**Assignment 4**

**Practice with Functions**

1. Write a function called halfNumber() that accepts one argument (a number), divide it by 2, and return the result. It should log a string like "Half of 5 is 2.5.".
2. Write a function called squareNumber() that accepts one argument (a number), square that number, and return the result. It should also log a string like "The result of squaring the number 3 is 9."
3. Write a function called percentOf() that accepts two numbers, figure out what percent the first number represents of the second number, and return the result. It should log a string like "2 is 50% of 4."
4. Write a function called findModulus() that accepts two numbers. Within the function write a statement that returns the modulus of the first and second parameters. It should log a string like "2 is the modulus of 4 and 10."
5. Create a JavaScript function that accepts a certain amount of numbers as parameters. Those numbers should be collected using a prompt and the numbers should be delimited by commas. Once the values are collected, find the sum of all of the numbers combined. You will need to use a function, loop, arguments object, and several type conversion global functions to accomplish this task.

**The Rock, Paper, Scissors Game**

Rock, paper, scissors is a classic 2 player game. Each player chooses rock, paper or scissors. The possible outcomes include:

* Rock destroys scissors.
* Scissors cut paper.
* Paper covers rock.

Our code will break the game into 3 phases:

1. User makes a choice. How will we collect the user’s choice?
2. Computer makes a choice. How will we collect the computer’s choice?
3. A conditional that determines who wins.

You will be responsible for figuring out some of the logic. These next steps will attempt to guide you down the right path:

1. Begin by prompting the user for their choice.
2. Create the computer’s choice. This will be generated similarly to the coin flip that you did in the last assignment. Remember though, rather than 2 options, there will be 3 here.
3. Depending on what the numeric value of the computer’s choice is, reset it to a string value of rock, paper, or scissors instead.
4. Create a conditional statement that checks the user’s choice in relation to the computer’s choice. Once a winner is defined, display a message within an alert box indicating who the winner is.
5. What if the result ends in a tie? Figure out how to handle that as well.
6. What if the user enters something other than rock, paper, or scissors into the prompt? Figure out how to handle that as well.
7. **const** getComputerChoice= () => {
8. **const** randomNumber = (Math.floor(Math.random() \* 3))
9. **if** (randomNumber === 0){
10. **return** 'Rock';
11. }
12. **else** **if** (randomNumber === 1){
13. **return** 'Paper';
14. }
15. **else** **if** (randomNumber === 2){
16. **return** 'Scissor';
17. }
18. }
19. console.log(getComputerChoice());

**The Basic Calculator**

In this part of the assignment you will build a simple calculator using the skills you’ve acquired in regards to functions and modules. You will ask the user for a number, a second number, and then finally ask them what kind of operation they want to perform, either add, subtract, multiply, or divide. Depending on what they select, you will perform that operation. To complete this part of the assignment follow the steps outlined here:

1. Create a module that abstracts away the main calculation logic of your application. This module will have a private function called calculate and expose four different functions (add, subtract, multiply, and divide). The shell of this module with some basic instruction has already been outlined for you.
2. In the main script, prompt the user for a number and store that in a variable. Convert that to a number.
3. Then, prompt the user for a second number and store that in a variable. Convert that to a number.
4. Finally, prompt the user for the operation they would like to perform (add, subtract, multiply, divide) and store that in a variable.
5. Use a switch statement to evaluate the operation parameter being passed in. Depending on what it is, perform that operation by calling the appropriate function from the module.
6. Display the result of the calculation within an alert.
7. Figure out how to make it so that if the user doesn’t enter either add, subtract, multiply, or divide, they are displayed an alert message telling them that. Then you’ll have to rerun the application so that the application asks the user for the numbers and operation once more. You’ll have to add this functionality within a do while loop similarly how the labs were done in the previous lecture.

In this module, I learned first-hand how to use functions in JavaScript, which are named sections of a program that performs a specific task. Functions can be referenced by name and executed as many times as necessary. There are several useful built-in functions, but we can write our own functions.

In addition, basic functions can be categorized as function declarations, function expressions, and function constructor. For Named Function Declarations, the functions are hoisted, and JS engines reads all declarations first before executing remaining code. For me, initially the concept of “calling the function”, used to run the named function, was not intuitive. Another challenging concept is “passing values” by value and by reference into functions. With more practice, these concepts will make more sense: primitive types, which variables are mutable, can be passed to a function by value; and since arrays are objects, objects can be passed into a function by reference. Returning values is another concept which means to return a value back to the calling application, by using return keyword with the variable being called.

Function Expressions covered are the variable assignment functions and arrow functions. The variable assignment functions are also referred to as anonymous functions because they don’t have a name. These functions must be coded before any statement that call it as opposed to named function declarations which are hoisted. Furthermore, since functions are immutable, the const keyword is used instead of the let keyword. The arrow function utilize => and makes the code more concise and simplify function scoping. At first, it was confusing, but looking over the examples and videos over again, it makes sense.

From this:

Let sum = function (num1, num2) {

Return num1 + num2;

};

To this:

const sum = (num1, num2) => num1 + num2;

Lastly, modules was introduced and it refers to blocks of independent, reusable code. From the lab, we covered public vs private variables and when modules are exposed using the import statement. There are lots of concepts in this module, I hope to have more practice to fully have it sink in.