JAVASCRIPT

JavaScript, created by Brendan Eich in 1995, is one of the most widely used web development languages. It was designed to build dynamic web pages at first. A script is a JS program that may be added to the HTML of any web page. When the page loads, these scripts execute automatically. A language that was originally designed to build dynamic web pages may now be run on the server and on almost any device that has the JavaScript Engine installed.
After HTML and CSS, JavaScript is the third biggest web technology. JavaScript is a scripting language that may be used to construct online and mobile apps, web servers, games, and more. JavaScript is an object-oriented programming language that is used to generate websites and applications. It was created with the intention of being used in a browser. Even today, the server-side version of JavaScript known as Node.js may be used to create online and mobile apps, real-time applications, online streaming applications, and videogames. Javascript frameworks, often known as inbuilt libraries, may be used to construct desktop and mobile programs. Developers may save a lot of time on monotonous programming jobs by using these code libraries, allowing them to focus on the production work of development.

1. What are the different data types present in javascript?

To know the type of a JavaScript variable, we can use the typeof operator.

1. Primitive types

String - It represents a series of characters and is written with quotes. A string can be represented using a single or a double quote.

Example :

```
var str = "Vivek Singh Bisht"; //using double quotes
var str2 = 'John Doe'; //using single quotes
```

- Number - It represents a number and can be written with or without decimals.

Example :

```
var x = 3; //without decimal
var y = 3.6; //with decimal
```

- BigInt - This data type is used to store numbers which are above the limitation of the Number data type. It can store large integers and is represented by adding “n” to an integer literal.

Example :

```
var bigInteger =  234567890123456789012345678901234567890;
```

- Boolean - It represents a logical entity and can have only two values : true or false. Booleans are generally used for conditional testing.

Example :

```
var a = 2;
var b =  3;
var c =  2;
(a == b) // returns false
(a == c) //returns true
```

- Undefined - When a variable is declared but not assigned, it has the value of undefined and it’s type is also undefined.

Example :

```
var x; // value of x is undefined
var y = undefined; // we can also set the value of a variable as undefined
```

- Null - It represents a non-existent or a invalid value.

Example :

```
var z = null;
```

- Symbol - It is a new data type introduced in the ES6 version of javascript. It is used to store an anonymous and unique value.

Example :

```
var symbol1 = Symbol('symbol');
```

- typeof of primitive types :

```
typeof "John Doe" // Returns "string"

typeof 3.14 // Returns "number"

typeof true // Returns "boolean"

typeof 234567890123456789012345678901234567890n // Returns bigint

typeof undefined // Returns "undefined"

typeof null // Returns "object" (kind of a bug in JavaScript)

typeof Symbol('symbol') // Returns Symbol
```

2. Non-primitive types

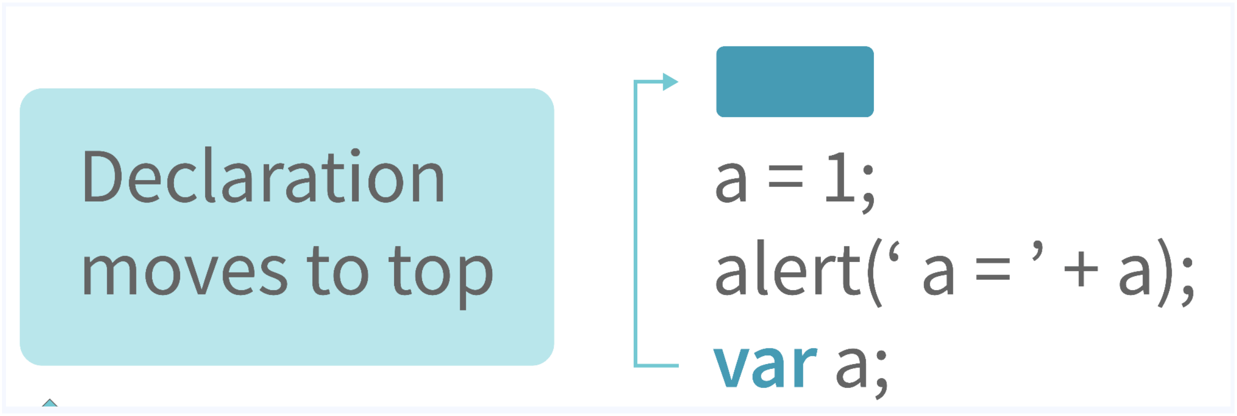
- Primitive data types can store only a single value. To store multiple and complex values, non-primitive data types are used.

- Object - Used to store collection of data.
- Example:

```
// Collection of data in key-value pairs

var obj1 = {

  x:  43,

  y:  "Hello world!",

  z: function(){

    return this.x;

  }

}

// Collection of data as an ordered list

var array1 = [5, "Hello", true, 4.1];
```

2. Explain Hoisting in javascript.

Hoisting is the default behaviour of javascript where all the variable and function declarations are moved on top.



This means that irrespective of where the variables and functions are declared, they are moved on top of the scope. The scope can be both local and global.

Example 1:

```
hoistedVariable = 3;

console.log(hoistedVariable); // outputs 3 even when the variable is declared after it is initialized

var hoistedVariable;
```

Example 2:

```
hoistedFunction();  // Outputs " Hello world! " even when the function is declared after calling

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

Example 3:

```
// Hoisting takes place in the local scope as well

function doSomething(){
  x = 33;
  console.log(x);
  var x;
}
```

doSomething(); // Outputs 33 since the local variable “x” is hoisted inside the local scope

Note - Variable initializations are not hoisted, only variable declarations are hoisted:

```
var x;

console.log(x); // Outputs "undefined" since the initialization of "x" is not hoisted

x = 23;
```

Note - To avoid hoisting, you can run javascript in strict mode by using “use strict” on top of the code:

```
"use strict";

x = 23; // Gives an error since 'x' is not declared

var x;
```

3. Why do we use the word “debugger” in javascript?

The debugger for the browser must be activated in order to debug the code. Built-in debuggers may be switched on and off, requiring the user to report faults. The remaining section of the code should stop execution before moving on to the next line while debugging.

4. Difference between “ == “ and “ === “ operators.

Both are comparison operators. The difference between both the operators is that “==” is used to compare values whereas, “ === “ is used to compare both values and types.

Example:

```
var x = 2;

var y = "2";

(x == y)  // Returns true since the value of both x and y is the same

(x === y) // Returns false since the typeof x is "number" and typeof y is "string
```

5. Difference between var and let keyword in javascript.

Some differences are

1. From the very beginning, the 'var' keyword was used in JavaScript programming whereas the keyword 'let' was just added in 2015.
2. The keyword 'Var' has a function scope. Anywhere in the function, the variable specified using var is accessible but in ‘let’ the scope of a

variable declared with the 'let' keyword is limited to the block in which it is declared. Let's start with a Block Scope.

3. In ECMAScript 2015, let and const are hoisted but not initialized. Referencing the variable in the block before the variable declaration results in a ReferenceError because the variable is in a "temporal dead zone" from the start of the block until the declaration is processed.