Name _____ Period _____ Role (Circle one) Programmer/Driver

Name _____ Period _____ Role (Circle one) Programmer/Driver

Creating Functions

Your Tasks (Mark these off as you go)

- ☐ Create a functions project folder
- ☐ Declare a function
- ☐ Call a function
- ☐ Have Ms. Pluska check off Declare a function and Call a function
- ☐ Pass a parameter to a function
- ☐ Use a *return* statement in a function
- ☐ Have Ms. Pluska check off functions with parameters and return statements
- ☐ Complete challenges 1 thru 2
- ☐ Receive credit for the group portion of this lab
- ☐ Receive credit for the individual portion of this lab

☐ Create a functions project folder

This lab will follow the same workflow as the last lab. You will begin by making a new project directory within which you will create an index.html file and an app.js file. To view the results of your JavaScipt code, you will be using console. If you forgot how to do this, refer to the first lab "Introduction to JavaScript".

- ➔ First create a new folder on your computer called Functions.
- ➔ Add two new files to this folder,
  - o Index.html
  - o App.js

In your Index.html file, add the following code

Index.html

```
<html>
      <head>
           <script src = "App.js"></script>
     </head>
</html>
```

☐ Declare a function

In JavaScript, there are many ways to create a function. One way to create a function is by using a function declaration. Just like how a variable declaration binds a value to a variable name, a function declaration binds a function to a name, or an identifier. Take a look at the anatomy of a function declaration below:

```
function greetWorld(){
     console.log(“Hello World!”);
}
```

A function declaration consists of:

- The function keyword.
- The name of the function, or its identifier, followed by parentheses.
- A function body, or the block of statements required to perform a specific task, enclosed in the function's curly brackets, { }.

➔ Using a function declaration, create a function called getReminder() that prints a reminder to the console.
  - In the function body of getReminder(), log the following reminder to the console: 'Water the plants.'

➔ Using a function declaration, create a function called greetInSpanish().
  - In the function body add a console.log() that prints: 'Buenas Tardes.'

□ Call a function

As we saw above, a function declaration binds a function to an identifier.

However, a function declaration does not ask the code inside the function body to run, it just declares the existence of the function. The code inside a function body runs, or executes, only when the function is called. To call a function in your code, you type the function name followed by parentheses.

```
GreetWorld();
```

The function call above executes the function body, or all of the statements between the curly braces in the function declaration below.

```
function greetWorld(){
     console.log("Hello World!");
}
```

We can call the same function as many times as needed.

Let's practice calling functions in our code.

➔ Imagine that you manage an online store. When a customer places an order, you send them a thank you note. Let's create a function to complete this task:
  - Define a function called sayThanks() as a function declaration.
  - In the function body of sayThanks(), add code such that the function writes the following thank you message to the console when called: 'Thank you for your purchase! We appreciate your business.'

➔ Call sayThanks() to view the thank you message in the console.
➔ Functions can be called as many times as you need them. Imagine that three customers placed an order and you wanted to send each of them a thank you message. Update your code to call sayThanks() three times.

□ Have Ms. Pluska check off Declare a function and Call a function

Before you continue have Ms. Pluska check off Declare a function and Call a function

Do not continue until you have Ms. Pluska's (or her designated TA's) signature _____

□ Pass a parameter to a function

So far, the functions we've created execute a task without an input. However, some functions can take inputs and use the inputs to perform a task. When declaring a function, we can specify its parameters. Parameters allow functions to accept input(s) and perform a task using the input(s). We use parameters as placeholders for information that will be passed to the function when it is called.

```
function calculateArea(width, height){
     console.log(width*height);
}
```

In the example above, calculateArea(), computes the area of a rectangle, based on two inputs, width and height. The parameters are specified between the parenthesis as width and height, and inside the function body, they act just like regular variables. width and height act as placeholders for values that will be multiplied together.

When calling a function that has parameters, we specify the values in the parentheses that follow the function name. The values that are passed to the function when it is called are called arguments. Arguments can be passed to the function as values or variables.

```
calculateArea(10, 6);
```

In the function call above, the number 10 is passed as the width and 6 is passed as height. Notice that the order in which arguments are passed and assigned follows the order that the parameters are declared.

```
var rectWidth = 10;
var rectHeight = 6;

calculateArea(10, 6);
```

The variables rectWidth and rectHeight are initialized with the values for the height and width of a rectangle before being used in the function call.

By using parameters, calculateArea() can be reused to compute the area of any rectangle! Functions are a powerful tool in computer programming so let's practice creating and calling functions with parameters.

- ➔ Add a parameter called name to the function declaration for sayThanks().
- ➔ With name as a parameter, it can be used as a variable in the function body of sayThanks().
  - Using name and string concatenation, change the thank you message into the following:

```
'Thank you for your purchase '+ name + '! We appreciate your business.'
```

- ➔ A customer named Cole just purchased something from your online store. Call sayThanks() and pass 'Cole' as an argument to send Cole a personalized thank you message.

□ Use a *return* statement in a function

When a function is called, the computer will run through the function's code and evaluate the result of calling the function. By default that resulting value is undefined.

```
function rectangleArea(width, height){
     var area = width * height;
}

console.log(rectangleArea(5, 7); //prints undefined
```

In the code example, we defined our function to calculate the area of a width and height parameter. Then rectangleArea() is invoked with the arguments 5 and 7. But when we went to print the results we got undefined. Did we write our function wrong? No! In fact, the function worked fine, and the computer did calculate the area as 35, but we didn't capture it. So how can we do that? With the keyword return!

```
function rectangleArea(width, height){
     var area = width * height;
     return area;
}

console.log(rectangleArea(5, 7); //prints 35
```

To pass back information from the function call, we use a return statement. To create a return statement, we use the return keyword followed by the value that we wish to return. Like we saw above, if the value is omitted, undefined is returned instead.

When a return statement is used in a function body, the execution of the function is stopped and the code that follows it will not be executed.

The return keyword is powerful because it allows functions to produce an output. We can then save the output to a variable for later use. Consider the example below,

```
function rectangleArea(width, height){
     var area = width * height;
     return area;
}

var myArea = rectanglearea(10, 5);

console.log(myArea);//prints 50
```

Imagine if we needed to order monitors for everyone in an office and this office is conveniently arranged in a grid shape. We could use a function to help us calculate the number of monitors needed!

- ➔ Declare a function monitorCount() that has two parameters. The first parameter is rows and the second parameter is columns.
- ➔ Let's compute the number of monitors by multiplying rows and columns and then returning the value. In the function body of the function you just wrote, use the return keyword to return rows * columns.
- ➔ Now that the function is defined, we can compute the number of monitors needed. Let's say that the office has 5 rows and 4 columns. Declare a variable named numOfMonitors and assign numOfMonitors the value of invoking monitorCount() with the arguments 5 and 4.
- ➔ To check that the function worked properly, log numOfMonitors to the console.

□ Have Ms. Pluska check off functions with parameters and return statements

Before you continue have Ms. Pluska check off functions with parameters and return statements

Do not continue until you have Ms. Pluska's (or her designated TA's) signature _____

□ Complete Challenges 1 thru 2

Challenge 1

Write a program that prompts the user for a shopping list of items.

Begin your program by declaring a variable called list.

Write a function called addItem that accepts a parameter. The parameter is the item the user provides. Each time addItem is called, the item should be added to the list. You should separate the items on the list using some kind of delimter (spaces, commas, new line ("\n"), etc)

Next, alert the user to create a list,

```
alert("Let's make a shopping list");
```

Your program should prompt the user for at least five items. For each item, call the function addItem to add the item to the list.

Write a function called displayList. This function alerts the user with their shopping list.

```
alert("Here is your list " + "\n" + list);
```

Run your program and create a shopping list!

Challenge 2

Write a temperature converter program. Your program will prompt the user for a temperature in Fahrenheit and will alert the user of the temperature in Kelvin and Celsius.

Begin your program by prompting the user for the temperature in Fehrenheit.

Write a function called fehrenheitToCelsius that accepts a parameter. The parameter is the Fehrenheit temperature provided by the user. Your function should convert the temperature to Celsiu and return the value.

(32°F− 32) × 5/9 = 0°C

Write a function called celsiusToKelvin that accepts a parameter. The parameter is the Celsius temperature converted using the previous function. Your function should convert the temperature to Kelvin and return the value.

Kelvin = °C + 273

Write a function called displayTemps. This function should alert the user of the original temperature in Fehrenheit and the corresponding converted tempetures.

Run your program and check out today's temperature!

□ Receive Credit for the group portion of this lab

Make sure to indicate the names of all group members, then submit this lab to the needs to be graded folder to receive credit for the group portion of this lab.

□ Receive Credit for the individual portion of this lab

Implement challenges 1 thru 2 on your computer. Show Ms. Pluska the completed challenges to receive credit for the individual portion of this lab.