

Functions in JavaScript

- A JavaScript function is a procedure or a subprogram, i.e., a block of code that performs a certain task.
- A function is a group of statements that perform specific tasks and can be kept and maintained separately from main program.
- Functions provide a way to create reusable code packages which are more portable and easier to debug.
- A function is a block of code which will be executed only if it is called.
- Every Function has two main parts in first part we define the function and in the second part we will call that.
- A Function is a subprogram which are used to compute a value or to perform a task.
- JavaScript functions are first-class objects. Means they are very powerful in JavaScript as compared to other programming languages. They are even more powerful than objects.
- The naming convention of function is similar as variable.

In JavaScript function is invoked :-

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

Advantages of functions

- Code reusability.
- Less coding.
- Easy to find errors.

Working with functions

In order to work with functions in JavaScript we need to define a function and after defining we have to call it in order to execute the code inside function.

a) declaring / defining / creating a function

- A function definition (also called a function statement or a function declaration) includes a 'function' keyword.
- In order to declare a function, we use the 'function' keyword, followed by the 'function' name, a list of parameters, and the function body.

b) calling a function

- A Function runs only when we call it, function cannot run on its own.
- When we call a function, the function executes the code inside its body. This process is also known as invocation.
- The 'function' keyword merely creates a function but does not call it. After the "function" execution, we can call (run) the function by adding parentheses after the function's name.

LABS

Syntax of Function 1 -- Without parameters

// Creating a function with 'function' keyword --

```
function functionName(){  
    // Block of Code  
    // return(variable or expression)  
}
```

// Calling a function --

```
functionName()
```

Syntax of Function 2 -- With parameter

// Creating a function with 'function' keyword --

```
function func_Name(parameter1,parameter2){  
    // Block of Code  
    // return(variable or expression)  
}
```

// Calling a Function --

```
func_Name(argument1,argument2)
```

significance of "function" keyword in JavaScript function

- The "function" keyword is a statement for defining a function in JavaScript.
- The "function" call creates the function object and assigns it to the name given.

Example

```
function abc() {  
    alert("Hello Function");  
}  
abc();
```

Example -- Function is executed Whenever we call it

```
console.log("Statement 1");  
console.log("Statement 2");  
function abc(){  
    console.log("Function Statement");  
}  
console.log("Statement 3");  
console.log("Statement 4");  
abc()
```

Function Hoisting

- In JavaScript, it is possible to use a function before it is declared.
- Whenever we call a function before defining the function is known as function hoisting.
- The function hoisting is a mechanism that the JavaScript engine physically moves the function declarations to the top of the code before executing it.

Example -- Simple Function

```
abc();  
function abc() {  
    alert("Hoisting Function");  
}
```

Example -- Function Returning Function

A Function can also returns a function.

```
function sayHi(){  
    return function(){ return "hi";  
}  
sayHi();           // return function  
sayHi();           // return "hi"
```

Ways to define a function

There are couple of different ways by which we can define a function that includes: -

- a) Function Declaration.
- b) Function Expression or Anonymous Function.
- c) Self-Invoking Function or Anonymous Function.
- d) Function Constructor.
- e) Arrow Function.



Function Declaration

This is the most common & simple way to define a function.

Syntax

```
function functionName(){  
    // code block of function  
}  
functionName();
```

Example

```
function hello(){  
    console.log("JavaScript Function")  
}  
hello()
```



Function Expression or Anonymous Function

- When we create a function and assign it to a variable, known as function expression.
- Function hoisting is not possible with function expression.
- These types of functions is also known as Named Function Expressions (NFE) & anonymous functions.

Syntax

```
var functionName = function(){  
    // code block of function  
}  
functionName()
```

Example

```
var abc = function(){  
    console.log("Hello I am function expression");  
}  
abc();
```


Self-Invoking Function or Anonymous Function

- These types of functions will call automatically.
- This type of function has no name.

Syntax

```
(function(){  
    // code block of function  
})();
```

Example

```
(function(){  
    console.log("Self-Invoking Function");  
})();
```



Function Constructor

- The purpose of Function constructor is to create a new Function object.
- We can create instance objects by using function constructor.
- Function constructor dynamically creates objects by passing values.
- We can create instance objects by using 'new' keyword.

Syntax

```
function functionName(arg1,arg2,arg3){  
    this.arg1 = argument1;  
    this.arg2 = argument2;  
    this.arg3 = argument3;  
}  
var variableName = functionName("value1","value2","value3");
```

Example

```
function abc()  
{  
    this.name = "Mohit";  
    this.age = 34;  
}  
var Person1 = new abc();  
console.log(Person1);
```

Example -- Creating Instance Objects

```
function abc(userName,userAge)  
{  
    this.name = userName;  
    this.age = userAge;  
}  
var Person1 = new abc("Mohit",34);  
var Person2 = new abc("Rohit",30);  
console.log(Person1);  
console.log(Person2);
```

Example -- By Using Function Expression

```
var abc = function(userName,userAge){  
    this.name = userName;  
    this.age = userAge;  
}  
var Person1 = new abc("Mohit",34);  
var Person2 = new abc("Rohit",30);  
console.log(Person1);  
console.log(Person2);
```

Example -- By using 'Function' keyword

```
var abc = new Function("a","b","return a + b");  
console.log(abc(3,2));
```



Arrow Function


- An arrow function expression is a syntactically compact alternative to a regular function expression.
- Arrow function have no 'this' keyword inside that.
- It is not necessary to use 'return' keyword inside of arrow function.
- Arrow functions cannot be used as a constructor.
- Arrow functions were introduced in ES6.

Syntax

```
let functionVariable = (a, b) => a * b;
```

Before, Arrow Function

```
hello = function() {  
  return "Hello World!";  
}
```



After, Arrow Function

```
hello = () => {  
  return "Hello World!";  
}
```

Arrow Functions Return Value by Default:

```
hello = () => "Hello World!";
```

Arrow Function With Parameters:

```
hello = (val) => "Hello " + val;
```

Arrow Function Without Parentheses:

```
hello = val => "Hello " + val;
```

Example -- Arrow Function with No Argument

```
let msg = () => console.log('Hello');  
msg(); // Hello
```

Example -- Arrow Function with One Argument

```
let msg = x => console.log(x);  
msg('Hello'); // Hello
```

Example -- Arrow Function as an Expression

```
let age = 5;  
let welcome = (age < 18) ?  
  () => console.log('Young') :  
  () => console.log('Adult');  
welcome(); // Baby
```

Example -- Multiline Arrow Functions

```
let sum = (a, b) => {  
  let result = a + b;  
  return result;  
}  
let result1 = sum(5,7);  
console.log(result1); // 12
```