**Functions - Step1**

**Functions**

A function is a block of organized, reusable code that is used to perform a single action. Functions repeat tasks that involve a similar sequence of steps. You have already seen the console.log() function, which allows you to predictably log some output to the console.

Look at this example of a function definition that returns the product of two numbers:

function multiply(n1, n2) {  
  return n1 \* n2;  
}  
   
//To make this function run, you need to call it like this:  
multiply(3, 15);

This is called a function declaration. You start by creating a function name, which you then define as a function by using the function(){} syntax. Functions can have optional inputs like 3 and 5 in the example above. These are called parameters.

Within the parentheses are a set of arguments, which are values that are used or manipulated by the function in some way. If you have multiple arguments, they are each separated by a comma. A function can also have no arguments.

The code to be executed by the function goes within the curly braces {}. If you expect the function to give back some value, it should include a return statement, which is done by using the keyword return, followed by the value you want to be returned. If you don’t expect your JavaScript function to return a value, you do not have to include a return statement. In this case the function returns undefined.

In order to make use of a function, you must invoke or call it.

In the example above, the function is called like this:

multiply(3, 15);

You can also write a function expression where you assign the returned value of the function to a variable. That variable can even be passed into another function as an argument, so you can reuse that logic. This is called a callback function, and you will learn more about that later.

var timesTen = function (x) {  
  return x \* 10;  
};  
   
console.log(timesTen(5));

**Why are Functions Important?**

The most important reason to use functions is that they allow you to reuse code and create modules to perform procedures you plan on using repeatedly. A good practice is to make the function do one task and one task only.

**Task Instruction**

You task in this activity is to complete the implementation of the function add. This function needs to do the following:

* It takes two parameters x and y
* It returns the sum of the two parameters x and y.

*Note: You can calculate the sum of two numbers simply adding them: x + y*

Task

Write a function that adds 2 numbers.

**Functions - Step 2**

#### Functions Practice

Now try writing a function that welcomes a friend.

To complete this task, you will need to implement the greet() function. This function needs to have a parameter called name and return a message that welcomes the person with that name.

For example, if you call the function as follow: greet("Dani"), the result should be: "Hello Dani" or "Greetings Dani'

Your code should be added to the greet.js file inside the functions-02 folder.

Task

Create a function call greet() with one parameter name which returns a string that welcomes the name passed in.

**Functions - Step 3**

**Functions**

**typeof Method**

typeof returns a string indicating the data type of the parameter being evaluated, as you can see in the examples below:

let number\_of\_transactions = 970  
console.log(typeof(number\_of\_transactions))  
//expected result: number  
   
console.log(typeof("The ceremony is scheduled at 8am."))  
//string  
   
console.log(typeof(true))  
//boolean  
   
console.log(typeof(sept\_record))  
//undefined

**Strict Equality Comparison - The Concept of Triple Equals (===)**

Strict equality compares two values for both content and type. For example, consider the following variables:

var numberOne= 1;  
var stringOne= "1";

Both variables here have the same value: "1", so they are equal. However, the first variable's type is number, and the second variable's type is string.

Using strict equality means that JavaScript will not only check the value of the variables for equality but also the type of variable too. In this instance, numberOne and stringOne would not be equal even though they hold the same value.

Here are more examples showcasing strict equality using the triple equals(===) operator:

let score = 10  
let score1 = "10"  
let age = 10  
let my\_street = "123 Van Nuys St"  
let isOpen = true  
let camys\_st = "123 Mission St"  
   
console.log(score === score) //true  
console.log(score === score1) //false, this is comparing a number and string  
console.log(score === age) //true, the values are the same as well as the types, which are both numbers  
console.log(my\_street === my\_street) //true  
console.log(my\_street === camys\_st) //false, the two strings have different values  
console.log(score === isOpen) //false, this is comparing a string to a boolean

**Task Instructions**

Your task in this activity is to create a function called isString that takes three arguments (a, b, c). This function does the following:

* It uses the typeof operator and strict equality comparison to check if the type of all three parameters a, b and c is string.
* If each argument is a string it returns the message strings.
* If any of the three parameters is not a string, then it returns the message not strings.

*Hint: Read about the typeof built-in function to help you solve the exercise.*

**Task**

Given the isString function inside functions-03, make use of the typeof method to check if all passed in arguments are strings. Otherwise, return a message. For more details check out the main.js file inside functions-03.