kaltura video
- HTML file
- CSS file
- JS file

No video = no grade. ***No voice track = no grade.***