FreeCodeCamp: JS Algorithms and Data Structure

On august 11, 2021, I attended a free training course rendering 300 hours at FreeCodeCamp about JavaScript algorithms and data structure. The course is composed of 9 courses: basic JavaScript, ES6, Regular Expressions, debugging, basic data structure, basic algorithm scripting, object-oriented programming, functional programming, intermediate algorithm scripting, and creating JS algorithm and data structure projects.

The first section of the course tackles the basics of JavaScript to learn the fundamental programming concepts of JavaScript. The course started with basic data structure like declaring variables, storing and assigning values, and concatenation. Following this, I learn how to work with arrays, objects, functions, loops, if/else statements, comparison using the operators. Some useful functions to manipulate arrays such as ***push()*** that is use to add element/s to the end of an array, ***pop()*** which removes the last element in an array, ***shift ()*** which is use to remove first element of an array, and its counterpart ***unshift()*** use to add element at the beginning of an array. And the very new topic I encountered in this course— using of recursion. Recursion is the pattern when a function calls itself, this is very helpful to use because it simplifies the task. Say example, instead of using iterative statement to create a countdown program, we can use recursion like this:

```
function countdown (n) {
   return n < 1 ? [] : [n].concat (countdown (n - 1));
}
```

The next course discussed ES6, the standardized version of JavaScript. The course taught the difference/scopes between/of let and var keywords, declaring read-only variable with ***const*** keyword, usage of arrow functions, setting and using parameters, using destructing assignment to assign variables and reassign array elements. In the ES6 course, Template literals and object literal declaration are also taught. Template literals can be used to define multiline strings, or can be used to define CSS strings. Template literals use back ticks instead of single or double quotes. I have also learned how to create a module script. Importing and exporting JavaScript application which allows creating reusable and separate components.

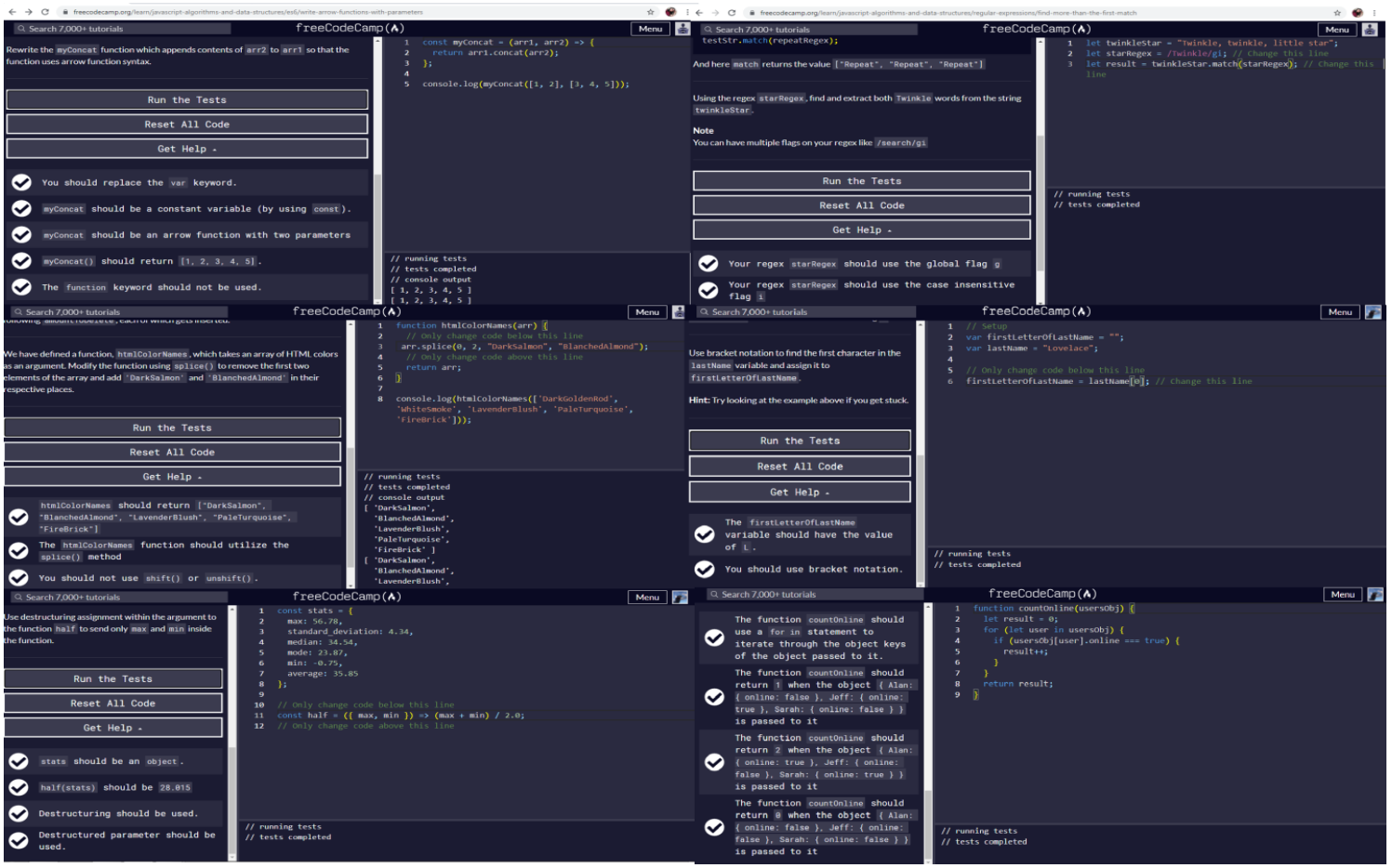
The next course is regular expressions or "regex" or "regexp". Regex are the patterns that help programmers match, search, and replace text. This is very powerful, but can be hard to read because regex use special characters to make more complex, flexible matches. To find the word in a string, we can use ***/some word here/*** and include ***.test()*** to test if the word can be find on that string. There are called flags which affect the search. These flags are /i wherein the search is case insensitive, meaning there's no difference between A and a. The /g indicates that the regular expression should be tested against all possible matches in a string. The /m is use for multi-line searching. The /d flag match a regular expression that contain the start and end indices of the substrings of each capture group. And the /u flag which enables various Unicode-related features.

The next section is debugging, the process of going through a code to find any issues, and fixing them. In this section it shows that issues in code come from three forms: first, syntax errors that prevent your program from running; second, runtime errors where your code has unexpected behavior; or third, logical errors where your code doesn't do what you intended. This section provides programs to debug.

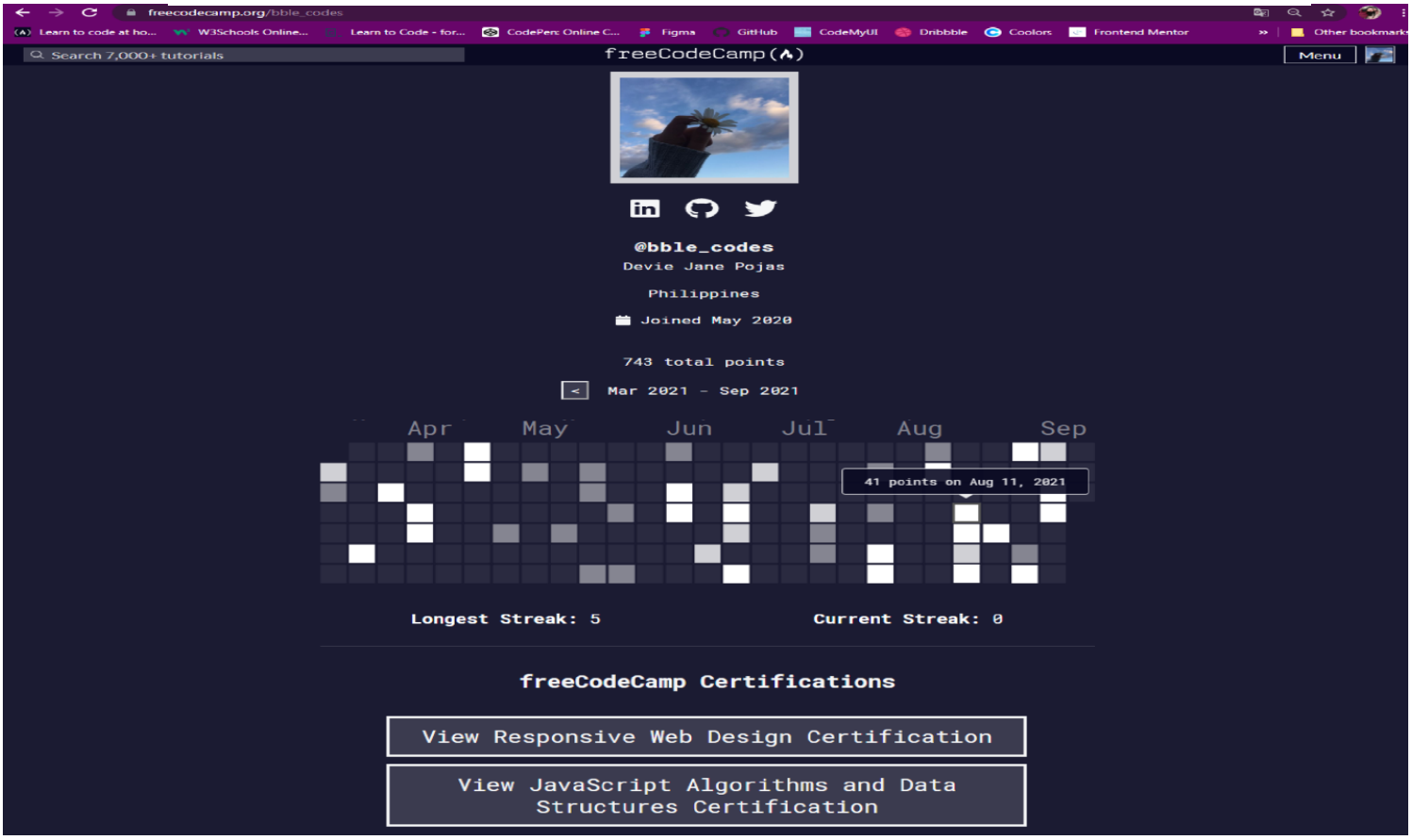
Following this are the basic data structure and basic algorithm scripting, where it tackles the differences between arrays and objects, and which to use in different situations, some helpful JS methods like ***splice()*** and ***Object.keys()*** to access and manipulate data, fundamentals of algorithmic thinking by writing algorithms that do everything from converting temperatures to handling complex 2D arrays. Next lesson tackled is the OOP or object oriented programming, one of the major approaches to the software development process. In OOP, objects and classes organize code to describe things and what they can do. And the last lesson of the course tackle the functional programming, a popular approach to software development where code is organized into smaller, basic functions that can be combined to build complex programs. This section taught the core concept of functional programming like pure functions, avoiding mutation, and writing clean code methods like .map() and .filter().

The last two sections of this course are JavaScript algorithm scripting and JavaScript algorithms and data structures projects. The JavaScript algorithm scripting is composed of challenges which I have to use the OOP and functional programming. The last sections are composed of five (5) projects needed to finish, the palindrome checker, roman numeral converter, Caesars cipher, Telephone number validator, and cash register in order to have the certification.

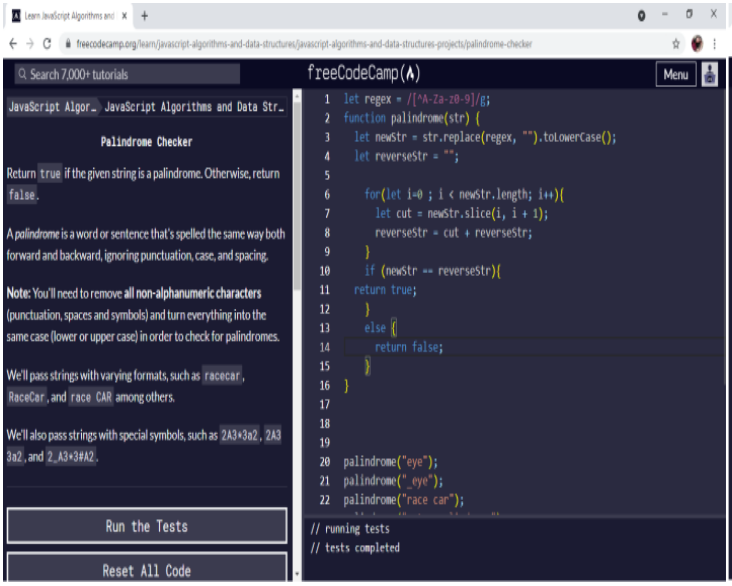
FreeCodeCamp: JS Algorithms and Data Structure Courses



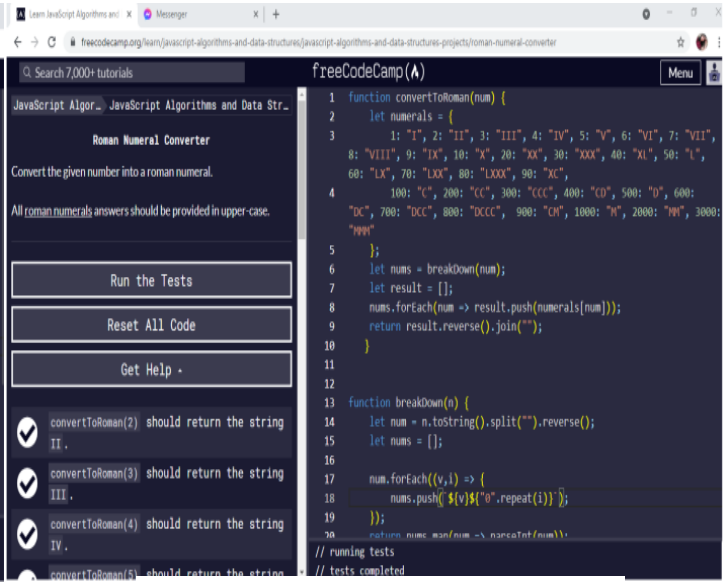
Some daily activities on each FreeCodeCamp: JS Algorithms and Data Structure Courses



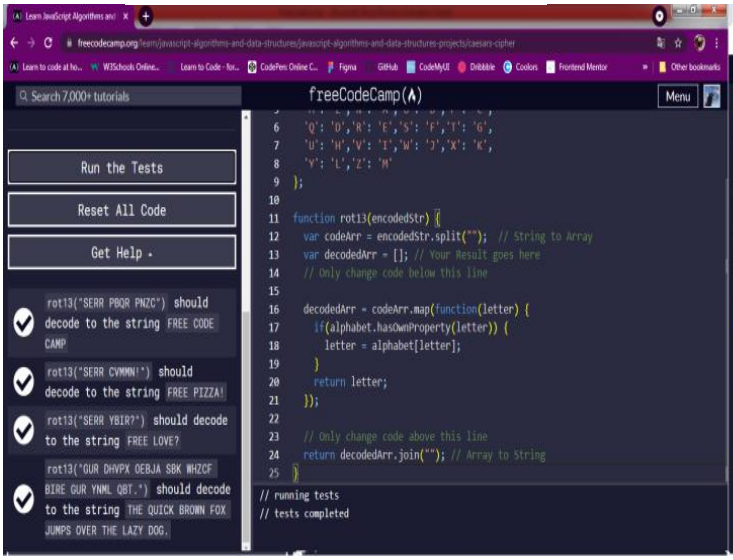
FreeCodeCamp Profile and History Tracker (August 11- Sept 1)



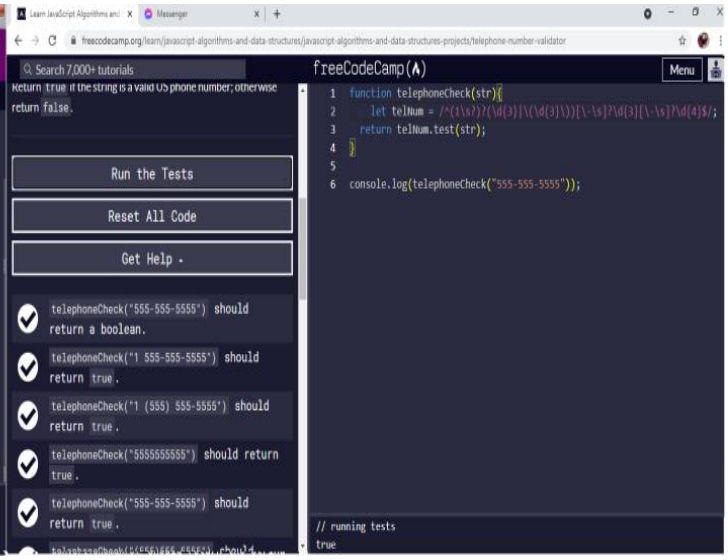
Project 1: Palindrome Checker



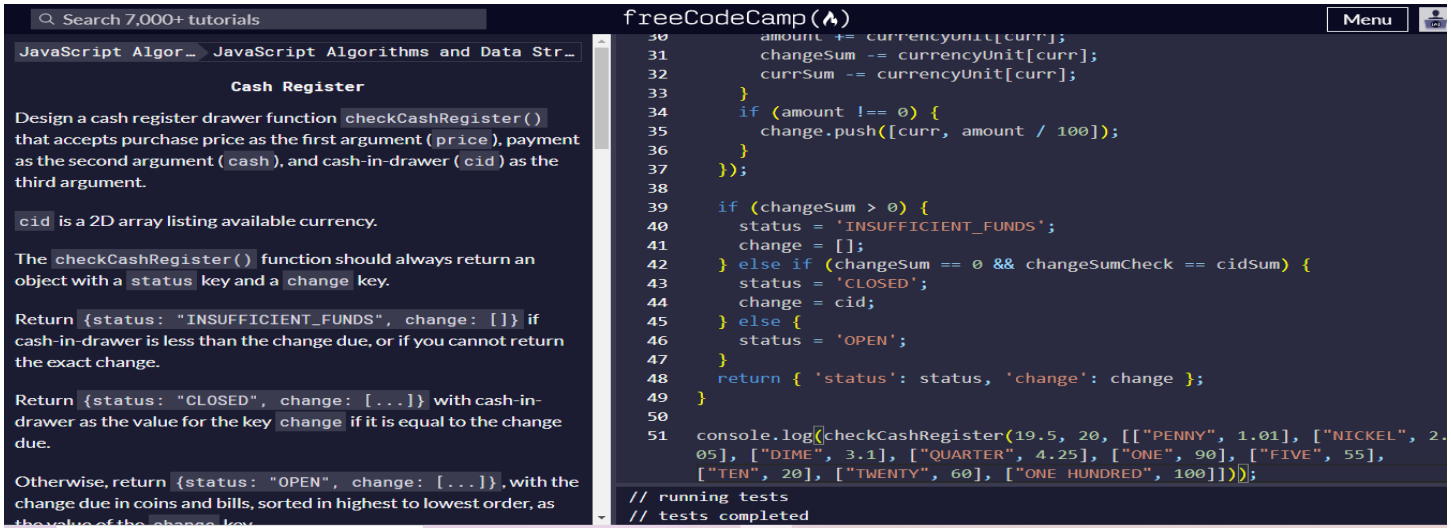
Project 2: Roman Numeral Converter



Project 3: Caesar's Cipher



Project 4: Telephone Number Validator



Project 5: Cash Register