Functions

JavaScript scripts can easily end up being hundreds if not thousands of lines of code. Some of those codes are repeating codes that can be simplified to minimize the amount of lines of code in our program. To help structure our code and make common operations reusable, we create functions and objects.

Functions are mini programs inside our program scripts. They can be used to segment sections of our code to make it easier to manage, or to perform repeated operations, or both. Functions are wrapped around code blocks, which contain the actual declarations that will be executed, and usually include some combination of variable assignments, operations, and conditions.

Functions do one of two things: they create a result immediately, such as changing the content of an element on a web page, or they provide a response or output to be used by other functions, called a return value. In JavaScript, we work with three types of functions: named functions, which are executed when called by name, anonymous functions, which are normally executed once they are triggered by a specific event, and immediately invoked function expressions, which they run the moment the browser finds them.

All functions have the same general structure. They start with the word **function**, which tells the browser, here I am declaring a function, followed by its name, two parentheses, and then a pair of curly braces that wrap the block of code.

```
Definition        function name
    ↓              ↙
function sum(x, y) {
   let add = x+y;
   return add;
}
```

To run a named function, we call it by name at a location in the script where we want it to run. That means we can define a function at the top of the script and choose to run it at the bottom of the script. In fact, this is the coding standard for functions.

```
// call for function sum. var z will store the return value add
let z = sum(3,-1);
```

Technically, it does not matter where a function is in the JavaScript script because the browser will load all the JavaScript's code first and then execute it. But to make it easier for humans to read, we normally place a function before it is called in the script. This provides an appearance of logical structure in our code when you read it from top to bottom.

Finally, functions can return values from where they were called using the return keyword. Everything that is returned is not executed directly, but is captured in a variable or used immediately in another function.

Creating a function: user defined functions

Named functions

Named functions are useful if we need to call a function many times to pass different values to it or just need to run it several times and it's also useful when functions just get really big and just clutter up the overall flow of the script. In that case, we create functions and put them above the main script to be called when needed. Named functions can have parameters and not, or with or without returning value.

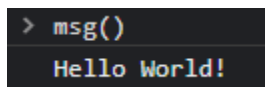Functions are reusable. We can call a function as many times as necessary in our JS program.

Parameters in JS are like a placeholder where you store the value of a variable, known as arguments when the function is called, and pass that value to the function to be used within the function. Basically, parameters are variables that pass values to the function as arguments to obtain the function's result.

Function without parameters or return value

Example 1) define a function that will print a message in the console

```
function msg() {

     console.log('Hello World!')

   }
```
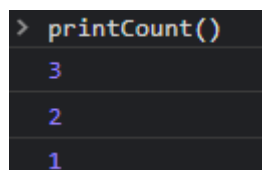
Calling the function in the console ➔

```
> msg()
  Hello World!
```

Example 2) define a function called **printCount** that prints out the number from 3 to 1, inclusive, every time the function is called.

```
function printCount(){
  for(let num =3; num>0; num -= 1){
    console.log(num)
   }
}
```

Calling the function from the console

```
> printCount()
  3
  2
  1
```

Function with parameters but no return value

Example 3) define a function that takes a parameter as *name* and print the argument value in the console

```
function greeting(name){
    console.log(`Hello ${name}`);
}
```

```
> greeting("Ana")
  Hello Ana
< undefined
> greeting("Peter")
  Hello Peter
< undefined
```

If a define function have parameter/s but when the function is called and no arguments is applied, the function will take the value as **undefined**

```
> greeting()
  Hello undefined
```

Example 4) define a function that will take a **message** as an argument and convert its value in uppercase.

```
function upperMessage(message){
  let upper = message.toUpperCase();
  console.log(upper);
}

//call function upperMessage(message)
upperMessage('Passed Final Exam!')
```

Result ➔ `PASSED FINAL EXAM!`

Function with multiple parameters

Order of arguments matters! The order of arguments align to each parameter in the function.

Example 5) define a function that will take two parameters, *firstName* and *lastName*, and print the first name with the initial letter of the last name.

```
 function greet(firstName, lastName){
    console.log(`Hello ${firstName} ${lastName[0].toUpperCase()}`)
 }
```

Calling the function in the console

```
> greet("peter","pan")
  Hello peter P
```

Example 6) define a function that has two parameters, *message and numCounts. message* parameter takes a string and parameter *numCounts* takes a number. Within the function, it prints *message* value *numCounts* time.

```javascript
function printMsg(message, numCount){
  for(counter=1; counter<=numCount ; counter++){
     console.log(`Counter ${counter} = ${message}`)
  }
}
```

```
> printMsg('Hello World!', 5)
  Counter 1 = Hello World!
  Counter 2 = Hello World!
  Counter 3 = Hello World!
  Counter 4 = Hello World!
  Counter 5 = Hello World!
```

Example 7) in some dice games like Craps, a roll of two 1's is called "Snake Eyes". It is generally not a good roll. Define a function called **isSnakeEyes** which accepts two numbers as inputs, representing two dice. If the two numbers are both 1's, please print "Snake Eyes!" otherwise print "Not Snake Eyes!"

```javascript
function isSnakeEyes(dice1,dice2){
    if (dice1===1 && dice2===1){
        console.log("Snake Eyes!")
    }
    else{console.log("Not Snake Eyes!")}
}
```

```
> isSnakeEyes(3,5)
  Not Snake Eyes!
```
```
> isSnakeEyes(3,3)
  Not Snake Eyes!
```
```
> isSnakeEyes(1,1)
  Snake Eyes!
```

Return values are values that a function returns when it has completed running the code within it. return is actually exporting the value out of the function. Also, the return of a function stops the execution of a function. Any lines after the return will not run.

Function without parameters but return value

**Example 8)** define a function that return the sum of two numbers

```javascript
function sumXY(){
  let x = 7, y = -3;
  return x+y;
```

```
    console.log("LINE AFTER RETURN")
}

// call function sumXY
let sum1 = sumXY()
console.log(sum1);
```

Result ➔ `4`

Remember, **return** stops a function, therefore, the line `console.log("LINE AFTER RETURN")` does not appear in a console.

Function with parameters and return value

Example 9) define a function that takes a number as parameter. The function will double the number and return the result.

```
function doubleNumber(number){
        let dNumber = number*2;
        return dNumber
     }
```
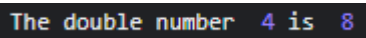
```
Alternative way:
function doubleNumber(number){
   return number*2
}
```

```
// call function doubleNumber()
let num = 4;
doubleNum=doubleNumber(num);
console.log(`The double number ${num} is ${doubleNum}`);
```

result ➔ `The double number  4 is  8`

Example 10) define a function that adds two numbers and return the sum of them. Save the return value in a variable **number1** and **number2**

```
let a = 5/9, b = 3, c = 9, d = -5;

function sumNum(num1, num2) {
let sum = num1 + num2;
return sum;
}

let number1 = sumNum(10, 30);
let number2 = sumNum(c, d);
```

Example 11) define a function that checks if the temperature outdoor is greater than 75. The function should accept a single number argument as **temperature** and return a **true** if the temperature is greater than or equal to 75 and a **false** if the temperature is less than 75.

```
function isShortsWeather(temperature){
    if (temperature>=75){return true;}
    else{return false}
}
```

```
> isShortsWeather(90)
<· true
```

```
> isShortsWeather(60)
<· false
```

Example 12) define a function that accepts a single array argument. The function should return the last item in an array. If the array is empty, the function should return **null.**

```
function lastElement(arrayInput){
    if(arrayInput ===undefined){return null}
    else if(arrayInput.length !==0){
        let lastIndex = arrayInput.length - 1;
        return arrayInput[lastIndex]
    }
    else{return null}
}
```

```
> lastElement([2,3,4,5,'dog'])
<· 'dog'
```

```
> lastElement()
<· null
```

Built-in function and methods. Math object

JavaScript provides number of built-in functions that are global functions. Some of those built-in functions are math object function.

JavaScript math object allows users to perform mathematical tasks on numbers such as to square, round off, or randomly generate a number.

Example 13) apply different Math methods to a PI value

```
let num = Math.PI;

console.log(`PI = ${num}`);

//Math.round() returns the nearest integer
console.log(`PI round =  ${Math.round(num)}`);
// Math.ceil() returns the value rounded up to its nearest integer
console.log(`PI ceil = ${Math.ceil(num)}`);
//Math.floor() returns the value rounded down to its nearest integer
console.log(`PI floor = ${ Math.floor(num)}`);
```

```javascript
// Math.trunc() returns the integer part of x
console.log(`PI trunc = ${Math.trunc(num)}`);
//Math.pow(base, exponent), returns the value of base to the power of exponent
console.log(`2^3 =  ${Math.pow(2,3)}`);
//Math.sqrt() returns the square root of a number
console.log(`square root of 81 = ${Math.sqrt(81)}`);
//Math.random() returns a random number between 0 (inclusive), and 1 (exclusive)
console.log(`Random = ${Math.random()}`);
```

```
PI = 3.141592653589793
PI round = 3
PI ceil = 4
PI floor = 3
PI trunc = 3
2^3 =  8
square root of 81 =  9
Random =  0.5492085018187762
```

Example 14) create a function that randomly generate a number between 10 and 20 inclusive

```javascript
function pickNum(){
  let pick = 10+Math.round(Math.random()*20);
  return pick;
}
console.log(pickNum());
```

Example 15) Create a function that randomly pick a color from an array.

```javascript
let colors = ['red','green','blue','yellow','orange'];

function colorPick(lastIndex){
  let pick = Math.round(Math.random()*lastIndex);
  return pick;
}

//call function colorPick()
pickedIndex = colorPick(colors.length - 1)
console.log(`The picked color is = ${colors[pickedIndex]}`)
```

Display in the console
```
The picked color is =  orange
```

Refresh the browser
```
The picked color is =  yellow
```