#### What is a Function?

1. A function is a **subprogram** designed to perform a particular task.
2. Functions are executed when they are called. This is known as **invoking** a function.

The code inside the function will execute when "something" **invokes** (calls) the function:

* When an event occurs (when a user clicks a button)
* When it is invoked (called) from JavaScript code
* Automatically (self invoked)

1. Values can be **passed** into functions and used within the function.
2. Functions **always** return a value. In JavaScript, if no return value is specified, the function will return undefined.
3. Functions are **objects**.

**Define a Function.**

There are a few different ways to define a function in JavaScript:

A **Function Declaration**defines a named function. To create a function declaration you use the function keyword followed by the name of the function. When using function declarations, the function definition is hoisted, thus allowing the function to be used before it is defined.

function name(parameters){  
 *statements*}

A **Function Expressions** defines a named or anonymous function. An anonymous function is a function that has no name. Function Expressions are not hoisted, and therefore cannot be used before they are defined. In the example below, we are setting the anonymous function object equal to a variable.

let name = function(parameters){  
 *statements*  
}

An **Arrow Function Expression** is a shorter syntax for writing function expressions. Arrow functions do not create their own this value.

let name = (parameters) => {  
 *statements*  
}

**Function Declaration**

A JavaScript function is defined with the **function** keyword, followed by a **name**, followed by parentheses **()**.

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:  
**(*parameter1, parameter2, ...*)**

The code to be executed, by the function, is placed inside curly brackets: **{}**

function *name*(*parameter1, parameter2, parameter3*) {  
    *code to be executed*  
}

Function **parameters** are listed inside the parentheses () in the function definition.

Function **arguments** are the **values** received by the function when it is invoked.

Inside the function, the arguments (the parameters) behave as local variables.

When using function declarations, the function definition is hoisted, thus allowing the function to be used before it is defined

Arrow functions (also called “fat arrow functions”) are undoubtedly one of the more popular features of ES6. They introduced a new way of writing concise functions.

Here is a function written in ES5 syntax:

function timesTwo(params) {  
 return params \* 2  
}

timesTwo(4); // 8

Now, here is the same function expressed as an arrow function:

var timesTwo = params => params \* 2

timesTwo(4); // 8

An **arrow function expression** has a shorter syntax than a [function expression](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/function) and does not have its own [this](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/this), [arguments](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/arguments), [super](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/super), or [new.target](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/new.target). These function expressions are best suited for non-method functions, and they cannot be used as constructors.