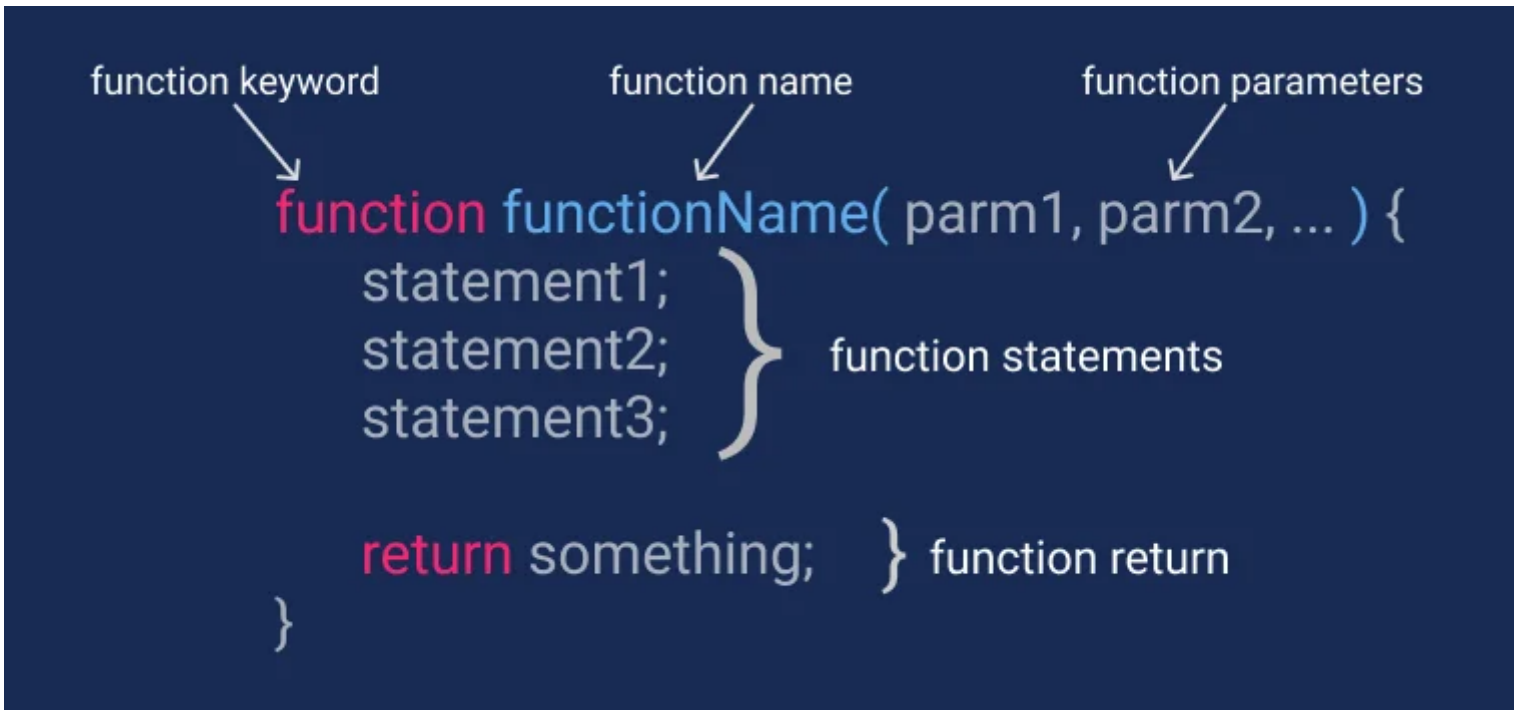
JavaScript - Function Part 2



Hemant Thapa

ADDITION

In [2]:
```
function sum(a,b){
    var c =  a + b
    return c
}
```

In [3]:
```
sum(10,20)
```

Out[3]:
```
30
```

In [4]:
```
sum(-10, -20)
```

Out[4]:
```
-30
```

In [5]:
```
function sumOfNumbers(){
    var i, res = 0;
    var number_of_params = arguments.length;
    for (i =0; i<number_of_params; i++){
        res = res + arguments[i];
    }
    return res;
}
```

In [6]:
```
sumOfNumbers(10, 20, 30)
```

Out[6]:
```
60
```

In [7]:
```
sumOfNumbers(10, -20, -90)
```

Out[7]:
```
-100
```

SUBTRACTION

In [8]:
```
function sub(a,b){
    var c = a - b
    return c
}
```

In [9]:
```
sub(10,20)
```

Out[9]:
```
-10
```

In [10]:
```
sub(50,5)
```

Out[10]:
```
45
```

MULTIPLICATION

In [11]:
```
function multiplication(a,b){
    var c = a * b
    return c
}
```

In [12]:
```
multiplication(10,20)
```

Out[12]:
```
200
```

In [13]:
```
multiplication(-30, 70)
```

Out[13]:
```
-2100
```

DIVISION

In [14]:
```
function division(a, b){
    var c = a / b
    return c
}
```

In [15]:
```
division(10,2)
```

Out[15]:
```
5
```

In [16]:
```
division(20,-7)
```

Out[16]:
```
-2.857142857142857
```

PRE DEFINE FUNCTIONS

1. parseInt

In [17]:
```
parseInt('123')
```

Out[17]:
```
123
```

In [18]:
```
parseInt('abc123')
```

Out[18]:
```
NaN
```

In [19]:
```
parseInt('123abc')
```

Out[19]:
```
123
```

In [20]:
```
parseInt('FF', 10)
```

Out[20]:
```
NaN
```

In [21]:
```
// hexadecimal
parseInt('FF', 16)
```

Out[21]:
```
255
```

In [22]:
```
parseInt('0377', 10)
```

Out[22]:
```
377
```

In [23]:
```
// octal
parseInt('0377', 8)
```

Out[23]:
```
255
```

In [24]:
```
parseInt('0377')
```

Out[24]:
```
377
```

In [25]:
```
parseInt('00377')
```

Out[25]:
```
377
```

In [26]:
```
parseInt('0x377')
```

Out[26]:
```
887
```

2. parseFloat

In [27]:
```
parseFloat('123')
```

Out[27]:
```
123
```

In [28]:
```
parseFloat('1.23')
```

Out[28]:
```
1.23
```

```
In [29]: parseFloat('1.23abc.00')
Out[29]: 1.23

In [30]: parseFloat('a.bc123')
Out[30]: NaN

In [31]: parseFloat('12a3.34')
Out[31]: 12

In [32]: parseFloat('123e-2')
Out[32]: 1.23

In [33]: parseFloat('123e2')
Out[33]: 12300

In [34]: parseFloat('1e10')
Out[34]: 10000000000
```

3. isNaN

```
In [35]: isNaN(NaN)
Out[35]: true

In [36]: isNaN(123)
Out[36]: false

In [37]: isNaN(1.23)
Out[37]: false

In [38]: isNaN('abc123')
Out[38]: true

In [39]: isNaN('1.23')
Out[39]: false

In [40]: isNaN('a1.23')
Out[40]: true
```

4. isfinite

```
In [41]: isFinite(-Infinity)
Out[41]: false

In [42]: isFinite(2)
Out[42]: true

In [43]: isFinite(1e308)
Out[43]: true

In [44]: isFinite(1e309)
Out[44]: false

In [45]: isFinite(NaN)
Out[45]: false
```

5. eval

```
In [46]: eval('var ii = 2;')

In [47]: console.log(ii)
2
```

Scope of Variables

In [1]:
```
// program to print a text
let a = "hello";

function greet () {
    console.log(a);
}

greet(); // hello
```

```
hello
```

Bubble Sort Algorithms

In [52]:
```
function bubbleSort(list) {
    for (var i = 0; i < list.length; i++) {
        var swapped = false;
        for (var j = 0; j < list.length - i - 1; j++) {
            if (list[j] > list[j + 1]) {
                // Use a temporary variable to swap the elements
                var temp = list[j];
                list[j] = list[j + 1];
                list[j + 1] = temp;
                swapped = true;
            }
        }
        // If no two elements were swapped in the inner loop, the array is already sorted
        if (!swapped) {
            break;
        }
    }
}
```

In [53]:
```
// Example usage:
var myList = [64, 34, 25, 12, 22, 11, 90];
bubbleSort(myList);
console.log(myList);
```

```
[
  11, 12, 22, 25,
  34, 64, 90
]
```