JAVASCRIPT INTRODUCTION CLASS-1

❖ Why to use JavaScript?
Functionality or performed dynamical behaviour and interactive to users (User friendly).

❖ When was JavaScript is invented and by whom?
JS was invented by "Brendan Eich" in 1995. JS was developed for Netscape 2 and became ECMA (A6 Version).

❖ How JavaScript is executed?
JS Code may be executed in one of two runtime Environments. Two runtime Environments are:

1.) Browers runtime Environment

- JS code → Html →Browers runtime Environment.
- In Browser, JS Code Engine / V8 Engine can dynamically behave accordingly to Applications for users to run the JS Code.
- V8 Engine is responsible for parsing, Compiling and executing JS Code.
- As a part of this, it needs to manage memory, control the flow of the execution, optimize the code and interact with the environment either browsers run time or Node runtime.
- V8 Engine is done in 2 phases:
    1. Runtime Environment through Memory.
       Memory allocation is done by Heap memory(Pool of memory where var, object, ref are stored) and Stack memory(That tracks the code at any time i.e.; direct storing).
    2. Execution of code Phase.

2.) Node Run time Environment
Runs outside the Browser and mostly useful for the Applications i.e.; Server set (which connects directly to the data set). The path is taken from Operating System(OS) then from that Nodejs Environment is created.

JAVASCRIPT

- ➤ JavaScript is <mark>Single Synchronous Single Threaded Language</mark>.
- ➤ JavaScript is Cross-platform, light weight, interpreted, Compiled and Scripting Language.
- ➤ JavaScript is "High-Level Language" (Easy for Humans to understand for reading and writing the code easily).

SAMPLE CODE (Steps for Execution)

Task Queue

Microtask Queue

Call Stack

Event Loop → 1. Evaluate Script

2. Run a Task

3. Run all Microtasks

4. Rerender

PROCESS OF EXECUTION

Browser

JSE

CALL STACK(GEC)

Memory Phase

Execution Phase

Web API's(method (Predefinedcode))

Console ,Location, dom, set, fetch

(Asynchronous single threaded)

EVENT LOOP

TASK QUEUE

MICROTASK QUEUE