Introduction to JavaScript

What is JavaScript ?

JavaScript is a lightweight, cross-platform, single-threaded, and interpreted compiled programming language. It is also known as the scripting language for webpages. It is well-known for the development of web pages, and many non-browser environments also use it.

JavaScript is a weakly typed language (dynamically typed). JavaScript can be used for Client-side developments as well as Server-side developments. JavaScript is both an imperative and declarative type of language. JavaScript contains a standard library of objects, like Array, Date, and Math, and a core set of language elements like operators, control structures, and statements.

- Client-side: It supplies objects to control a browser and its Document Object Model (DOM). Like if client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation. Useful libraries for the client side are AngularJS, ReactJS, VueJS, and so many others.
- Server-side: It supplies objects relevant to running JavaScript on a server. For if the server-side extensions allow an application to communicate with a database, and provide continuity of information from one invocation to another of the application, or perform file manipulations on a server. The useful framework which is the most famous these days is node.js.
- Imperative language – In this type of language we are mostly concerned about how it is to be done. It simply controls the flow of computation. The procedural programming approach, object, oriented approach comes under this as async await we are thinking about what is to be done further after the async call.
- Declarative programming – In this type of language we are concerned about how it is to be done, basically here logical computation requires. Her main goal is to describe the desired result without direct dictation on how to get it as the arrow function does.

How to Link JavaScript File in HTML ?

JavaScript can be added to HTML file in two ways:

- Internal JS: We can add JavaScript directly to our HTML file by writing the code inside the <script> tag. The <script> tag can either be placed inside the <head> or the <body> tag according to the requirement.
- External JS: We can write JavaScript code in another files having an extension.js and then link this file inside the <head> tag of the HTML file in which we want to add this code.

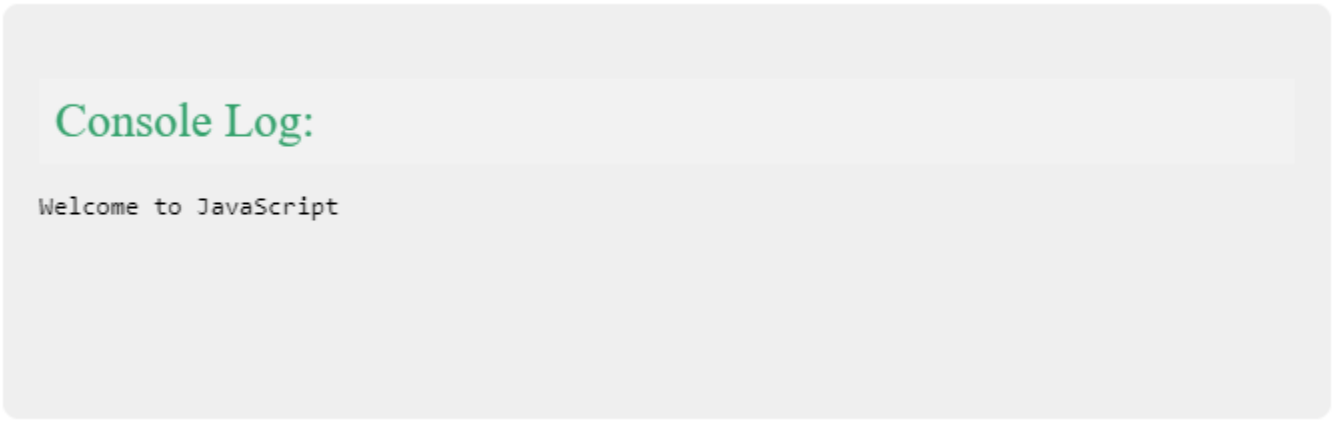
Syntax:

```
<script>
    // JavaScript Code
</script>
```

Example

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>
        Basic Example to Describe JavaScript
    </title>
</head>

<body>

    <!-- JavaScript code can be embedded inside
        head section or body section -->
    <script>
        console.log("Welcome to JavaScript");
    </script>
</body>

</html>

```

Output: The output will display on the console.



History of JavaScript

It was created by Brendan Eich in 1995. He was an engineer at Netscape. It was originally going to be named LiveScript but was renamed. Unlike most programming languages, JavaScript language has no concept of input or output. It is designed to run as a scripting language in a host environment, and it is up to the host environment to provide mechanisms for communicating with the outside world. The most common host environment is the browser.

Features of JavaScript

According to a recent survey conducted by Stack Overflow, JavaScript is the most popular language on earth. With advances in browser technology and JavaScript having moved into the server with Node.js and other frameworks, JavaScript is capable of so much more. Here are a few things that we can do with JavaScript:

- JavaScript was created in the first place for DOM manipulation. Earlier websites were mostly static, after JS was created dynamic Web sites were made.
- Functions in JS are objects. They may have properties and methods just like other objects. They can be passed as arguments in other functions.
- Can handle date and time.
- Performs Form Validation although the forms are created using HTML.
- No compiler is needed.

Applications of JavaScript

- **Web Development:** Adding interactivity and behavior to static sites JavaScript was invented to do this in 1995. By using AngularJS that can be achieved so easily.
- **Web Applications:** With technology, browsers have improved to the extent that a language was required to create robust web applications. When we explore a map in Google Maps then we only need to click and drag the mouse. All detailed view is just a click away, and this is possible only because of JavaScript. It uses Application Programming Interfaces(APIs) that provide extra power to the code. The Electron and React are helpful in this department.
- **Server Applications:** With the help of Node.js, JavaScript made its way from client to server and Node.js is the most powerful on the server side.
- **Games:** Not only in websites, but JavaScript also helps in creating games for leisure. The combination of JavaScript and HTML 5 makes JavaScript popular in game development as well. It provides the EaseJS library which provides solutions for working with rich graphics.
- **Smartwatches:** JavaScript is being used in all possible devices and applications. It provides a library PebbleJS which is used in smartwatch applications. This framework works for applications that require the Internet for their functioning.
- **Art:** Artists and designers can create whatever they want using JavaScript to draw on HTML 5 canvas, and make the sound more effective also can be used p5.js library.
- **Machine Learning:** This JavaScript ml5.js library can be used in web development by using machine learning.
- **Mobile Applications:** JavaScript can also be used to build an application for non-web contexts. The features and uses of JavaScript make it a powerful tool for creating mobile applications. This is a Framework for building web and mobile apps using JavaScript. Using React Native, we can build mobile applications for different operating systems. We do not require to write code for different systems. Write once use it anywhere!

Limitations of JavaScript

- **Security risks:** JavaScript can be used to fetch data using AJAX or by manipulating tags that load data such as <img>, <object>, <script>. These attacks are called cross-site script attacks. They inject JS that is not part of the site into the visitor's browser thus fetching the details.

- **Performance:** JavaScript does not provide the same level of performance as offered by many traditional languages as a complex program written in JavaScript would be comparatively slow. But as JavaScript is used to perform simple tasks in a browser, so performance is not considered a big restriction in its use.
- **Complexity:** To master a scripting language, programmers must have a thorough knowledge of all the programming concepts, core language objects, and client and server-side objects otherwise it would be difficult for them to write advanced scripts using JavaScript.
- **Weak error handling and type checking facilities:** It is a weakly typed language as there is no need to specify the data type of the variable. So wrong type checking is not performed by compile.

Why JavaScript is known as a lightweight programming language ?

JavaScript is considered lightweight due to the fact that it has low CPU usage, is easy to implement, and has a minimalist syntax. Minimalist syntax as in, has no data types. Everything is treated here as an object. It is very easy to learn because of its syntax similar to C++ and Java.

A lightweight language does not consume much of your CPU's resources. It doesn't put excess strain on your CPU or RAM. JavaScript runs in the browser even though it has complex paradigms and logic which means it uses fewer resources than other languages. For example, NodeJs, a variation of JavaScript not only performs faster computations but also uses fewer resources than its counterparts such as Dart or Java.

Additionally, when compared with other programming languages, it has fewer in-built libraries or frameworks, contributing as another reason for it being lightweight. However, this brings a drawback in that we need to incorporate external libraries and frameworks.

Is JavaScript Compiled or Interpreted or both ?

JavaScript is both compiled and interpreted. In the earlier versions of JavaScript, it used only the interpreter that executed code line by line and shows the result immediately. But with time the performance became an issue as interpretation is quite slow. Therefore, in the newer versions of JS, probably after the V8, the JIT compiler was also incorporated to optimize the execution and display the result more quickly. This JIT compiler generates a bytecode that is relatively easier to code. This bytecode is a set of highly optimized instructions.
The V8 engine initially uses an interpreter, to interpret the code. On further executions, the V8 engine finds patterns such as frequently executed functions, and frequently used variables, and compiles them to improve performance.

JavaScript is best known for web page development but it is also used in a variety of non-browser environments. You can learn JavaScript from the ground up by following this JavaScript Tutorial and JavaScript Examples.

What is the difference between JavaScript and Java?

JavaScript and Java are distinct languages with different purposes. Java is an object-oriented programming language for building standalone applications, while JavaScript is primarily used to enhance web pages with interactivity and dynamic content.

Variables and Datatypes in JavaScript

Variables and data types are foundational concepts in programming, serving as the building blocks for storing and manipulating information within a program.

Variables

In JavaScript, variables are used to store and manage data. They are created using the var, let, or const keyword.

1. var Keyword

The var keyword is used to declare a variable. It has a function-scoped or globally-scoped behavior.

```
var x = 10;
```

Example: In this example, we will declare variables using var.

```
1  var a = "Hello World"
2  var b = 10;
3  var c = 15;
4  var d = b + c;
5
6  console.log(a);
7  console.log(b);
8  console.log(c);
9  console.log(d);
10
```

Output:

```
Hello World
10
15
25
```

2. let Keyword

The let keyword is a block-scoped variables. It's commonly used for variables that may change their value.

```
let y = "Hello";
```

Example: In this example, we will declare variables using let.

```
let a = "Hello World"
let b = "I am software developer";
let c = " 21";
let d = b + c;

console.log(a);
console.log(b);
console.log(c);
console.log(d);

```

Output:

```
Hello World
I am software developer
 21
I am software developer 21
```

3.const Keyword

The const keyword declares variables that cannot be reassigned. It's block-scoped as well.

```
const PI = 3.14;
```

Example: In this example, we will declare the variable using the const keyword.

```
const a = "Hello World"
console.log(a);

const b = 121;
console.log(b);

const c = "11";
console.log(c);
// Can not change a value for a constant
// c = "new"
// console.log(c) will show error

```

Output:

```
Hello World
121
11
```

Data Types

JavaScript is a dynamically typed (also called loosely typed) scripting language. In JavaScript, variables can receive different data types over time. The latest ECMAScript standard defines eight data types Out of which seven data types are Primitive (predefined) and one complex or Non-Primitive.

Primitive Data Types

The predefined data types provided by JavaScript language are known as primitive data types. Primitive data types are also known as in-built data types.

1. **Number**: JavaScript numbers are always stored in double-precision 64-bit binary format IEEE 754. Unlike other programming languages, you don't need int, float, etc. to declare different numeric values.
2. **String**: JavaScript Strings are similar to sentences. They are made up of a list of characters, which is essentially just an "array of characters, like "Hello World" etc.
3. **Boolean**: Represent a logical entity and can have two values: true or false.
4. **Null**: This type has only one value that is null.
5. **Undefined**: A variable that has not been assigned a value is undefined.
6. **Symbol**: Symbols return unique identifiers that can be used to add unique property keys to an object that won't collide with keys of any other code that might add to the object.

Non-Primitive Data Types

The data types that are derived from primitive data types of the JavaScript language are known as non-primitive data types. It is also known as derived data types or reference data types.

- **Object**: It is the most important data type and forms the building blocks for modern JavaScript. We will learn about these data types in detail in further articles.

JavaScript Operators

JavaScript Operators are symbols used to perform specific mathematical, comparison, assignment, and logical computations on operands. They are fundamental elements in JavaScript programming, allowing developers to manipulate data and control program flow efficiently. Understanding the different types of operators and how they work is important for writing effective and optimized JavaScript code.

There are different types of JavaScript operators:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- Bitwise Operators
- Ternary Operators

Arithmetic Operators are used to perform arithmetic on numbers

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

JavaScript Assignment Operators Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

JavaScript Comparison Operators are used to test for true or false

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that x = 5, the table below explains the comparison operators:

Operator	Description	Comparing	Returns
==	equal to	x == 8	false
		x == 5	true
		x == "5"	true
===	equal value and equal type	x === 5	true
		x === "5"	false
!=	not equal	x != 8	true
!==	not equal value or not equal type	x !== 5	false
		x !== "5"	true
		x !== 8	true
>	greater than	x > 8	false
<	less than	x < 8	true
>=	greater than or equal to	x >= 8	false
<=	less than or equal to	x <= 8	true

Logical Operators
Logical operators are used to determine the logic between variables or values.

Given that x = 6 and y = 3, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true

JavaScript Bitwise Operators
Bit operators work on 32 bits numbers.

Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

OPERATOR NAME	USAGE	DESCRIPTION
Bitwise AND(&)	a & b	Returns true if both operands are true
Bitwise OR(|)	a | b	Returns true even if one operand is true
Biwise XOR(^)	a ^ b	Returns true if both operands are different
Bitwise NOT(~)	~ a	Flips the value of the operand
Bitwise Left Shift(<<)	a << b	Shifts the bit toward the left
Bitwise Right Shift(>>)	a >> b	Shifts the bit towards the right
Zero Fill Right Shift(>>>)	a >>> b	Shifts the bit towards the right but adds 0 from left

Conditional (Ternary) Operator
JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

```
variablename = (condition) ? value1:value2
```

JavaScript Arrays

An array in JavaScript is a data structure used to store multiple values in a single variable. It can hold various data types and allows for dynamic resizing. Elements are accessed by their index, starting from 0.

You have two ways to create JavaScript Arrays: using the Array constructor or the shorthand array literal syntax, which is just square brackets. Arrays are flexible in size, so they can grow or shrink as you add or remove elements.

Basic Terminologies of JavaScript Array

- Array: A data structure in JavaScript that allows you to store multiple values in a single variable.
- Array Element: Each value within an array is called an element. Elements are accessed by their index.
- Array Index: A numeric representation that indicates the position of an element in the array. JavaScript arrays are zero-indexed, meaning the first element is at index 0.
- Array Length: The number of elements in an array. It can be retrieved using the length property.
- Declaration of an Array
- There are basically two ways to declare an array i.e. Array Literal and Array Constructor.

1. Creating an Array using Array Literal
Creating an array using array literal involves using square brackets [] to define and initialize the array. This method is concise and widely preferred for its simplicity.

Syntax:

```
let arrayName = [value1, value2, ...];
```

Example:

```
// Creating an Empty Array
let names = [];
console.log(names);

// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "Javascript", "React"];

console.log(courses);

```

Output:

```
[]
[ 'HTML', 'CSS', 'Javascript', 'React' ]
```

2. Creating an Array using JavaScript new Keyword (Array Constructor)

The "Array Constructor" refers to a method of creating arrays by invoking the Array constructor function. This approach allows for dynamic initialization and can be used to create arrays with a specified length or elements.

Syntax:

```
let arrayName = new Array();
```

Example:

```
// Creating and Initializing an array with values
let courses = new Array("HTML", "CSS", "Javascript", "React");
console.log(courses);

```

Output:

```
[ 'HTML', 'CSS', 'Javascript', 'React' ]
```

Basic Operations on JavaScript Arrays

1. Accessing Elements of an Array

Any element in the array can be accessed using the index number. The index in the arrays starts with 0.

Example:

```
// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "Javascript", "React"];

// Accessing Array Elements
console.log(courses[0]);
console.log(courses[1]);
console.log(courses[3]);

```

Output:

```
HTML
CSS
React
```

2. Accessing the First Element of an Array

The array indexing starts from 0, so we can access first element of array using the index number.

Example:

```
// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "JavaScript", "React"];

// Accessing First Array Elements
let firstItem = courses[0];

console.log("First Item: ", firstItem);

```

Output:

```
First Item:  HTML
```

3. Accessing the Last Element of an Array

We can access the last array element using [array.length – 1] index number.

Example:

```
// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "JavaScript", "React"];

// Accessing Last Array Elements
let lastItem = courses[courses.length - 1];

console.log("Last Item: ", lastItem);

```

Output:

```
Last Item:  React
```

4. Modifying the Array Elements

Elements in an array can be modified by assigning a new value to their corresponding index.

Example:

```
// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "Javascript", "React"];
console.log(courses);

courses[1]= "Bootstrap";
console.log(courses);

```

Output:

```
[ 'HTML', 'CSS', 'Javascript', 'React' ]
[ 'HTML', 'Bootstrap', 'Javascript', 'React' ]
```

5. Adding Elements to the Array
Elements can be added to the array using methods like push() and unshift().

The push() method add the element to the end of the array.
The unshift() method add the element to the starting of the array.

Example:

```
// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "Javascript", "React"];

// Add Element to the end of Array
courses.push("Node.js");

// Add Element to the beginning
courses.unshift("Web Development");

console.log(courses);

```

Output:

```
[ 'Web Development', 'HTML', 'CSS', 'Javascript', 'React', 'Node.js' ]
```

6. Removing Elements from an Array

To remove the elements from an array we have different methods like pop(), shift(), or splice().

The pop() method removes an element from the last index of the array.
The shift() method removes the element from the first index of the array.
The splice() method removes or replaces the element from the array.

Example:

```
// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "Javascript", "React",
"Node.js"];
console.log("Original Array: " + courses);

// Removes and returns the last element
let lastElement = courses.pop();
console.log("After Removing the last elements: " + courses);

// Removes and returns the first element
let firstElement = courses.shift();
console.log("After Removing the First elements: " + courses);

// Removes 2 elements starting from index 1
courses.splice(1, 2);
console.log("After Removing 2 elements starting from index 1: "
+ courses);

```

Output:

```
Original Array: HTML,CSS,Javascript,React,Node.js
After Removing the last elements: HTML,CSS,Javascript,React
After Removing the First elements: CSS,Javascript,React
After Removing 2 elements starting from index 1: CSS
```

7. Array Length

We can get the length of the array using the array length property.

Example:

```
// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "Javascript", "React", "Node.js"];

let len = courses.length;

console.log("Array Length: " + len);

```

Output:

```
Array Length: 5
```

8. Increase and Decrease the Array Length

We can increase and decrease the array length using the JavaScript length property.

Example:

```
// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "Javascript", "React", "Node.js"];

// Increase the array length to 7
courses.length = 7;

console.log("Array After Increase the Length: ", courses);

// Decrease the array length to 2
courses.length = 2;
console.log("Array After Decrease the Length: ", courses)

```

Output:

```
Array After Increase the Length:  [ 'HTML', 'CSS', 'Javascript', 'React',
'Node.js', <2 empty items> ]
Array After Decrease the Length:  [ 'HTML', 'CSS' ]
```

9. Iterating Through Array Elements

We can iterate array and access array elements using for loop and forEach loop.

Example: It is an example of for loop.

```
1  // Creating an Array and Initializing with Values
2  let courses = ["HTML", "CSS", "JavaScript", "React"];
3
4  // Iterating through for loop
5  for (let i = 0; i < courses.length; i++) {
6      console.log(courses[i])
7  }
8
9
```

Output:

```
HTML
CSS
JavaScript
React
```

Example: It is the example of Array.forEach() loop.

```
// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "JavaScript", "React"];

// Iterating through forEach loop
courses.forEach(function myfunc(elements) {
    console.log(elements);
});
```

Output:

```
HTML
CSS
JavaScript
React
```

10. Array Concatenation

Combine two or more arrays using the concat() method. It returns new array containing joined arrays elements.

Example:

```
// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "JavaScript", "React"];
let otherCourses = ["Node.js", "Expess.js"];

// Concatenate both arrays
let concateArray = courses.concat(otherCourses);

console.log("Concatenated Array: ", concateArray);
```

Output:

```
Concatenated Array:  [ 'HTML', 'CSS', 'JavaScript', 'React', 'Node.js',
'Expess.js' ]
```

11. Conversion of an Array to String

We have a builtin method toString() to converts an array to a string.

Example:

```
// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "JavaScript", "React"];

// Convert array ot String
console.log(courses.toString());

```

Output:

```
HTML,CSS,JavaScript,React
```

12. Check the Type of an Arrays

The JavaScript typeof operator is used ot check the type of an array. It returns “object” for arrays.

Example:

```
// Creating an Array and Initializing with Values
let courses = ["HTML", "CSS", "JavaScript", "React"];

// Check type of array
console.log(typeof courses);


```

Output:

```
object
```