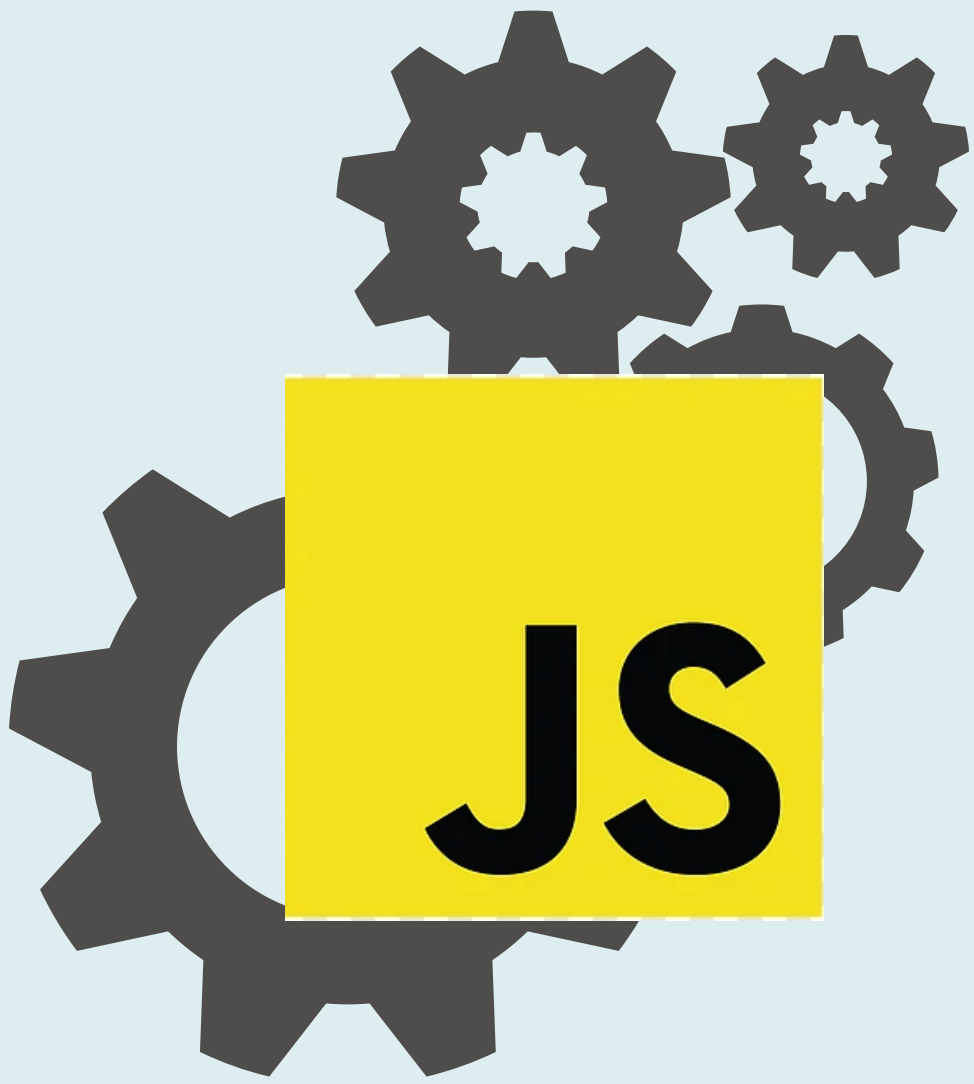
How JavaScript Engine Works

Let’s explore behind the scenes of JS Engine



@ritika-pethkar

From Code to Execution: The 3 Stages of JS Engine

ever wondered how your JS code actually runs behind the scenes?

The JavaScript Engine processes your code in 3 major phases:

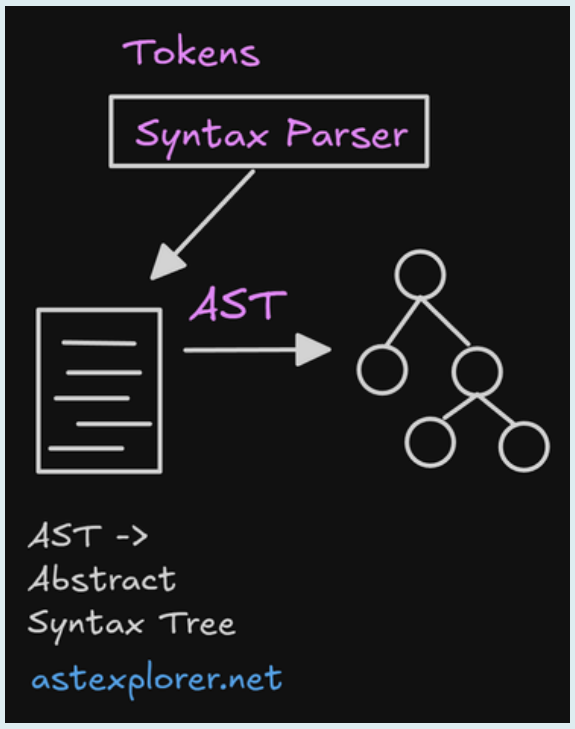
1. **Parsing –** Turns code into a structured format (AST)
2. **Compilation –** Converts it to bytecode using JIT
3. **Execution** – Runs the code using memory heap & call stack

Let’s break it down step by step 👉

@ritika-pethkar

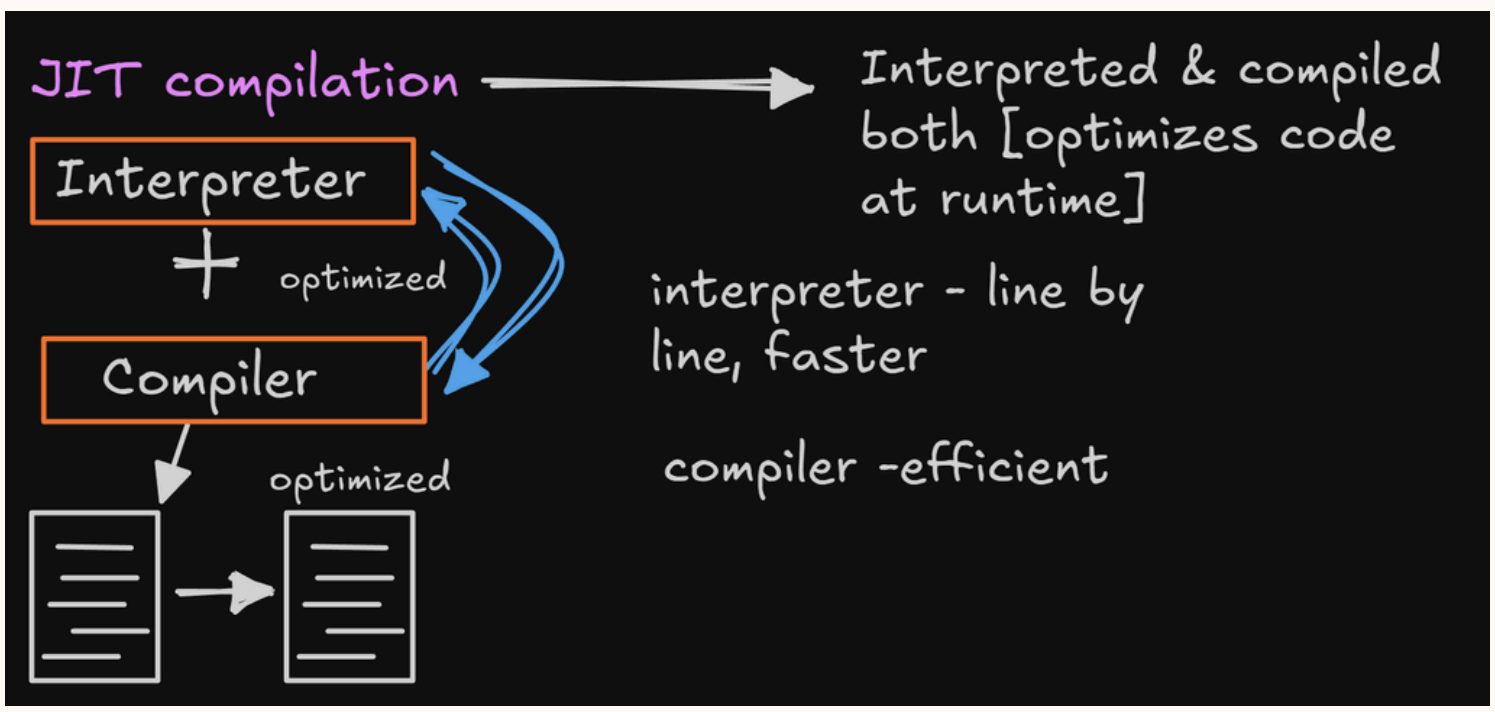
Parsing Phase

- The engine breaks your code into **tokens**
- These tokens are passed to the **Syntax Parser**
- The parser creates an **AST (Abstract Syntax Tree)** — a structured representation of your code
- **Useful Tool:** astexplorer.net to visualize ASTs
- **Goal:** Convert raw code into a format the engine can understand



Compilation (JIT)

- AST is compiled into bytecode using JIT **(Just-In-Time) compilation**
- JIT uses both:
  - 🚶 **Interpreter –** Executes code line-by-line for fast startup
  - ⚙️ **Compiler –** Optimizes the code for better long-term performance
- **Result:** Fast and optimized code execution!

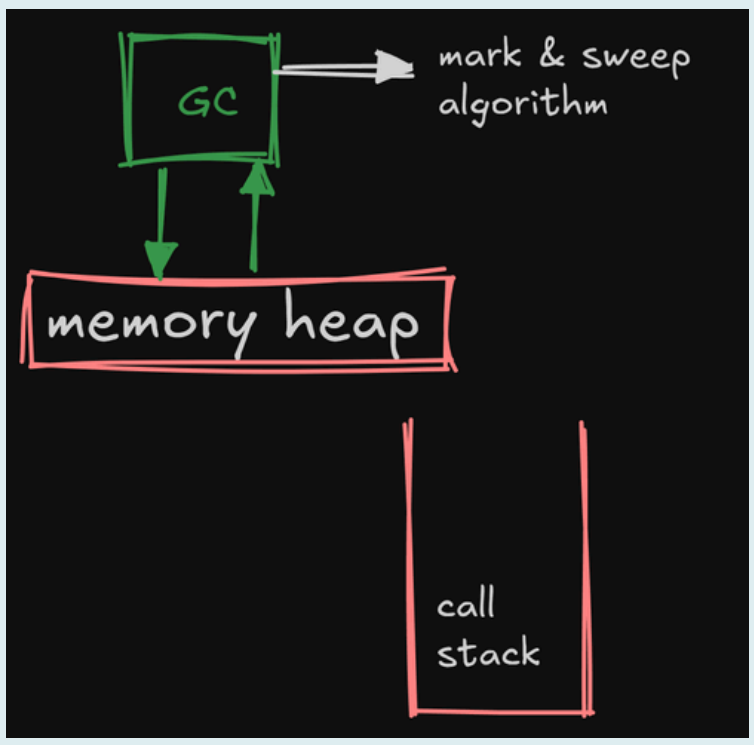


Execution

The engine now executes the bytecode using:

- 🧠 **Memory Heap –** Where variables, functions, and objects are stored.
- 📞 **Call Stack –** Keeps track of function calls and manages control flow.

This is where the actual output of your code is produced.



@ritika-pethkar

Memory Management (Garbage Collection)

- JS engine includes a **Garbage Collector (GC)**
- It automatically cleans up memory by removing data that’s no longer used
- Most engines use the Mark-and-Sweep algorithm

Why it matters: Keeps apps **memory-efficient and prevents leaks**

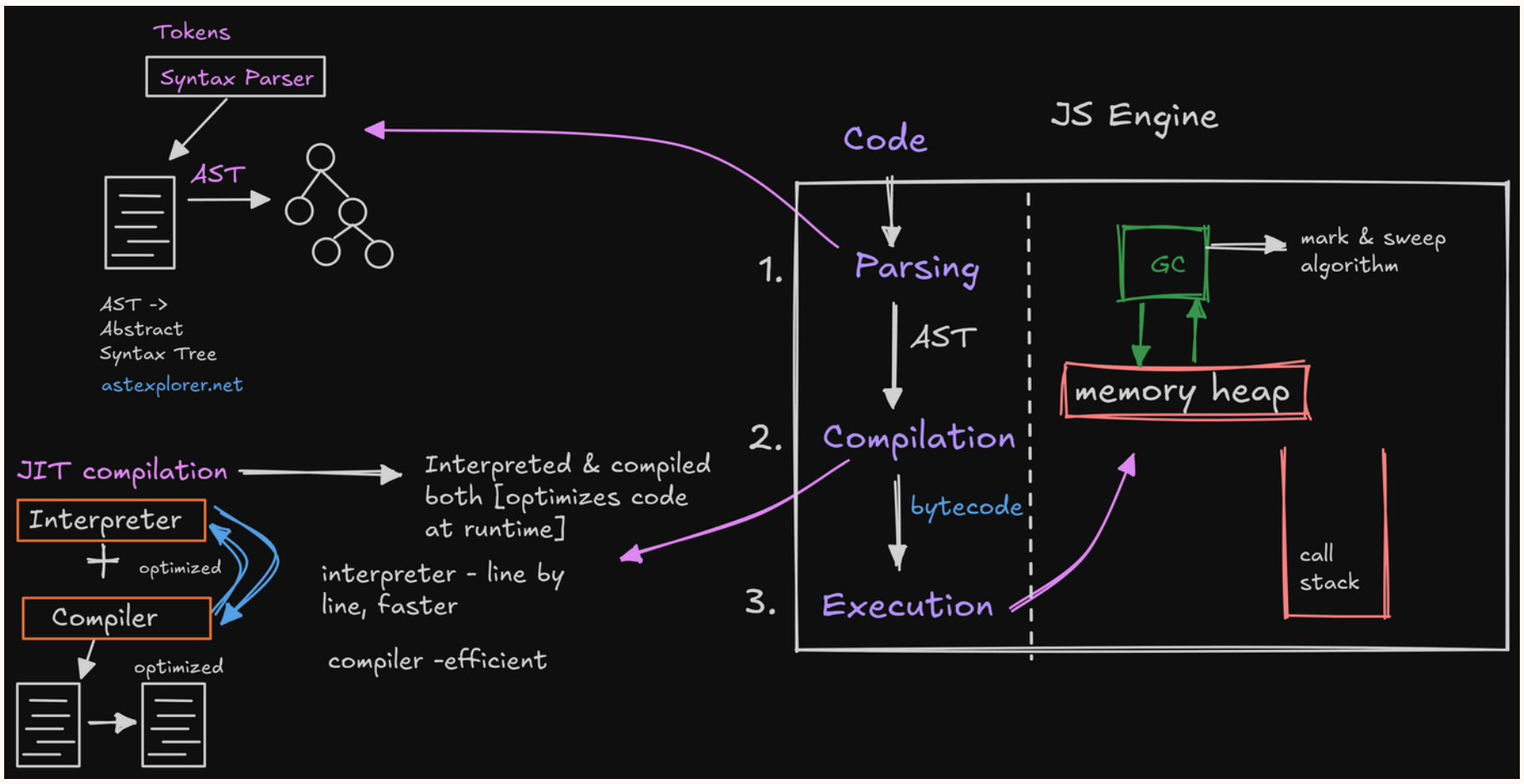
@ritika-pethkar

Recap – JS Engine in Action

✅ **Parsing** → Converts code to AST
✅ **Compilation** → Bytecode via JIT (interpreter + compiler)
✅ **Execution** → Managed by memory heap & call stack
✅ **GC** → Automatically handles memory cleanup

@ritika-pethkar

JS Engine Representation



@ritika-pethkar