INTRO TO FUNCTIONS IN JAVASCRIPT

A **function** is a block of code that can be run on command.

To decide to make a function instead of just using a variable or other statement, two things have to be true.

- The code is **reusable**, and can be called on command.
- It does just **one thing well** over and over.

A function essentially is called, does something, then it stops. It works a bit like calling a channel with a remote control.

Functions can be **language-defined** or **user-defined**. What most people think of as functions are user-defined functions.

Since these are more common, we will first approach **user-defined versions** based on their parts. Keep in mind that function types are decided up on by their purpose, and their purpose or goal will determine what parts are used.

There are two external parts to a function. They include a **function definition** and a **function call**. A function call might also be known as a function invocation, using it as ‘invoking a function’.

- A function definition creates a function.
- A function call runs it, sets it in motion.

We will begin with the example of a basic function that outputs a string, and demonstrate a single call to it. This is the simplest function possible.

```
function sayHello() {
    console.log(“HELLO WORLD”);
    }

sayHello();
```

Now, let’s discuss more formally **the internal parts or components of a function itself.** Let’s break it down, like the parts of an engine.

A function traditionally has 5 parts.

- the function keyword
- the function name
- the parameters of a function
- the function body
- the return statement

The **function keyword** is necessary to get the system to recognize something as a function, and to make it work as such.

The **function name** is a unique identifier that gives that function its own identity. It has to be named to separate it from other possible functions.

The **function body** describes what the function is supposed to do, and how it does it.

form or syntax:

```
function functionName( ) {

    //ACTING FUNCTION CODE;

}
```

example:

```
function sayHello() {
    console.log(“HELLO WORLD”);
   }
```

If a function has **parameters**, then value ‘slots’ are reserved for variable values. Parameters are placeholders.

Here is an example with a single **parameter** that is reused with different text strings. The values of “Jack” and “Jill” are the arguments for the parameter.

Using a single parameter:

// FUNCTION ‘UP THE HILL’

```
function heyThere(name) {

    console.log("Hey there " + name);

}


heyThere("Jack");

heyThere("Jill");
```

using multiple parameters:

```
function heyThere2(name1, name2) {

    console.log(name1 + " and " + name2 + " went up the hill, each with a buck and a quarter. " );

}


heyThere2("Jack", "Jill");
```

The **return statement** ends the function's execution and sends a value (or result) back to the part of the code that called it, allowing that value to be stored in a variable.

Example of returning a value:

```
function subtract(num1, num2) {

    return num1 - num2;

}

console.log("AFTER THE RETURN: ");

const result = subtract(10, 2);

console.log(result);
```

Start thinking of functions in terms of:

- parameters
- arguments
- return values

These parts perform a process. Something goes in, some work is done to it, and that same something afterward comes back out.

Running the Code:

Notice that this part of the code is a **function call**:

```
sayHello();
```

It causes the function to initiate or run **one time**.
Running it again would mean using additional calls.

```
sayHello();
sayHello();
sayHello();
```

(runs three times)
I know this is repeating, but it is important to remember:

In the case of a function being coded, there is:
- the function definition
- the function call

Both are required to make the function **work**.

This is the case of the simplest function possible.
There are no return values to be concerned with.
There are no arguments.
There are no parameters.

In short:

Parameters are the named variables listed in the function definition (e.g., in function greet(name), name is a parameter).

Arguments are the actual values passed to the function when it is called or invoked (e.g., in greet("Alice"), "Alice" is the argument). Arguments fill the role of parameters during function execution.

Return values are the values a function produces and sends back to the part of the code that called it. Functions use the return keyword to specify what value to output, and if no return statement is present, the function implicitly returns undefined.

Using return values: Here is a simple example of a JavaScript function that calculates the square of a number and uses a return value:

```
// Function definition
function square(number) {
  return number * number;
}

// Function call and return value assignment
let result = square(5);

// The value of 'result' is now 25
console.log(result);
```

In this example:

1. The square() function is called with the argument 5.
2. The expression number * number (which evaluates to 25) is specified after the return keyword.
3. The function stops executing and passes the value 25 back to where it was called.
4. The variable result is assigned the returned value, 25 [1].