# Javascript Functions

## Road Map

1.Functions
-Declaring Functions.
-Function Parameter And Arguments.
-Using different functions together - Argument Points.
-Function Declaration vs Expression
-Arrow Function vs Normal functions

2.Constructor Function
-What is it?
-Syntax
-The new keyword.
-The this keyword.
-Example

3.Scoping
-What is it
-Global Scope
-Local Scope

4.Default Parameter
-What is it?
-Old vs New way of setting default parameters.

5.First Class and Higher Function
-What are They?
-Function as 1st class citizen:-
-Variable assignment to function.
-Returning a function from a function.
-Passing a function as an argument in another function.

6.Call and Apply Method
-Defination:- They used to call methods from various objects that had this keyword.
-You can use one method to write different objects,Key names are same.
-Call and Apply syntax
  call(objectName,  function arguments goes here)
  apply(objectName, [ function arguments goes here])

6.Bind Method
-Defination.
-Syntax

**<u>Functions</u>**
-Functions are used to perform actions and allow us to write more maintainable code.
-Define the function with the function keyword function which has parenthesis ().
-Name your function, function nameOfFunction() this is followed by curly brackets {}.
-Call function,Invoke Function,Running Function.
-They can be reused in many instances.

**Syntax**
```
function name_of_function(){
                              Code to be executed goes here
                                       }
 name_of_function()
```

**Example**
```
function name_of_function(){ console.log("Data") }
name_of_function()
```
**Output:-**Data

```
function name_of_function(){ return "Data"}
var input = name_of_function()
console.log(input)
```
**Output:-**Data

**Function Parameters & Arguments**

-Parameter is one of the variables passed in a function.

-Arguments are data passed into the method parameters.

**Syntax**

-x,y is our parameter which is the variable.

-2,3 is our arguments which is the data passed

```
function add(x,y){
        return x+y
}
add(2,3)
```

**Output:**-5

**Example**

```
function name(first_name,second_name){
                    return `Hello ${first_name} ${second_name}`
}                                                    }
 name(“Manu”,”Koech”)
```

**Output:-**Hello Manu Koech

**Function Declarations vs Function Arguments.**

-The normal way of declaring a function is function declaration.

-With function declarations we declare our function with function name only.

-We call our function with the function name.

-With function expression we declare our function in a variable.

-In our values we declare our function but without the name as we do.

-When we call our function we call it with the variable name

**Function declaration.**

```
function yourAge(birthYear){
```

```
                    return 2022 - birthYear;
                             }
let personAge = yourAge(2002);
```
**Output:**-20

**Function Expression**

```
Let yourAge = function(birthYear){
                    return 2022-birthYear
                   }
yourAge(2002)
```
**Output:**-20

**Arrow Functions**

-Arrow Functions allows us to write shorter Function Syntax.

-In arrow function our function name is not declared.

-Also we don't have to mention the return.

-If you want to put one parameter it comes before the arrow function.

-If we declare 2 parameters then we put calibraces () followed by an arrow function.

**Note:-**

**-**Never Use an Arrow function as a method.

- Arrow Function does Not  Accept this keyword as the result will be undefined

**Syntax**

**-With function Expression**

```
let functionName = ()=>{}
functionName()
```
-With function Declaration

```
function functionName =>{}
functionName()
```

**Example**

```
let yourAge =  birthyear => 2022 - birth year;
let yourCurrentAge = yourAge(2002)
```
**Output:-** 20

```
let yourAge = (birthyear,currentYear)=>currentYear - birth year
let youCurrentAge= yourAge(yourAge(2002,2022))
```
**Output:**-20

<u>**Constructor Function**</u>

-A constructor function is a regular function used to create multiple similar objects.
-Instead of creating many object with the same blueprint a constructor function is used to help us avoid repeating ourselves.

**Syntax**

-Function name starts with a capital letter.
-This keyword  is used to refer to the new object being created.
-When called we use new then name of function,We get the new instance of our object.
-The “new keyword” creates a new empty object,then assigns the argument to the properties of object

**Example**

```
function Car(engine,model,year){
        this.engine=engine,
        this.model=model,
        this.year=year,
}
const carOne = new Car(`V8`,`Toyota`,2002);
const carTwo = new Car(`V9`,`Lexus`,2022);
const carThree = new Car(`V8`,`BMW`,2005);
```

**Output:**-{`V8`,`Toyota`,2022}
**Output:**-{`V9`,`Lexus`,2022}
**Output:**-{`V8`,`BMW`,2005}

## Scoping

-This is the region where variable,functions and objects are accessible during runtime.

### Global Scope

-A variable declared outside the function is a  globalscopes.

-Values inside the function can be changed also.

**Example**

```
let hello = `Hello`;
function greet(){
    console.log(hello);
    console.log(hello = `How are you`);
}
greet()
Outputs:- Hello, How are you
```

### Local scope

-A variable declared inside a function, which means it can only be accessed inside the function.

-When called outside the function an error is output

**Example**

```
function greet(){
    let hello = `Hello`;
    console.log(hello);
    console.log(hello = `How are you`);
}
greet()
```

## Hoisting

-This is the behavior of using a function or variable before it is Declared.

-It involves moving declaration of function,variables,classes to the top of the scope before code execution.

-Hoisting is only supported in function declarations and when calling variables.

-Hoisting is not supported in Function expressions.

-Hoisting is not supported in Arrow Functions.

### Syntax

Usage -> Declaration/Assignment

### Example

```
console.log(fullName);
const fullName =`Emmanuel Koech`;
```

**Variable**

**Output:-** Emmanuel Koech

```
yourName(`Emmanuel`)
function yourName(name){
  console.log(`Hello ${name}`)
}
```

**Function Declaration**

**Output:-**Emmanuel

```
yourName(`Emmanuel`)
let yourName = function(name){
  console.log(`Hello ${name}`)
}
```

**Function Expression**

**Output:-**Error

```
yourName(`Emmanuel`)
let yourName= name=>console.log(`Hello ${name}`)
```
**Arrow Function**

**Output:-** Hello Emmanuel

## First Class and Higher order Functions

-Functions in Javascript are first class citizens which  means you can Assign/treat them as variables also they can be passed as an argument to other functions and  lastly they can be returned by another function.

**Syntax**

```
let  myFunction = ()=>{console.log(Hello)}
myFunction()
```
**Output:-**Hello

**Treating Functions as variables(Function expression)**

```
let  myFunction = ()=>{console.log(`Hello`)}
let  myOtherFunction = ()=>{console.log(`World`)}
myFunctionOther(myFunction())
```
**Output:-**Hello,  World

**Passing Functions as arguments to other functions.**

```
let  myFunction = ()=>{
        return ()=>{console.log(`Hello`)}
}
```
**Output:-**Hello

**Returning a Function from another function**

2.Pass a function as an argument to another function.

```
const hiFunction = ()=>`Hello World`;
hifunction(function(){return “Hello World”})
```

3.Return a Function from another Function.

```
const hiFunction = (greeting)=>{
        return function(name){ ` ${greetings},${name}`
}
hiFunction(`Hello`)(`Manu`)
```

-Higher Order functions are functions that use other functions arguments or return functions.