Java Script

JavaScript is a ***high-level, object oriented***, **interpreted programming** language primarily used for creating **interactive and dynamic web content**. It is widely used for adding functionality, interactivity, and behaviour to websites.

Why use JavaScript?

JavaScript can be embedded directly into HTML code using **<script> tags** or included as an external file. ***It provides a wide range of features and capabilities, including***

- **Manipulating HTML elements:** JavaScript allows you to access and modify the content, structure, and styling of HTML elements on a web page. You can dynamically create, delete, or modify elements, change their attributes, apply CSS styles, and handle events such as mouse clicks or keyboard inputs.

- **Interacting with the browser:** JavaScript provides APIs (Application Programming Interfaces) that enable communication with the web browser. This allows you to control browser behaviour, manipulate the browser history, display alerts or prompts, and interact with cookies and local storage.

- **Handling user input:** JavaScript can capture and process user input, such as form submissions, button clicks, or keyboard events. You can validate and process user input, perform calculations, and update the web page accordingly.

- **Making HTTP requests:** JavaScript enables you to make asynchronous HTTP requests to retrieve data from servers using technologies like AJAX (Asynchronous JavaScript and XML). This allows you to fetch data in the background and update parts of a web page without requiring a full refresh.

- **Implementing logic and algorithms:** JavaScript supports various programming constructs like loops, conditional statements, functions, and objects. This allows you to implement complex logic and algorithms, perform calculations, manipulate data, and create reusable code.

- **Working with data:** JavaScript provides different data types, including numbers, strings, arrays, objects, and more. It supports operations like arithmetic, string manipulation, and array manipulation. Additionally, JavaScript supports JSON (JavaScript Object Notation), a popular data interchange format.

- **Building interactive web applications**: JavaScript is the foundation for many modern web frameworks and libraries, such as **React,** Angular, and Vue.js. These frameworks simplify the development of complex web applications by providing abstractions and tools for building user interfaces, managing state, and handling data flow.

JavaScript plays a crucial role in modern web development, enabling developers to create engaging, interactive, and dynamic web experiences for users.

e.g.

```
        <html>
<body>
        principle <input id="principle" type="text"><br>
        rate <input id="rate"  type="text"><br>
        time <input id="time"  type="text"><br>
        <button onclick="si()">simple interest</button><br>
        simple interest<p id="para1"></p>

        <script>
        function si() {
                var p = document.getElementById("principle");
                var r = document.getElementById("rate");
                var t = document.getElementById("time");

                const s_interest = (p.value * r.value * t.value )/ 100 ;
                var para1 = document.getElementById("para1");
                para1.innerHTML = s_interest;


        }
        </script>

</body>
</html>
```

BRIEF HISTORY OF JS

JavaScript was created by Brendan Eich while he was working at Netscape Communications Corporation in 1995. The language was developed in a very short period, and its initial name was "Mocha." Later, it was briefly named "LiveScript" before settling on the name "JavaScript."

brief timeline of significant events in the history of JavaScript:

- **1995:** JavaScript was introduced in Netscape Navigator 2.0 as a scripting language to enable client-side interactivity in web browsers.

- **1996:** Microsoft adopted a similar scripting language for its Internet Explorer browser, naming it JScript. This led to a certain degree of fragmentation in the early days of web development, with developers having to account for browser-specific differences.

- **1997:** JavaScript was submitted to the **Ecma International standards organization**, and the standardization process began. The standardized version was named ECMAScript, with the first edition published in June 1997.

- 1999: **ECMAScript 3** was released with various enhancements, and it became widely adopted. This version formed the basis for JavaScript in web browsers for many years.

- **2005:** The term "Ajax" (Asynchronous JavaScript and XML) was coined by Jesse James Garrett, describing a technique using JavaScript to create more dynamic and responsive web applications by making asynchronous requests to the server.

- **2009:** ECMAScript 5 was released, bringing several new features and improvements to the language. This version added support for strict mode, JSON (JavaScript Object Notation) parsing, and more.

- **2015:** ECMAScript 6 (also known as ECMAScript 2015 or ES6) was a major update to the language, introducing significant enhancements such as arrow functions, classes, template literals, and the let and const keywords.

- **Subsequent Years:** ECMAScript continued to evolve with regular updates. ECMAScript 2016, 2017, 2018, and so on, introduced additional features and improvements to the language.

- **Present:** JavaScript is an integral part of web development, used on both the client and server sides. Modern web development frameworks and libraries, such as React, Angular, and Vue.js, leverage JavaScript to create powerful and interactive user interfaces.

JavaScript's rapid evolution and widespread adoption have made it a fundamental technology for building dynamic and interactive web applications. The development community actively contributes to its growth through proposals and discussions in the ECMAScript standards process.

JAVASCRIPT ENGINES

JavaScript is not understandable by computer but the only browser understands JavaScript. So, we need a program to convert our JavaScript program into computer-understandable language. A JavaScript engine is a computer program that executes JavaScript code and converts it into computer understandable language.

List of JavaScript Engines:

Browser	Name of Javascript Engine
Google Chrome	V8
Edge (Internet Explorer)	Chakra
Mozilla Firefox	Spider Monkey
Safari	Javascript Core Webkit

EXECUTION CONTEXT:

In JavaScript, an execution context is an abstract concept that helps to manage the execution of code. It represents the environment in which a piece of code is executed, including variables, functions, and other resources available at a particular time during the program's execution. Understanding execution contexts is crucial for comprehending how JavaScript code is processed.

Global Execution Context:

- The global execution context is the outermost context and is created when the script is executed.
- It encompasses the entire script and includes global variables and functions.
- There is only one global execution context in a JavaScript program.

Note:

JavaScript engine creates the global execution context before it starts to execute any code. Variables, and function that is not inside any function.

The global execution context just like any other execution context, except that it gets created by default. It is associated with Global Object. Which means a **window object**. Ex. **this.window**, **name = = window.name**

```
e.g. <html>
<body>
<script>
   var i = 5 // global context
   function check1(){
      var a = 5 // function specific context
   }
   function check2(){
      var b = 1 // function specific context
      document.write("a = " + a )
   }

   document.write("i = " + this.i)
   document.write("i = " + this.window.i )
   this.window.document.write("i = " + this.i )
   document.write("a  = " + a )

   // check2()
</script>
</body>
</html>
```

CHARACTER SET

JavaScript uses the Unicode character set. Unicode is a standardized character encoding that assigns a unique number (code point) to each character for every writing system, language, and symbol in the world. It aims to cover all possible characters across all languages and scripts, making it a comprehensive character set.

In JavaScript, strings are sequences of Unicode characters. Unicode supports a vast range of characters, including letters, numbers, punctuation, symbols, and special characters from various languages and scripts.

JavaScript supports Unicode escape sequences in strings using the \u prefix followed by four hexadecimal digits. For example:

var ch = "\u0041"; // represents the copyright symbol A

Keywords

Keywords in JavaScript are reserved words that have special meanings and play specific roles in the language's syntax and functionality. These words cannot be used as identifiers (variable names, function names, etc.) because they are already predefined and serve specific purposes within the JavaScript language.

These are the reserve words whose meaning is predefine. In JavaScript you cannot use these

abstract	arguments	await*	boolean	break	byte	case	catch	char	class*
const	continue	debugger	default	delete	do	double	else	enum*	Eval
export*	extends*	false	final	finally	float	for	function	goto	If
implements	import*	in	instanceof	int	interface	let*	long	native	New
null	package	private	protected	public	return	short	static	super*	Switch
synchronized	this	throw	throws	transient	true	try	typeof	var	void
volatile	while	with	Yield						

Note: JavaScript is a **case-sensitive language**.

Identifier

An identifier is a name given to a variable, function, class, object, or any other user-defined item in the code. Identifiers serve as labels or names that help developers refer to and manipulate different elements within a program. They play a crucial role in making code more readable, understandable, and maintainable.

- There are some rules to create identifier: -
  - It can be any combination of capital letters, small letters, currency symbol and underscore.
  - It cannot be begun with digits.
  - Identifiers cannot be the same as reserved keywords or language-defined constructs in the programming language.
  - Choosing meaningful and descriptive names for identifiers is important for code readability.
  - Consistent naming conventions for identifiers help maintain a clean and organized codebase.
    - Common conventions include using camelCase for variables and function names (e.g., myVariable, **calculateSum**) and PascalCase for class names (e.g., **MyClass)**.

STATEMENT

In JavaScript, a statement is a complete and executable unit of code. Statements are the building blocks of JavaScript programs and are typically terminated by a semicolon (;). A JavaScript program consists of a sequence of statements that are executed in order, unless control flow structures (such as conditionals or loops) alter the normal flow.

Declaration statement

As we know variables are containers for storing values and the declaration statement allow us to create variable.

- There the four ways to declare variable.
- **Var (variant) keyword**

  In JavaScript, the var keyword is used to declare variables. Variables declared with var have function scope or global scope, but they do not have block scope. This means that a variable declared with var is accessible throughout the entire function or script in which it is declared, regardless of the block (if statement, loop, etc.) in which it appears.

e.g.

```
<html>

<body>
 <script>
  function example() {
   var x = 10;
   if (true) {
    var y = 20;
    console.log(x); // accessible, prints 10
   }
   console.log(y); // accessible, prints 20
  }

  example();
  console.log(x); // Error: x is not defined outside the function
  console.log(y); // Error: y is not defined outside the block

 </script>
</body>

</html>
```

let keyword

In JavaScript, the let keyword is used to declare variables. Variables declared with let have block scope, meaning they are limited to the block, statement, or expression where they are defined. Unlike variables declared with var, let does not have function scope and is not hoisted to the top of the block during the compilation phase.

```
let x = 50 ;
```

e.g.

```
<html>

<body>
 <script>
  function example() {
   let x = 10;
   if (true) {
    let y = 20;
    console.log(x); // accessible, prints 10
    console.log(y); // accessible, prints 20
   }
    console.log(y); // Error: y is not defined outside the block
  }
```

```
  example();
  // console.log(x); // Error: x is not defined outside the function

 </script>
</body>

</html>
```

CONST KEYWORD

In JavaScript, the const keyword is used to declare constants, which are variables whose values cannot be reassigned after their initial assignment. Constants provide a way to create variables that should remain unchanged throughout the execution of a program. It introduces block scope like let and var, and it is also not hoisted to the top of the block.

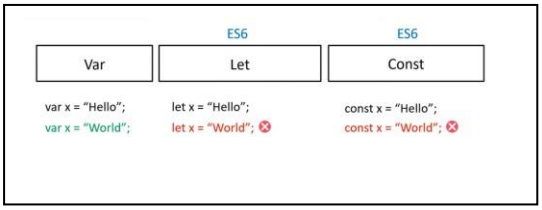
- It allow us to create **read only variable.**

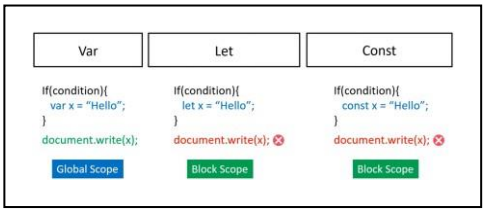
e.g.

```
<html>

<body>
 <script>
  const pi = 3.14;
  // pi = 3.14159; // Error: Assignment to constant variable

  if (true) {
    const x = 10;
    console.log(x); // accessible, prints 10
    // x = 20; // Error: Assignment to constant variable
  }

  // console.log(x); // Error: x is not defined outside the block

 </script>
</body>

</html>
```

Difference between var & let

- The var keyword was introduced with JavaScript whereas The let keyword was added in ES6 (ES 2015) version of JavaScript.
- var has global scope whereas let is limited to block scope.
- var can be declared globally and can be accessed globally where let can be declared globally but cannot be accessed globally.





- Variable declared with var keyword can be re-declared and updated in the same scope.

```
<script>
   var a = 10 ;
   var a = 20 ;
</script>
```

```
<script>
   let a = 10 ;
   let a = 20 ;
</script> //error
```

Undeclared

An undeclared variable in JavaScript is a variable that has been used in the code without being explicitly declared using var, let, or const.

e.g.

```
<html>
<body>
 <script>
  x = 10;
   function example() {
    if (true) {
     x = 10;
     console.log(x); // accessible, prints 10
    }
    console.log(y); // accessible, prints 20
  }
  example();
  console.log(x); // Error: x is not defined outside the function
  console.log(y); // Error: y is not defined outside the block
 </script>
</body>
</html>
```

What happed when we declared the variable

in JavaScript, it goes through a process known as variable instantiation. The JavaScript engine allocates memory for the variable and sets its initial value to **undefined.** This process occurs during the variable instantiation phase, which is part of the creation of the variable's execution context.

e.g.

```
var a; // Memory is allocated for 'a' and initialized with 'undefined'
```

1. **memory Allocation:** The JavaScript engine allocates memory space for the variable a in the current execution context (e.g., within a function or at the global level).
2. **Initialization with undefined:** The engine sets the initial value of the variable a to undefined. This is the default value assigned to variables during the variable instantiation phase.
3. **Variable Instantiation:**The process of declaring a variable with var involves both memory allocation and initialization. This combination is referred to as variable instantiation.

STRICT MODE

Strict mode is a feature in JavaScript that was introduced with ECMAScript 5 (ES5) to enhance the language by catching common coding errors and preventing the use of certain "unsafe" features. When strict mode is enabled, the JavaScript interpreter becomes more strict in its parsing and error handling.

```
<script>
        "use strict";
        var amt;
       ant = 10 ; // Throws a ReferenceError.
       console.log(amt);
</script>
```

Akhilesh Kumar Gupta

(Technical Training specialist)
TechCarrel LLP