IFQ714: Introduction to JavaScript Programming
Assignment 2: Programming Project

Matthew Campbell
27/03/2024
Queensland University of Technology

In this report I will outline my approach for completing assignment 2. Before I do so, I'd like to make some quick comments on the logic of my file structure to avoid any confusion. The merged dataset (for Step 1) can be found in main.js along with the mapping function and the code that writes a new json file of the merged data (mergedFlightsData.json). The "analysis" of the data (Step 3) can be found in the getFlightBy.js file. Another file contains a "route identifier" and flights counter that creates an object with distinct routes and a count of said routes. This data is then analysed in the calculatedRoutesData.js file. Please note that I have adapted some of the code in calculatedRoutesData.js in a separate DOM.js file for functions for buttons that appear on the webpage and that I've only kept the file for reference purposes. The work I did in the DOM (Step 4) can be found in the three DOM.js files. Finally, the Jest tests can be found in the tests folder.

Step 1: Loading data sources

Loading the data presented no difficulties, but I initially had a hard time finding a solution for merging the data. My initial function (commented out in main.js) was based on a sample solution shared by my OLA, but my solution ended up being insufficient because it was only returning details (like "city") for the source airport in the new array.

For a better result, I added separate constant variables for the source and destination flights that search through the airports JSON object with the .find method and match the airport ID with the source and destination airport IDs respectively. This I then mapped onto a new array which returns distinct flight information for source and destination airports as well as codeshare, aircraft and airline information.

Specifying explicitly in the return statement what kind of information I wanted to store in my new array provided a much more satisfying and useful set of data.

To create the new .json file with the merged data I followed the same approach as in assignment 1, noting however that this created some unintended consequences that I discuss in Step 5 (Unit Tests).

Step 2: Mapping function for databases

I acknowledge here that I don't think my mapping function meets the requirements outlined in Step 2, namely as they involve being able to "perform operations" on the data. I claim this because the function amounts more or less to a simple duplication of the merged data that I created in the previous step, with the addition of a flight number and the timestamp, the latter of which's utility being questionable given the state of my code.

For the function, I've taken the constant variable "mergedData" as a parameter, and using the .map method, mapped the data onto another data array which adds the flight number via a simple counter. I was a little overwhelmed looking at all the options available for creating a timestamp[1], but eventually landed on toISOString(), which returns the current UTC timezone.

1 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/toISOString

The rationale behind giving each flight a unique flight number was that I couldn't see an obvious way of fetching single flights based on the data as is. This becomes apparent when considering my analysis functions in the getFlightBy.js file, where most of the functions (such as getFlightBySourceAirport) return an array of flights based on a value specified as an argument in the function. Although the utility of having a unique flight number for each flight is not proven within the overall context of my completed work, I believe in some sense it's not hard to imagine how it could be useful for future analysis or for users interacting with the webpage or a webpage from a similar project.

The mapped data is finally written into a new .json file which is what is used for the actual "analysis", or displaying of the data.

Although my mapping function does what it's written to do, it does not enable me to perform any actual operations on the data as specified by the assignment instructions. The reason for this is that even at the time of writing this report, I'm yet to reach a complete understanding of how to approach this part of the assignment. Furthermore, by the time I realised the function wasn't exactly doing what it's supposed to do, I had already completed other parts of the assignment, and I was concerned that changing what I had done would lead to other parts of my code unravelling.

I am aware that in a "real life" situation (e.g. in a workplace) this might not be accepted, but for the time being I have to accept that it's not within my current understanding of JavaScript to fix the function.

Step 3: Data analysis

As mentioned, my analysis of the data appears in two files: getFlightBy.js and DOM_button_functions.js. I'll report on the former first.

The getFlightBy.js returns flights based on a given input, namely the flight number, the source and destination iata codes, the codeshare values, the airline name and aircraft types. Writing these functions was comparatively easy since they were similar to some of the functions completed for the first assignment.

One roadblock I came across was in writing the function "getFlightByFlightNum". The function makes use of a parameter flightNum and the .find method to return a flight when its number is passed into the function as an argument. The initial difficulty was in actually accessing the flight number information for the function since it needed to be typed in manually into the return statement and up until that point I was only familiar with accessing data objects with dot notation. This I was able to resolve when I familiarised myself with other ways of accessing properties, here with bracket notation[2].

The getFlightBySourceAirport and getFlightByDestinationAirport functions work on the same idea whereby an empty array is created inside the function, the data is iterated over with a for loop and "if" the source/destination airport iata matches the iata entered as an argument to the function, adds (using the .push method) all flights that have that source/destination iata to the empty array.

With a mind to varying my approach to creating functions a little bit, the getFlightByCodeshare function uses the forEach method instead of a for loop to iterative over the data, and again "pushes" the data into an empty array which is returned with either true or false, depending on what's specified as an argument to the function.

2 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Property_accessors

The getFlightByAirlineName and getFlightByAircraftType functions also work in the same way. The mergedFlightsDataMap array object is give as the first parameter and airlineName/aircraftType as the second. The data is then filtered with the .filter method which takes another function as an argument that only returns a new array of airline names/aircraft types based on the airline name/aircraft type specified as the second argument to the function. The aircraft type function differs only in that the .includes method is utilised because in the data itself, the aircraft type is presented as an array.

Additional analysis functions are found inside DOM_button_functions.js, alongside the DOM code I used to create the buttons for the webpage. It made sense to me to do it in this way because this was the only bit of "analysis" I decided to include for the interactivity part of this assignment.

To get the information on busiest routes and the average number of flights across all routes, I attempted to create two functions: one to create unique route identifiers and another to count the number of flights that occur in that route based on the name of the airport. For the function for creating unique route identifiers I've used a string comparison[3] (<) inside an if-else statement that returns either the source destination to/from the destination airport name and vice versa. This was an attempt to ensure that the count for each route was direction-agnostic (as in, Melbourne to Sydney would be the same as Sydney to Melbourne).

In the countRoutes function, an object (flightCounts) is initialised. The data is then iterated over using forEach and within this operation, a route variable is created using createRouteIdentifier as a callback. Using an if statement, the function checks if the route doesn't exist and if it doesn't, creates a new object with the source and destination airports and a flights count initialised to 0. The flights count is then incremented in the next part of the function body.

In the DOM_button_functions.js file there are functions written for calculating the busiest route, the top ten busiest routes and the average number of flights across all routes. As the values required for this analysis were stored in an object called flightCountsObj, I found the Object.values[4] method helpful in converting it to an array for easier access.

The showBusiestRoute and findTopTenBusiestRoutes functions both use the .sort(a, b) method in finding the busiest routes, the exception being that the latter uses .slice(0, 10) to return the ten busiest routes. The calculateRouteAverage function uses a for loop to iterate over the data and a simple equation for returning the average.

Step 4: Interactive web page

This part of the assignment was by far the most difficult for me as it was my first proper foray into writing asynchronous JavaScript and writing DOM code. For this section I will attempt to explain some of the difficulties I had and my approach for some of the functions. It should be noted that I feel my understanding of asynchronous JavaScript and using async/await is still fairly rudimentary, so I relied heavily on a mixture of going back to some of the modules for the course that use these elements, imitating some approaches I found on the internet and some experimentation. For the sake of brevity, I will refer only to the DOM_flightsData_functions.js file, which contains the DOM functions for interacting with the "Flights Data" part of the web page.

3 https://canvas.qutonline.edu.au/courses/1530/pages/1-dot-5-operators-numbers-and-strings?module_item_id=87173
4 https://www.shecodes.io/athena?tag=Math.max%28%29+method#:~:text=In%20order%20to%20use%20the,to%20find%20into%20an%20array

To create page interactivity for the "Flights Data" section of the web page, I decided to work with my merged flights data JSON, which I imported asynchronously following the examples we worked with in Code-play 21[5]. The easiest method I could find for working with the data within my functions was simply to create a global variable, which is assigned its value (the data) inside the async function. Also inside the async function are the functions for populating the dropdown boxes[6]. This was an initial source of frustration for me because I was trying to call the functions just after declaring them, in effect trying to get them working synchronously in an asynchronous program. I suspect this had something to do with the fact that these functions in particular take "data" as parameters and that the data itself is being loaded asynchronously. The other functions are loaded separately within the window.onload function[7].

Two separate functions were created for each dropdown box for two levels of interactivity: one for seeing a populated list in the dropdown box when it's clicked and another for displaying information when a particular item is chosen. The two different types of function follow the same pattern for each dropdown box.

The functions for populating the dropdown boxes initiate an array for storing the information. The data is iterated over using the .forEach method which contains a function with an if statement which establishes if the "airportName" is included in the array, and if not "pushes" the airport name into the array. Since the HTML element being used is a "select" element, I assigned a variable "option" to "option" using the .createElement method[8], which then takes the value of airportName and is appended using the .appendChild method to the filter div.

The functions for displaying the information for the items when selected follow this pattern:

```
function selectSourceAirport() {
   const selectSource = document.getElementById("filterSourceAirportSelect");
   const selectedAirport = selectSource.value;
   const airportInfoDisplay = document.getElementById("flightFilterDisplayDiv");
   airportInfoDisplay.innerHTML = " "

   document.getElementById("filterDestinationAirportSelect").selectedIndex = 0;
   document.getElementById("filterAirlineSelect").selectedIndex = 0;
   document.getElementById("filterAircraftSelect").selectedIndex = 0;

   const infoFromSelectedAirport = mergedDataGlobal.filter(flight => flight.source_airport.name
=== selectedAirport);
      if (infoFromSelectedAirport.length > 0) {
         infoFromSelectedAirport.forEach(flight => {
         const info = document.createElement("p");
         info.textContent = `Destination: ${flight.destination_airport.name}; Airline: $
{flight.airline.name}; Aircraft: ${flight.aircraft}`;
         airportInfoDisplay.appendChild(info);
      })
   }else {
      airportInfoDisplay.textContent = "No flights found for the selected source airport."
   };
```

5 https://canvas.qutonline.edu.au/courses/1530/external_tools/retrieve?display=in_rce&resource_link_lookup_uuid=0907af80-c9df-42a1-9c3c-884ebb0c03f5

6 https://raddy.dev/blog/javascript-async-await-fetch-and-display-data-from-api/

7 https://bito.ai/resources/javascript-windowonload-javascript-explained/#:~:text=Table%20of%20Contents-,Javascript%20Window.,DOM%20(Document%20Object%20Model).

8 https://www.w3schools.com/jsref/dom_obj_option.asp

```
    };
```

Two points of difficulty I had here were clearing the display once a different route is selected and restoring the text in the dropdown box to its default value ("Any"). The former issue I was able to solve simply by having an .innerHTML method with with empty quotation marks[9]. The latter issue was solved with the .selectedIndex method[10].

Step 5: Unit tests

I've attempted to include some graceful error handling in the functions that either load data, read data or write data, including some try/catch blocks in the DOM files, but error handling is otherwise largely missing from other portions of my code. It was enough of a challenge for me getting the code to work on some level before submitting this assignment, so for now I accept that this is an area (among many) that I will need to improve upon.

For testing functions I wrote a few tests using Jest, namely for my merging function and for a couple of the getFlightBy functions. Trying to figure out how to test a function that merges two arrays of data opened up a whole new world of confusion for me because up until this point I'd only written Jest tests for returning single vales (like the maximum value in an array). The best I could do in the end was to create a "mock array" of data (found in mockData.js) that imitates the flights and airports JSON data and then tests the function (within main.test.js) for merging the data. In the test statement I entered the expected data output manually: a time-consuming an error-prone task as far as getting the syntax right was concerned. In the expect statement, I made use of the toEqual method because of its utility in working with arrays[11].

getFlightBy.test.js contains tests for the getFlightBySourceAirport and getFlightByDestinationAirport functions. For these tests I merged the two mock data arrays using the same method used originally to merge the two JSON data sets, and then used the merged array object in the tests.

One issue that arose when running "npm test" — and this is a similar issue that blighted my project at almost every turn — occurred when running the tests and receiving errors in other parts of the project, namely where I've written files:

9 https://www.tutorialspoint.com/how-to-clear-the-content-of-a-div-using-javascript#:~:text=Using%20the%20innerHTML%20Property&text=When%20we%20assign%20an%20empty,element%2C%20including%20the%20div%20element.

10 https://stackoverflow.com/questions/49588862/how-to-reset-values-of-select-dropdown

11 https://nidhisharma639593.medium.com/a-guide-to-jest-matchers-understanding-and-using-the-essential-matchers-42a8ae284a2#:~:text=toEqual()%20matcher%20in%20Jest,are%20equivalent%2C%20and%20false%20otherwise.

I gather this had something to do with writing the files synchronously, but when I tried to change the code to write the files asynchronously, the functions seemed not to work at all.

Additionally, an unintended consequence of running the Jest tests was that my merged data and flights data map JSON files were being written anew in the tests folder. It is beyond the scope of my current understanding of JavaScript to fix this.

Step 6: Functional JavaScript

I've tried to write pure functions where possible, but adhering to the principles of functional programming was a challenge. As far as I can gather, my getFlightBy functions are pure functions, however for the most part I wasn't able to follow the principles and get the project done in a way that made sense to me and was within my capabilities at present. Perhaps the most glaring example of breaking with the functional programming principles was the creation of global variables in completing the interactivity part of the assignment. Unfortunately this was the only way I knew how to get this part of the assignment to work.

Conclusion

This was an interesting assignment that I intend on returning to and improving once I have the time. I was not able to complete it to a level that I'm entirely satisfied with, and that was due in part to not understanding some of the requirements, but also not having a clear idea of how I wanted to approach each step from the outset. Also, once the wheels were in motion with the various steps, I was reluctant to go back and change any of the code I had written when things stopped working in various places because my solutions were hard-won through hours of trial and error.

The biggest challenge was in understanding all the dependencies that were created once the project started to grow. A prime example is the one I mentioned above where running the Jest tests was creating new JSON files, but there were many other times in other parts of my project where a piece of code in one file was affecting another part of the project for reasons I still don't understand. An example of this was where at one stage running anything in calculatedRouteData.js in the console was removing all of the data inside my flightsDataMap.json file. The only workaround I could manage to find for this was writing new JSON files for any piece of data that a new part of the project was dependent on and reading it with the fs module. I'm confident that as my understanding of asynchronous programming improves, so will my ability to tackle these issues.

References:

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/toISOString
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Property_accessors
- https://canvas.qutonline.edu.au/courses/1530/pages/1-dot-5-operators-numbers-and-strings?module_item_id=87173
- https://www.shecodes.io/athena?tag=Math.max%28%29+method#:~:text=In%20order%20to%20use%20the,to%20find%20into%20an%20array
- https://canvas.qutonline.edu.au/courses/1530/external_tools/retrieve?display=in_rce&resource_link_lookup_uuid=0907af80-c9df-42a1-9c3c-884ebb0c03f5
- https://raddy.dev/blog/javascript-async-await-fetch-and-display-data-from-api/
- https://bito.ai/resources/javascript-windowonload-javascript-explained/#:~:text=Table%20of%20Contents-,Javascript%20Window.,DOM%20(Document%20Object%20Model).
- https://www.tutorialspoint.com/how-to-clear-the-content-of-a-div-using-javascript#:~:text=Using%20the%20innerHTML%20Property&text=When%20we%20assign%20an%20empty,element%2C%20including%20the%20div%20element.
- https://stackoverflow.com/questions/49588862/how-to-reset-values-of-select-dropdown
- https://nidhisharma639593.medium.com/a-guide-to-jest-matchers-understanding-and-using-the-essential-matchers-42a8ae284a2#:~:text=toEqual()%20matcher%20in%20Jest,are%20equivalent%2C%20and%20false%20otherwise.