JavaScript

JavaScript is used as front-end as well as the back-end

Front-end: React JS

Back-end: Node JS and Express JS

The Netscape hired the Developer Brenden Eich and said to develop a scripting language for their browser. So, Brenden Eich developed a Scripting language called as Mocha in 1995 with 10days, later it is named as Live Script but as a marketing strategy the company named it as JavaScript. Later, the Netscape company handled JavaScript to ECMA (European Computer Manufacturer Association) company then it named as ECMA Script. The ECMA company released the new version of JavaScript every year in the month of June. But now the trademark of JS is owned by ORACLE.

- It is a scripting Language designed for web pages.
- It is a client-side Scripting language.
- An Interpreted, event based language.
- High level Language.
- Weekly typed and case-sensitive
- Dynamically typed.
- It is embedded directly into HTML pages.

JavaScript is one of the sub applications of the Browser.
Brenden Eich invented JS because to avoid to get the data from the server side.
Since, JS is client side scripting language it will Read, Analyze and Execute the code in the Browser itself.
JavaScript is also a Programming Language.
Weekly typed specifies need not declare the datatype.
Dynamically typed because we can reuse the code.
It is an High Level Language because it is the human readable format.

JavaScript is an Interpreted Language

Q. What id Interpreted language?

Interpreted language is the codes are read line-by-line and executed. If at all it finds an error, it will not move to the next line of the code, it rather displays runtime error.

<u>Characteristics of JavaScript</u>

- It is an Interpreted language.
- It is a Synchronous language.
- It is purely Object Based Programming Language.

Everly Browser has a Java Script Engine which is responsible to execute JavaScript code.

Tokens : Tokens are the smallest unit of any programming language.

In tokens we have:

- Keywords
- Identifiers
- Literals

Keywords	Identifiers	Literals
• Pre-defined /reserved words. • Should be lowercase.	• Can't starts with numbers (but it can have number). • Except dollar($) and underscore(_) no other special character is allowed. • Keywords can't be used as identifiers	The Data which is used in JS programs. • Number • Boolean • Null • Undefined • Object • Bigint • Symbol.

Example: var, let, const, if, while,…	Example: name, age, skills,…	Example: name="Jhon";
		age=25;

Output methods of JavaScript

document.write(): Prints the output in the browser window (without space)
document.writeln(): Prints the output in the browser window (with space)
console.log(): Prints the output in the developers window (also called developer output)
prompt(): It is a pop-up message, takes the input from the user.
alert(): It is a pop-up message, gives the alert message to the user.
confirm(): It is a pop-up message, gives the confirm messages.

Containers of JavaScript

var	let	const
var is a global scope.	let is a script scope.	const is a script scope.
We can declare multiple variable with same name(The most recently created variable will be used)	We cannot declare two variables with the same name within a block.	We cannot declare two variables with the same name within a block.
The value of the variable can be modified.	The value of the variable can be modified.	The value of the variable cannot be modified.

var a=10;
a=20;
console.log(a);
	let a=10;
	a=20;
	console.log(a);
		const a=10;
		a=20;

//20	//20	**console.log(a);** //we cannot modify the data with new value. //type error
We can declare var without initialization. **Ex: var x;**	We can declare let without initialization. **Ex: let y;**	We cannot declare const without initialization. **Ex: const z;** //syntax error

The variable declared using var Belongs to global scope(window), we can access them using window object.
 	
Ex: var a=10;
console.log(window.a);
//10
	The variable declare using let does not belongs to global scope we cannot use them with the help of window. (script/block)
	Ex: let b=20;
	Console.log(window.b);
	//undefined
		The variable declare using const does not belongs to global scope we cannot use them with the help of window. (script/block)
		Ex: const c=30;
		Console.log(window.c);
		//undefined

Window object

When a JS file is given to browser by default a global window object is created and the reference is stored in window variable.
The global object consist of pre-defined members (functions and variables) which belong to browser window.
Any member we declare var in JS file is added inside global window object by JS engine. So we can use member with the help of window.
Any members(functions and variable) created in global scope will be added into the window object implicitly by JS Engine

```
var a=10; //var a is added into the global window object.
console.log(window.a); //10
```

It present in global execution context.
Window is a variable, store the address of object and its point global context which present in the heap memory.
We can access window object using window variable.
Anything inside window we can access using dot operator. Ex: window.a
Whenever we create variable using const does not belong to global object and are not added to window object.
The variable declared using var belongs to global object(window), We can access them using window Object.
The variable declared using let and const does not belong to global object and are not added to window object (cannot use them with the help of window).

Hoisting: Utilizing the variable before declaration and initialization is called as Hoisting.

Hoisting can be achieved by var, because var is a global scope or global variable.
Hoisting cannot be achieved by let and const, because let and const are script scope.
Whenever we hoist var the result is undefined.

Whenever we try to hoist let and const the result is Uncaught ReferrenceError.

<u>Temporal Dead Zone (TDZ)</u>: In the process of hoisting the time taken between Utilization of the variable before declaration and initialization.
TDZ is achieved only in let and const.
Because, whenever we try to hoist let and const the result is Uncaught ReferrenceError.
TDZ cannot be achieved in var.
Because, whenever we hoist var the result is undefined.

DATATYPES IN JAVASCRIPT:

1.PRIMITIVE DATATYPE : Which is immutable

number, string, boolean, null, undefined ,bigInt ,Symbol.

2.NON-PRIMITIVE DATATYPE: which is mutable.

functions, arrays,

objects =>date object ,math object.

<u>Operators</u>:

Operators are the symbol which performs specific operations on the operators.

Types of operators

Arithmetic Operator
Assignment Operator
Relational / Compressional Operator
Logical Operator
Ternary / Conditional Operator.

Arithmetic Operators

+ - * / %

Assignment Operators

= += -= *= /= %= **= //=

Relational / Compressional Operator

< > <= >= == != === !==

Logical Operator

&& || !

Ternary / Conditional Operator

(condition) ? expression1 : expression2

Functions

Functions are the block of statements which will get executed whenever it is called or invoked.

Types of Functions

- Anonymous Function
- Named Function
- Function with expression
- Nested Function
- Arrow Function
- Higher Order Function
- Callback Function
- Immediate Invoke Function
- Generator Function

Anonymous Function: The function which does not have function name is called as Anonymous Function.

```
Syntax:         function    ( ) {
                        Statements;
                }( );
```

Named Function: The function which has function name is called as Named Function.

```
Syntax:         function    function_name ( ) {
                        Statements;
                }
                function_name( );
```

Function with Expression: The function which is assigned as a value to a variable then the expression is called as Function with Expression.

```
Syntax:         var  variable_name   =   function    function_name ( ) {
                                          Statements;
                                          }
                                          variable_name( );
```

Arrow Function: A function having a fat arrow is called as Arrow function. To reduce the function syntax we will go for Arrow function.
this keyword doesn't work in Arrow Function.

```
Syntax:         variable_name = ( ) = > {
                        Statements;
                }
variable_name( );
```

Nested Function: A function inside another function is called as Nested Function.
Note: One inside another function. Not one inside many function.

```
Syntax:         function    function_name1 ( ) {
                        Statements;
                            function    function_name2( ) {
                              Statements;
```

```
                                    function    function_name3( ) {
                                        Statements;
                                    }
                            return function_name3( );
                            }
                return function_name2( );
                }
                function_name1( )( )( );
```

In Nested Function we can achieve Lexical Scope / Scope chain

Lexical Scope: The ability of the JavaScript engine to search of the variable in the global scope when it is not available in the local scope. It is also known as Scope chain.

Closure: Closure holds the data of the parent function which is required by the child function
Not all the data of the parent function.
Closure object address will be stored in the child function for further/future utility.

Immediate Invoke Function Execution: IIFE is a function executes a function which is enclosed within the parenthesis ( ).

```
Syntax:         (function function_name( ){
                        Statements;
                })
                ( );
```

IIFE should be called immediately right after the declaration.
IIFE cannot be reused, it is only one time usable function.
IIFE can have function name but it cannot be called with function name.
Anonymous function, Named function, Nested function, Arrow function can be IIFE.

Higher order function: The function which takes another function as an argument is called as Higher order function.
Callback function: The function which is passed as an argument to the function / Higher order function is called as Higher order function.

```
Syntax:         function function_name(para1,para2)
                {
                        Para2( );
                        Return;
                } fuction_name(argu1,function ( ) {  } );
```

NOTE:
Higher order function accepts callback function as an argument.
Higher order function calls the call back function.

Array Methods

push( ), pop( ), unshift( ), shift( ), toString( ), slice( ), splice( ), concat( ),fill( ), sort( ), includes( ), indexOf( ), lastindexOf( ), some( ), every( ).

Array Methods

map()
reduce()
filter()

map()- The map() method is one of the array method.
A map() is also a Higher order function which accepts the callback functions,
where that callback function accepts three arguments.

element
index
array

Syntax: array.map((element,index,array)=> { })

Example:

```
arr=[15,25,35,45,55,65]
arr.map((x,y,z)=>{
console.log(x,y,z)
})
```

Output:

```
15 0 (6) [15, 25, 35, 45, 55, 65]
25 1 (6) [15, 25, 35, 45, 55, 65]
35 2 (6) [15, 25, 35, 45, 55, 65]
45 3 (6) [15, 25, 35, 45, 55, 65]
55 4 (6) [15, 25, 35, 45, 55, 65]
65 5 (6) [15, 25, 35, 45, 55, 65]
```

In the callback, only the array element is required. Usually some action is performed on the value and then a new value is returned.
The map() method acts similar to the for loop which fetch each elements of an array and perform the specified task.
The map() returns the result in the form of array.

Example1:
```
var add=arr.map((x)=>{return x+25});
console.log(add)
```

Output: (6) [40, 50, 60, 70, 80, 90]

Example2:
```
var add=arr.map((x)=>{return x>25});
console.log(add)
```

Output: [false, false, true, true, true, true]

filter()- The filter() method is one of the array method.
The filter() is also a Higher order function which accepts the callback functions.

The filter() method call the predicate function one time for each element in the array.
The filter() methos helps to filter individual element of an array based on the condition, we pass in filter method as a callback function and it returns the value in the form of array.

Example:
```
arr=[15,25,35,45,55,65];
var a=arr.filter((y)=>{return y>25});
console.log(a)
```

```
//[35, 45, 55, 65]
```

Note: (Example for thisarg in filter)
```
  const musics = [
   {
     name: 'stinks like unwashed adolescent',
     tags: ['teen', 'energetic', 'excelent']
   },
   {
     name: 'After beethovens 4th',
     tags: ['older', 'moving', 'excelent']
   },
   {
     name: 'wap',
     tags: ['hip-hop', 'pounding', 'excelent']
   }
  ]

  const filterTags = function(element, index, arr) {
    console.log(this.tag)
    return element.tags.includes(this.tag)
  }

  console.log(musics.filter(filterTags, {tag:'teen'}))
```

reduce()- The reduce() method is one of the array method.
The reduce() is also a Higher order function which accepts the callback functions.
That call back functions accepts two parameters, that is: (accumulator, value)
accumulator fetch the first element of an array and stores the final result.
value fetch the second element of an array and the perform the specified operation.

Example:
```
var arr=[15,25,35,45,55,65];
var result=arr.reduce((accu, value)=>{
        return accu+value;
})
console.log(result);
```

Output: 240

Example:
```
var arr=[12, 15, 10, 30];
var result=arr.reduce((accu, value)=>{
return ans=accu+value;
},10)
console.log(ans)
```

Output: 77

String Methods

length, repeat, toUppercase( ), toLowercase( ), indexOf( ), includes( ), replace( ), replaceAll( ), substring( ), slice( ), substr( ), concat( ), trim( ), trimStart( ), trimEnd( ), padStart( ), padEnd( ), charAt( ), charCodeAt( ), endswith( ), startswith( ),reverse().

Object: Object is associated with properties separated with commas and every property is in the form of key and value pair.

Syntax:
```
let object_name={
        Key1: value1,
        Key2: value2,
        .
        .
}
```

Every property of an is Object is displayed in the form of array but cannot be accessed through index values.
The JavaScript Objects keys accepts values of all datatype.
Object is mutable mean Object address cannot be changed.
We can fetch the key and value.

Value can be fetched using one these notations

.	dot notation
[ ]	Square notation

We can reinitialize the value but key cannot be reinitialized.
We can remove the value but key cannot be removed.
Nested Object can be created.

The function can also be given as a value to a variable.

JavaScript Objects can be created using these below mentioned ways:

Object Literals

Syntax:
```
let object_name = {
                Key1: value1,
                Key2: value2,
                .
                .
        }
```

new keyword

Syntax:

```
let  object_name = new  Object( );
```

Constructor function

Advantage: Using Constructor function multiple objects can be created at a time.

Syntax:

```
function  function_name(key1, key2, …){
        this.key1=key1;
        this.key2=key2;
        .
        .
}
let object_name1= new  function_name(value1, value2, …);
let object_name2= new  function_name(value1, value2, …);
```