Functional Programming

(Functions, Function Expressions, Arrow Functions)

Part 1

Challenge 1

Step 1

First, create a variable called `greetingPt1` and set it equal to the string "Hello".

Next, create a variable called `greetingPt2` and set it equal to the string 'World!'.

Lastly, concatenate those stings together in a way that it will print the string "Hello World!" to the console.

Step 2

Now, if we wanted to print that message from Step 1 to the console again we would have to write the logic a second time to concatenate those variables together (with a space in the middle). That seems inefficient 😕.

Rather than writing that logic all over again, create a function called `greeting` that will do it for us. That way anytime we want to print that message to the console we can simply execute the `greeting` function.

The `greeting` function should create our `greetingPt1` and `greetingPt2` variables, and log it to the console so that it prints "Hello World!".

Lastly, execute the `greeting` function several times to check your work.

Challenge 2

Step 1

Create a function called `createPartnerGreeting` .

In the function body of `createPartnerGreeting` you should create a `partnerName` variable and set it equal to your pair programming partner's name.

`createPartnerGreeting` should then return a string that says "Hey, (partners name). It's great to work with you on these challenges!".

Then, create a variable called `partnerGreeting` and set it equal to the returned value of executing the `createPartnerGreeting` function.

Once you are finished, log the value of `partnerGreeting` to the console to check your work.

Step 2

The `createPartnerGreeting` function you created is useful. However, it is only a useful function for when you are working with that partner. I.e. if you were working with someone else you would have to go in and change that function definition to use the new partner's name.

See if you can refactor your `createPartnerGreeting` function so that when you run it you can pass in a name as an argument. This way your function will be more dynamic and can be used for anyone that you are pair programing with 😃.

Be sure to test your refactored `createPartnerGreeting` with a few different names.

Challenge 3

Define a function called `addTwo` that returns the sum of any two numbers.

Challenge 4

Define a function called `arraySum` that takes an array of numbers as an argument and returns the sum of all the numbers in the array.

Challenge 5

Create a function called `arraySumEven` that takes an array of numbers as an argument and returns the sum of all the EVEN numbers in the array.

Challenge 6

Create a function called `stringCreator` that takes in an array and a string to be removed from the array as its arguments.

`stringCreator` should remove all of the strings that match our string argument from our array argument and return out a single string of the remaining array elements.

Challenge 7

Create a function expression called `lengthChecker` that takes in a string and a number as its inputs.

`lengthChecker` should return the boolean value of true if the input string's length is greater than or equal to the input number. If it is less than the input number `lengthChecker` should return the boolean value of false.

Hint: If you are need to refresh on function expressions check out the MDN Docs on Function Expressions

Challenge 8

Create a function expression called `valueChecker` that takes in an object and string as its inputs.

`valueChecker` should check to see if the string argument exists as a key of one of the properties on the object argument.

If the key does exist `valueChecker` should return the value stored on that key. Otherwise it should return the string 'Sorry, "(key name)" does not exist on the object'.

Challenge 9

Step 1

First, start by declaring a normal function called `findWaldo` that accepts an array as an input.

`findWaldo` will search through an input array containing strings. If the string "Waldo" is found in the input array `findWaldo` will return the string "I found Waldo (number of times) time(s)!". If the string "Waldo" does not exist in the input array `findWaldo` should return the string "Where's Waldo?!".

Test that you function is working properly by uncommenting the code.

Step 2

Next, refactor your `findWaldo` function into an Arrow Function expression.

Hint: If you get stuck check out the MDN documentation for Arrow Function Expressions 🙂.

Test your code to make sure it's still working properly.

***