IFQ714 Introduction to JavaScript Programming
Assignment 2: Programming Project
Jenny Guidetti
Queensland University of Technology

Introduction
This report looks at the use of JavaScript, Node and HTML to create a basic web page. Two datasets containing flights and airports information from OpenFlights were used for the code. The web page allows for simple filtering and analysis of this information with user interaction.This report covers how the data was loaded, mapped, analysed, used on the web page, tested and created with functional JavaScript in mind.

Step 1: Loading data sources
After downloading and saving the two datasets, two functions were created to import the information. Each function uses a 'try' and 'catch' block. The data is first read in synchronously using 'fs.ReadFileSync', before being parsed into a JSON object using 'JSON.parse'. Each function will return an error if the data is not imported correctly.

These datasets were merged by firstly iterating through each object in the flight data using '.map'. Then using '.find', the id of the airports in the airports data were matched to the id of airports in the flights data, linking the properties of the airports. A new object for the airline was also created with the properties code, name and country. A new object containing the merged data is returned with the source airport, destination airport, airline, aircraft and codeshare information. The function overall returns a new array of objects representing the merged data from both arrays. This merged data allows for easy analysis of the data in one array without having to reference the separate datasets.

Step 2: Mapping function for data
To allow modification across all elements in the combined dataset, a mapping function was created. The mapping function 'mapData' takes two parameters, an array and a function that modifies each item in the array. An empty array will store the modified items using '.push' and a timestamp using 'new Date().toISOString()' will be used to track when each item was last modified. The function iterates over each item in the list using a 'for of' loop and applies the function. If an item is modified, a timestamp will be added. The function then returns a new array containing all the modified items.

Step 3: Data analysis
Several functions were created to assist in analysing and filtering the data. The 'displayFlightInfo' formats the flight information into a string for simple display purposes. Five functions were made to filter information based on the source airport and destination airport, airline, codeshare status and aircraft type. Each function uses an 'if' statement which checks if the input matches any objects using '===' and returns them. The 'aircraftType' function alternatively checks if the input is included in any items using '.includes'. If there are no matches, each function will return 'undefined'. These functions can be used with the 'mapData' function to return the modified filtered arrays. These functions were attempted in a functional style with pure functions and immutability considered.

In order to complete further analysis, the airports were paired together so the time zone differences and number of flights between airports could be explored. Firstly the function 'flightFromAirports' was made to filter based on the source and destination airports that are provided as parameters. Next the main function 'airportPairs' iterates over the 'mergedData' to find all possible airport pairs using 'for' loops. It then ensures the airport and destination airports do not match using '!==' and creates a unique key for each pair 'using .sort().join('-')' to ensure that reverse pairs are considered as the same pair forwards. The time difference is calculated using 'Math.abs' to get the absolute value of the numbers. The flight

information and number of flights is calculated using the 'flightsFromAirports' function and '.length'. The function then returns the time difference and flight information about all unique pairs and filters out pairs with ≤1 flight using '.filter'. Although this function works, it could be improved by using separate functions for each task and implementing a way to handle errors. Currently there is an extended time complexity due to its nested loops and multiple responsibilities that violate the functional style.

The two functions 'flightStats' and 'timeStats' were used to calculate the statistics using the filtered airport pairs returned from the 'airportPairs' function. Each function initially sorts the arrays into descending order based on the property being investigated using '.sort((a, b) => b - a)'. The top ten highest numbers are extracted using '.slice(0,10)' whilst the maximum is the first element in the sorted array at index 0. The minimum is the last element in the array and the average is calculated by adding all the elements and dividing by the total number of elements. Each function then returns the minimum, maximum, average and top ten values for the number of flights between airport pairs and the time zone differences.

Step 4: Interactive web page
A basic HTML file with pre-existing elements was provided to work in conjunction with the analysis functions created. Dropdown elements and search term input allow for filtering and display of the data. Additionally buttons displaying summaries of the statistics calculated were added.

To load the data, two functions asynchronously fetch the flight and airport data from their respective files. They utilise a 'try' and 'catch' block to gracefully handle errors with importing data. The 'fetchFlightData' function also merges the two datasets, populates the dropdowns and calculates the statistics of the data.

The functions which populate the dropdowns are similar in nature. Firstly the specific dropdown element is selected by 'id' in the HTML document using 'document.getElementById'. All existing options are cleared and the input data array is sorted alphabetically. The default option 'any' is appended to the dropdown menu. It then iterates over the data and creates an option element for each item before setting the value attribute and text content and appending it to the dropdown menu. The 'searchBox' function also adds an event listener for when input is added.

The filtering functions 'filterFlights' and 'filterAirports' use the logic of the filtering functions created in step 3. They filter for direct matches using '===' and '.includes' based on the selected criteria. They also use the 'onlyTen' function to push only ten items to limit how much information is output.

The display functions work in conjunction with one another to format and display the filtered results on the webpage. The 'displayMatchingFlights' and 'displayMatchingAirports' functions show the results that match the selected criteria from the dropdowns and search box. The 'displayFilteredFlights' and 'displayFilteredAirports' then retrieve the correct HTML div element to display the information in. They clear any previous content and create a table element, '<table>', to populate with the data and append to the div. Table rows ('<tr>') and table data cells ('<td>') contain the properties of the flights and airports. If there is no matching data, a message acknowledging this is displayed neatly.

To create the buttons, basic code was added to the HTML script using '<button>'. The statistics functions from step 3 were used to output the information which is displayed. The 'busyAirports' and 'farAirports' functions format the information for display using paragraphs ('<p>') and lists ('<ol>', '<li>'). There was a third button planned to return aircraft statistics but the code was removed due to time constraints.

The final function is triggered when the webpage loads using 'window.onload'. It fetches the data asynchronously and has the event listeners for the dropdowns and buttons that trigger the display of information.

Step 5: Unit tests
Unit testing was completed with all functions to ensure they worked as expected. As much as possible, a variety of inputs was included to demonstrate how they are handled by the program. Jest matchers that were most commonly used were 'toBe', 'toEqual' and 'toContain'. Some external dependencies were mocked using 'jest.spyOn' to simulate scenarios such as invalid JSON data or a file not found. More important functions such as 'readAirportData' were tested to ensure it returned a valid array, handles empty input and handles errors gracefully.

To test the web related code, 'jsdom' was installed and used with jest to enable mocking the DOM environment. The window.onload async function was moved to a second script to allow the rest of the functions to be tested without triggering an error. The asynchronous functions were not tested due to errors in the testing suite. Results from the tests are provided in figure 1.
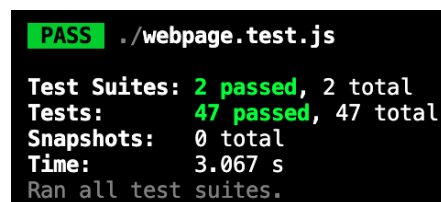
Step 6: Functional JavaScript
The initial set of functions for importing, mapping and analysis were created with the goal of functional JavaScript. However, as the need for more complex analysis grew, the functions deviated from these fundamental principles. This deviation is evident in the gradual decrease in readability and simplicity of the JavaScript code.

Conclusion
Overall this project demonstrates the practical application of Javascript and HTML in order to create a functional webpage. JavaScript can be used to import, merge and display multiple datasets which can then be analysed together. Although using JavaScript with the browser can present new challenges, browser debugging tools can assist in solving these problems. JavaScript can be a dynamic language when used with HTML to build interactive web pages.

Figure 1


```
 PASS   ./webpage.test.js

Test Suites: 2 passed, 2 total
Tests:       47 passed, 47 total
Snapshots:   0 total
Time:        3.067 s
Ran all test suites.
```

Note: Results of unit testing showing 47 out of 47 tests passing in the 2 testing suites utilised