Topics to be covered: Introduction to JS, JS and ES, variable, variable declaration, keywords- var, let, const, data types.

Introduction to JavaScript

JavaScript is a cross-platform, object-oriented scripting language used to make webpages interactive (e.g., having complex animations, clickable buttons, popup menus, etc.).

JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements. Core JavaScript can be extended for a variety of purposes by supplementing it with additional objects.

- ***Client-side JavaScript*** extends the core language by supplying objects to control a browser and its Document Object Model (DOM).
- ***Server-side JavaScript*** extends the core language by supplying objects relevant to running JavaScript on a server.

JS is **dynamically typed** language. Dynamically-typed languages are those (like JavaScript) where the interpreter assigns variables a type at runtime based on the variable's value at the time.

JavaScript and the ECMAScript specification - JavaScript is standardized at Ecma International. The European association for standardizing information and communication systems (ECMA was formerly an acronym for the **European Computer Manufacturers Association**) to deliver a standardized, international programming language based on JavaScript. This standardized version of JavaScript, called ECMAScript, behaves the same way in all applications that support the standard. The ECMAScript standard is documented in the ECMA-262 specification.

JavaScript documentation versus the ECMAScript specification - The ECMAScript specification is a set of requirements for implementing ECMAScript. It is useful if you want to implement standards-compliant language features in your ECMAScript implementation or engine (such as SpiderMonkey in Firefox, or V8 in Chrome).

Note: The ECMAScript document is not intended to help script programmers. Use the JavaScript documentation for information when writing scripts.

Variable Declaration

JavaScript has three kinds of variable declarations.

var	Declares a variable, optionally initializing it to a value.
Let	Declares a block-scoped, local variable, optionally initializing it to a value
const	Declares a block-scoped, read-only named constant.

Variable name - The names of variables, called identifiers, conform to certain rules.

A JavaScript identifier usually starts with a letter, underscore (_), or dollar sign ($). Subsequent characters can also be digits (0 – 9). Because JavaScript is case sensitive, letters include the characters A through Z (uppercase) as well as a through z (lowercase).

Note: You can use most Unicode letters such as å and ü in identifiers.

Examples of legal variable names are Number_hits, temp99, $credit, and _name.

Declaring variables: You can declare a variable in two ways:

- With the keyword var. For example, var x = 42. This syntax can be used to declare both local and global variables, depending on the execution context.

- With the keyword const or let. For example, let y = 13. This syntax can be used to declare a block-scope local variable. (See Variable scope below.)

You can declare variables to unpack values using the destructuring assignment syntax.

For example, const { bar } = foo.

This will create a variable named bar and assign to it the value corresponding to the key of the same name from our object foo.

Variables should always be declared before they are used. JavaScript used to allow assigning to undeclared variables, which creates an undeclared global variable. This is an error in strict mode and should be avoided altogether.

In a statement like let x = 42, the let x part is called a **declaration**, and the = 42 part is called an **initializer.**

Note: *const* declarations always need an initializer

Variable Scope

A variable may belong to one of the following scopes:

- **Global scope:** The default scope for all code running in script mode.
- **Module scope:** The scope for code running in module mode.
- **Function scope:** The scope created with a function.

In addition, variables declared with let or const can belong to an additional scope:

- **Block scope:** The scope created with a pair of curly braces (a block).

When you declare a variable outside of any function, it is called a **global variable**, because it is available to any other code in the current document. When you declare a variable within a function, it is called a **local variable**, because it is available only within that function.

let and *const* declarations can also be scoped to the **block statement** that they are declared in.

Example:

```
if (Math.random() > 0.5) {
  const y = 5;
}
console.log(y); // ReferenceError: y is not defined
```

However, variables created with var are not block-scoped, but only local to the function (or global scope) that the block resides within.

Data types

The latest ECMAScript standard defines **eight** data types:

Primitive data types:

- **Boolean:** true and false.
- **null:** A special keyword denoting a null value. (Because JavaScript is case-sensitive, null is not the same as Null, NULL, or any other variant.)
- **undefined**: A top-level property whose value is not defined.
- **Number**: An integer or floating-point number. For example: 42 or 3.14159.
- **BigInt**: An integer with arbitrary precision. For example: 9007199254740992n.
- **String**: A sequence of characters that represent a text value. For example: "Howdy".
- **Symbol:** A data type whose instances are unique and immutable.

Non-Primitive data type:

- **Object:** In JavaScript, objects can be seen as a collection of properties. With the object literal syntax, a limited set of properties are initialized; then properties can be added and removed.

  Important points to remember about data types:

  - Number data type covers both integer and floating point number
  - String data type covers character also
  - Syntax for creating is given below

    const sym = Symbol('first');
  - Each symbol created is unique

    Example:

    ```
    const sym1 = Symbol("foo")
    const sym2 = Symbol("foo")

    if(sym1 === symb2)   // output is false
    ```
  - *new* keyword is not used while creating symbol.

Note: Students, try to solve the questions in following practice sheet on your own

PRACTICE SHEET I

Q. No.	Description
1.	Out of following which are valid variable names 2ndName, full-name, for, $f
2.	Can you declare multiple variables on one line
3.	Can you redeclare a variable with same name in JavaScript

4.	What will be the output of below code snippet
	`var text = 'outside';`
	`function logIt(){`
	`    console.log(text);`
	`    var text = 'inside';`
	`};`
	`logIt();`

5.	The scope describes the _____ and _____ of a variable. a. visibility and accessibility b. visibility and value c. accessibility and value d. accessibility and dependency
6.	State true false: let and const are block scoped
7.	What will be the output of the following code snippet? console.log (typeof NaN)
8.	What will be the output of the following code snippet? console.log (Math.max())
9.	What will be the output of the following code snippet? console.log (“123”+123)
10.	What will be the output of the following code snippet? console.log (`Number.NEGATIVE_INFINITY`)

11.	What will be the output of the following code snippet?
	var a = Number.NEGATIVE_INFINITY/-2;
	console.log(a)

12.	What is block in JavaScript?

13.	What will be the output of the following code snippet?
	var a = true + true + true * 3;
	console.log(a)

14.	What will be the output of the following code snippet?
	const sym = new Symbol(“First”);
	console.log(sym);

15.	When an operator’s value is NULL, the typeof returned by the unary operator is: • Boolean • Undefined • Object • Integer
16.	When the switch statement matches the expression with the given labels, how is the comparison done? • Both datatype and result of expression is compared • Only datatype of expression is compared

	• Only the value of expression is compared • None of the above
17.	Which of the following support **hoisting property** • var • let • const • None of the above
18.	In which of the following its mandatory to initialize the variable while declaration • var • let • const • None of the above
19.	What is output of following code snippet console.log(typeof 5987456n)
20.	What is output of following code snippet console.log(parseInt(‘123abc’))