JAVASCRIPT BY SARVESH

What is JavaScript?

Javascript is a high-level , object based ,programming language. Javascript is a interpreted, user friendly ,client-side scripting language.

History of javascript:

javascript developed by Brenden Eich-1995

First name of javascript --> MOCHA

next --> LIVESCRIPT

next--> JAVASCRIPT

FEATURES OF JAVASCRIPT:

1)Light Weight

2)interpreted language

3)Dynamically typed language

4)Weakly typed language

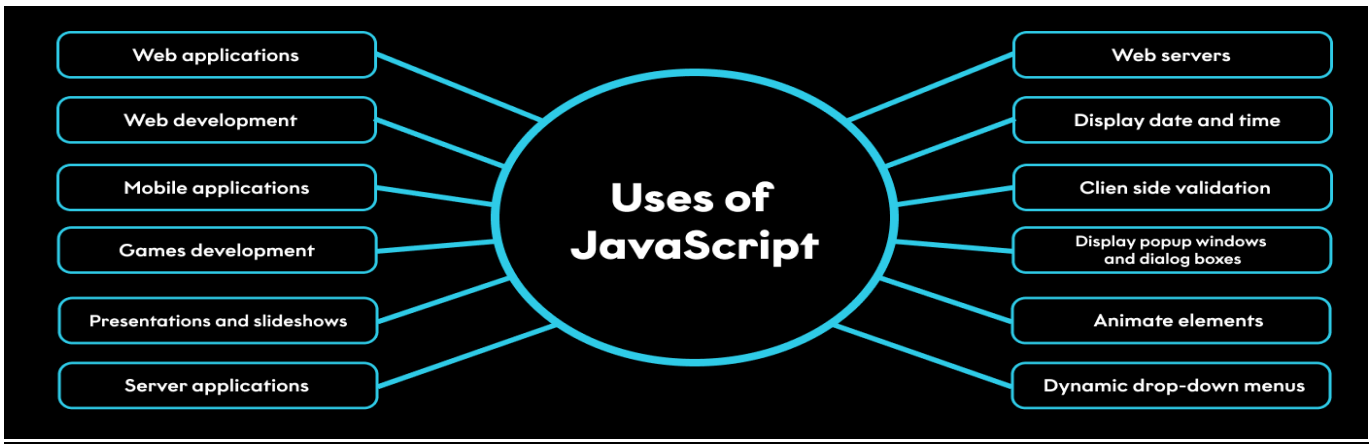
5)client-side scripting language

6)Synchronous language

Javascript characteristics:

1. *Client-side-Scripting language - no compiler and without help of server logic we can update the data*
2. *High-level language - user-friendly*
3. *Interpreted language – line by line execution*
4. *Loosely typed – no strong syntax*
5. *Dynamic language – can change the datatype during the runtime*
6. *Object-based language – In JavaScript everything is object*

Uses of JavaScript:



Difference between Java and JavaScript

Java	javascript
s a strongly typed language and variables be declared first to use in the program. In the type of a variable is checked at compile-	cript is a loosely typed language and has a relaxed syntax and rules.
s an object-oriented programming language rily used for developing complex logic ations.	cript is a scripting language used for ng interactive and dynamic web pages.
pplications can run in any virtual ne(JVM) or browser.	cript code used to run only in the browser, w it can run on the server via Node.js.
ts of Java are class-based even we can't any program in java without creating a	cript Objects are prototype-based.
rogram has the file extension ".Java" and ates source code into bytecodes which are ted by JVM(Java Virtual Machine).	cript file has the file extension ".js" and it is reted but not compiled, every browser has vascript interpreter to execute JS code.
upports multithreading, which allows le threads of execution to run concurrently a single program.	cript does not support multithreading, gh it can simulate it through the use of orkers.
as a rich set of libraries and frameworks for ng enterprise applications, such as Spring, nate	cript has a wide variety of libraries and works for building web applications, such act, Angular, and Vue.
s mainly used for backend. s statically typed, which means that data are determined at compile time. ses more memory	cript is used for the frontend and backend cript is dynamically typed, which means ata types are determined at runtime. cript uses less memory.

Note: The file extension for the javascript is **.js**

Attaching the javascript file to html file:

We can attach the js file with html file in 2 ways.

1.**Internal Way**

2.External Way

1.Internal Way

By writing the code of javascript inside <script></script> tag

Syntax:

```
<body>
    <h1>hello world</h1>
    <script>
        // javascript code…….
    </script>
</body>
```

2. External Way

1. To write javascript externally to html file, we need to create an external javascript file with extension as **.js (**ex: index.js**)**
2. After that link that javascript file to the html by using script tag.

Example:

```
<body>
    <script src = "./index.js "></script>
</body>
```

Printing Statements in javascript:

In javascript we have 2 types of printing statements

1.document.write();

2.console.log();

1.document.write()

document.write() is a printing statement which is used to see the output in the web browser (client purpose)

2.console.log()

Console.log() is a printing statement which is used to see the output in the console window (developer view)

Here, console. is an object , dot is a period (or) access operator and log is a function and member of console. that accepts argument as data to print in console.

TOKENS:

In JavaScript, a token is the smallest unit of a program that is meaningful to the interpreter. JavaScript code is made up of various types of tokens, including

JAVASCRIPT BY SARVESH

1. **Keywords:**
   Reserved words that have special meanings in the language. Examples include `var`, `if`, `else`, `for`, `while`, etc.
   NOTE: Every keyword must be in the lower case and it is not used as Identifiers.

2. Identifiers:

These are names that are used to identify variables, functions, and other objects. For example, the identifier `myVariable` could be used to identify a variable that stores the value 5.

Rulers: -an identifiers can't start with number. -We can't use keywords as identifiers. -Except $, _ no other Special Characteristics are allowed.

3.Literals:

Data which is used in the js programming is called as literals. For example, the literal `5` is a numeric literal, and the literal `"Hello, world!"` is a string literal.

4. Operators:

These are symbols that are used to perform operations on operands. For example, the operator `+` is used to add two numbers together.

JavaScript Variable

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

1. Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
2. After first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

Correct JavaScript variables

```
1. var x = 10;
2. var _value="sonoo";
```

Incorrect JavaScript variables

```
1. var  123=30;
2. var *aa=320;
```

i. .

JAVASCRIPT BY SARVESH

Javascript Data Types

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

- Number
- String
- Undefined
- Null
- Boolean
- bigint

JavaScript non-primitive data types

The non-primitive data types are as follows:

- Object
- Array
- Function

Javascript var,let,const

	Declaration	Initialization	Declaration and initialization	Re-Declaration	Re-Initialization	Re-declaration & Re-Initialization
var	✓	✓	✓	✓	✓	✓
let	✓	✓	✓	✗	✓	✗
const	✗	✗	✓	✗	✗	✗

GEC (Global Execution Context)

- Global Execution Context is a block of memory
- The global Execution context has two parts
    1. Variable Phase
    2. Execution Phase / Functional Phase
- JavaScript engine generally uses two phases to execute a JS code.

ii. **Phase I (Variable Phase):** All the memory is allocated for the declaration in top to bottom order and assignment with the default value undefined in variable area of global execution context.

Phase II (Execution Phase): All the instruction get executed in top to bottom order in execution area of global execution context

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
```

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.

Temporal Dead Zone (TDZ): 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.

JavaScript Functions:

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

JavaScript Function Syntax

A JavaScript function is defined with the `function` keyword, followed by a **name**, followed by parentheses **()**.

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:
(*parameter1, parameter2, ...*)

The code to be executed, by the function, is placed inside curly brackets: **{}**

Example:

```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Function Invocation

The code inside the function will execute when "something" **invokes** (calls) the function:

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

Function Return:

When JavaScript reaches a `return` statement, the function will stop executing.If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

Why Functions?

With functions you can reuse code

You can write code that can be used many times.

You can use the same code with different arguments, to produce different results.

Types of Functions in JS:

1. Anonymous function
2. Named function
3. Function with expression
4. Nested function
5. Immediate invoking function
6. Arrow function
7. Higher order function
8. Callback function

1.Anonymous function

A function without name is known as Anonymous function

Syntax :

```
function(parameters) {
                    // function body
                   }
```

2.Named Function

A function with name is called as named function

Syntax :

```
function functionName(parameters) {
        // function body
        }
```

3 . Function with expression

It is the way to execute the anonymous function

Passing whole anonymous function as a value to a variable is known as function with expression.

The function which is passed as a value is known as first class function

```
EX: let x=function(){
//block of code
}
x();
```

4.Nested function

A function which is declared inside another function is known as nested function.

Nested functions are unidirectional i.e., We can access parent function properties in child function but vice-versa is not possible.

The ability of js engine to search a variable in the outer scope when it is not available in the local scope is known as lexical scoping or scope chaining.

Whenever the child function needs a property from parent function the closure will be formed and it consists of only required data.

A closure is a feature of JavaScript that allows inner functions to access the outer scope of a function. Closure helps in binding a function to its outer boundary and is created automatically

whenever a function is created. A block is also treated as a scope since ES6. Since JavaScript is event-driven so closures are useful as it helps to maintain the state between events.

```
Function parent(){
        let a=10;
        function child(){
        let b=20;
        console.log(a+b);
}
child ();
}
parent ();
```

JavaScript currying

Calling a child function along with parent by using one more parenthesis is known as java script currying

Example:

```
Function parent () {
        let a=10;
        function child () {
        let b=20;
        console.log(a+b);
}
return child;
}
parent () ();
```

==➔JavaScript currying

Immediate invoking function(IIF):

A function which is called immediately as soon as the function declaration is known as IIF

We can invoke anonymous function by using IIF

Example:

```
(function () {
        console.log("Hello");
        })();
```

Arrow function:

It was introduced in ES6 version of JS.

The main purpose of using arrow function is to reduce the syntax.

Example:

```
Let x= (a, b) =>console.log(a+b);
Let y=(a,b)=>{return a + b };
```

Rules to write arrow function:-

1) In arrow function no need of writing 'function' keyword
2) In arrow function no need of writing function_name
3) we can neglect {} whenever we have only 1 printing statement
4) we can neglect () whenever we have only 1 parameter
5) we can also neglect () whenever we dont have any parameter but it has to be replaced with '_'
6) we can write return type function even without return keyword using arrow function
7) if we are using {} it is mandatory to use return keyword

Higher order function(HOF):

A function which accepts a function as a parameter is known as HOF
It is used to perform multiple operations with different values.

Example:

```
Function hof (a, b, task){
Let res=task(a,b);
return res;
};
Let add=hof(10,20,function(x , y){
return x + y;
}
Let mul = hof(10,20,function(x , y){
return x*y;
}
Console.log(add());
Console.log(mul());
```

Callback function:

A function which is passed as an argument to another function is known as callback function.
The function is invoked in the outer function to complete an action
Example :

```
function first(){
Console.log(“first”);
}
function second(){
Console.log(“third”);
}
function third(callback){
Console.log(“second”);
Callback()
}
first();
```

third(second);

STRING CONCEPT IN JAVASCRIPT

STRING:-Collection of characters (or) bunch of characters we called it as string

String methods:-

String.length
String.slice()
String.substring()
String.substr()
String.replace()
String.replaceAll()
String.toUpperCase()
String. toLowerCase
String.concat()
String.trim()
String.trimStart()
trimEnd()
padStart()
padEnd()
String charAt()
String charCodeAt()
String split()

JAVASCRIPT ARRAYS

ARRAYS:- Array is collection of different elements.Array is heterogrnous in nature.

- In javascript we can create array in 3 ways.
    1. By array literal

2.By using an Array constructor (using new keyword)

1) JavaScript array literal:-

- The syntax of creating array using array literal is: var arrayname=[value1,value2.....valueN];
- As you can see, values are contained inside [ ] and separated by , (comma).
- The .length property returns the length of an array.

Ex:-**<script>**
var emp=["Sonoo","Vimal","Ratan"]; for (i=0;i**<emp.length**;i++){
document.write(emp[i] + "**
**");

```
}
</script>
```

2) JavaScript array constructor (new keyword):-
Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.
Ex:-

```
<script>
var emp=new Array("Jai","Vijay","Smith"); for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

Output:- Jai
Vijay Smith

Java script Array methods:

push() : It will insert an element of an array at the end
unshift() : It will insert an element of an array at the first
pop() : It will remove elements of array from the end
shift() : It will remove elements of array from the start
indexOf() :It will return a index of particular element
Includes() : It will check whether the particular element is present in array or not.
At() : It will return the element which is present in particular index.
Slice() : It will give slice of an array and it will not affect the original array.
Splice() : It is used to add or remove elements from an array
Join() : It is used to join all elements of an array into a string.
Concat() : It is used to join/concat two or more arrays.
toString() : It converts all the elements in an array into a single string and returns that string.
Reverse() : It is used to reverse an array.

JAVASCRIPT OBJECTS

JavaScript object is a non-primitive data-type that allows you to store multiple collections of data.

The syntax to declare an object is:

```
const object_name = {
  key1: value1,
  key2: value2
}
```

Here, an object `object_name` is defined. Each member of an object is a **key: value** pair separated by commas and enclosed in curly braces `{}`.

Example:

```
// object creation
const person = {
    name: 'John',
age: 20
};
console.log (typeof person); // object
```

Accessing Object Properties

You can access the **value** of a property by using its **key**.

1. Using dot Notation

Here's the syntax of the dot notation.

```
objectname.key
```

Example:

```
const person = {
    name: 'John',
    age: 20,
};


// accessing property
console.log(person.name); // John
```

2. Using bracket Notation

Here is the syntax of the bracket notation.

```
objectName[“key”]
```

Example:

```
const person = {
    name: 'John',
    age: 20,
};
// accessing property
console.log(person["name"]); // John
```

JavaScript Nested Objects

An object can also contain another object. For example,

```
// nested object
const student = {
    name: 'John',
    age: 20,
    marks: {
        science: 70,
        math: 75
    }
}
// accessing property of student object
console.log(student.marks); // {science: 70, math: 75}
// accessing property of marks object
console.log(student.marks.science); // 70
```

Destructuring in js

The destructuring assignment syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.

Example: (Array Destructuring)

```
let arr = [10,20,30,40,50]
let [a,b,c,d,e] = arr //Destructuring
console.log(a);

let arr = [10,20,[1000,"hello",["hii",2000]]]
let [a,b,[c,d,[e,f]]] = arr //Destructuring
console.log(e);
```

Example: (Object Destructuring)

```
let obj={
    ename:"Raj",
    company:"Google",
    sal:60000
}
let {ename,company,sal} = obj //Destructuring
console.log(ename);


let obj={
    ename:"Raj",
    company:"Google",
    sal:60000,
    games:{
        outdoor:["cricket","volleyball","football"],
        indoor:["ludo","chess"]
    }
}
//Destructuring
let {ename,company,sal,games:{outdoor:[a,b,c],indoor:[x,y]}} = obj
console.log(ename);
console.log(a,x);
```

Advance array methods

Filter() :

filter() is a HOF which will check a particular condition for each element in the original array. If the element satisfy the condition the element will be pushed to the new array.

Syntax :

```
arr.filter((ele,index,arr)=>{
return condition
})
```

Example :

```
let numbers = [1, 2, 3, 4, 5];
let evenNo = numbers.filter((x) => x % 2 === 0);
console.log(evenNo); //[2, 4]
```

callback: This is the function that is used to test each element in the array. It takes three arguments:

- **element**: The current element being processed in the array.
- **index (optional):** The index of the current element being processed.
- **array (optional):** The array filter was called upon.

Map() :

The map() method in JavaScript is used to create a new array by applying a function to every element in an existing array. It does not modify the original array. Instead, it returns a new array with the modified elements.

EX:

```
const numbers = [1, 2, 3, 4, 5];
const doubledNumbers = numbers. Map(number => number * 2);
console.log(doubledNumbers); // Output: [2, 4, 6, 8, 10]
```

Reduce() : the reduce() is a HOF which will return a single value from the original array. if no initial value is given, the accumulator will be assigned with the first value of the array

SYNTAX:-

```
arr.reduce((acc,ele,index,arr)=>{
//statements
},init)
```

Example: -

```
let arr = [1, 2, 3, 4, 5];
let sum = arr.reduce(( accumulator,currentValue,index,arr) => {
console.log( accumulator, currentValue,index);
return accumulator+currentValue
},100);
console.log(sum);//115
```

Object: -

A JavaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

Creating Objects in JavaScript:

There are 2 ways to create objects.

1.By object literal**.**

2.By creating instance of Object directly (using new keyword).

1)JavaScript Object by object literal:

The syntax of creating object using object literal is given below:

VariableName object=

{

property1:value1,

property2:value2,

propertyN:valueN

}

As you can see, property and value is separated by : (colon).

Example of creating object in JavaScript.

<script>

Let emp={

id:102,

name:"Kumar", salary:40000

}

document.write(emp.id+" "+emp.name+" "+emp.salary);

</script>

2) By creating instance of Object:

The syntax of creating object directly is given below:

var objectname=new Object();

Here, **new keyword** is used to create object.

Example of creating object directly.

<script>

var emp=new Object(); emp.id=101; emp.name="Ravi";
emp.salary=50000;

document.write(emp.id+" "+emp.name+" "+emp.salary);

</script>

In JavaScript, to access and manipulate object properties: dot notation

Dot notation

Dot notation is the most common way to access object properties. It uses a period (.) to access the value of a property by its key.

Here's an example:

```
Const person = {  name: 'John',  age: 30,  address: {

street: '123 Main St',

city: 'New York'

}

};

console.log(person.name); // John  console.log(person.address.city); // New York
```

OBJECT METHODS:

1.keys : It will return array of keys

2.Values : It will return array of values

3.Entries : It will return array of keys and values

Example:-

```
let obj4={

ename:"lavanya",

id:123,

sal:20000

};

let obj5={

ename1:"bujji",  id1:12,  sal1:40000

};

console.log(Object.keys(obj4));//array of keys
console.log(Object.values(obj4))//array of values
console.log(Object.entries(obj4))//array of keys and values
```

Loops in js

forEach - loops through a block of code a number of times

for/in - loops through the properties of an object

for/of - loops through the values of an iterable object

forEach():-

The forEach() method calls a function for each element in an array.

The forEach() method is not executed for empty elements.

Example:

```
const array1 = ['a', 'b', 'c’];

array1.forEach((element) => console.log(element));
```

Output:- 1

2

3

4

5

For of loop:-

Iterable objects are objects that can be iterated over with for..of. <script>
Ex:-const name = "W3Schools";

let text = ""

for (const x of name){ text += x + "
";

}

For in loop:-

The JavaScript for in loop iterates over the keys of an object

Ex:-

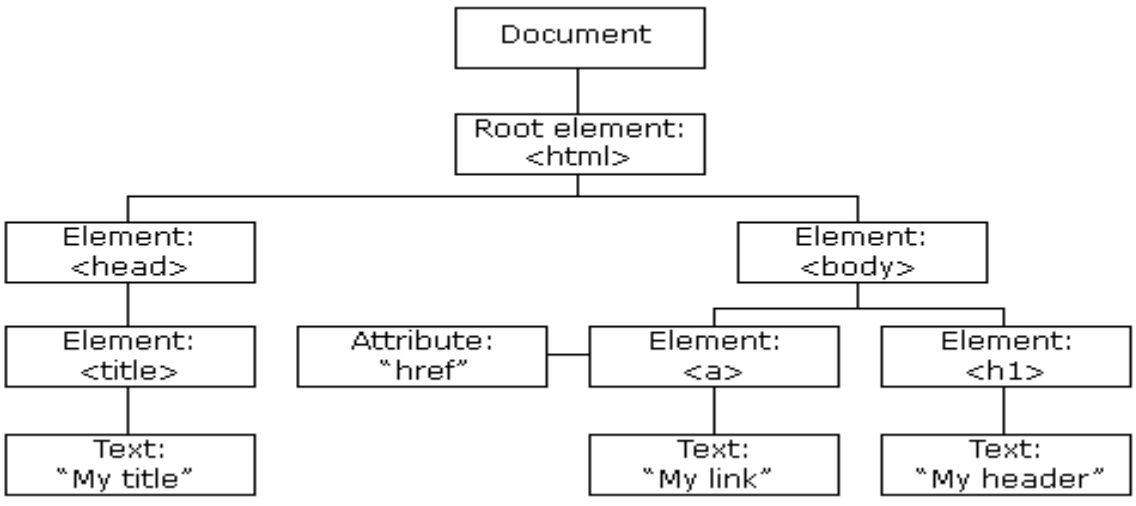
```
const object = { a: 1, b: 2, c: 3 };

for (const property in object) {

console.log(`${property}: ${object[property]}`);

}

// Expected output:

// "a: 1“

// "b: 2“

// "c: 3“
```

DOM

In JavaScript, the DOM (Document Object Model) is a programming interface for web documents. It represents the structure of a document as a tree-like model where each node is an object representing a part of the document, such as elements, attributes, and text.

When a web page is loaded, the browser creates
a Document Object Model of the page.

The HTML DOM model is constructed as a tree of Objects:



DOM Methods: -

Methods used to target HTML elements in JavaScript file

getElementById(id): This method allows you to retrieve an element from the document by its unique id.

getElementsByClassName(className): This method returns a collection of all elements in the document with a specified class name.

getElementsByTagName(tagName): Returns a collection of elements with the specified tag name.

document.querySelector(selector): Returns the first element that matches a specified CSS selector.

document.querySelectorAll(selector): Returns a NodeList of all elements that match a specified CSS selector.