






Episode 16: JS Engine Exposed – Google's V8 Architecture



JavaScript is Everywhere!

From **smartwatches** , **robots** , to **browsers**  — JavaScript runs on everything thanks to the **JavaScript Runtime Environment (JRE)** .


What is JRE (JavaScript Runtime Environment)?

 Think of it as a **container** filled with everything needed to run JS code:




-  **JS Engine** (core of JRE)
-  **Web APIs** to interact with the outside world
-  **Event Loop**
-  **Callback Queue**
-  **Microtask Queue**

 Browsers like Chrome  can run JS because they come with a built-in **JRE**.

ECMAScript & JS Engines



 **ECMAScript (ES)** is like the **constitution**  of JavaScript.

JS Engines follow ES rules. Some popular ones:

-  **V8** – Chrome, Edge (by Google)
-  **SpiderMonkey** – Firefox (created by Brendan Eich himself!)
-  **Chakra** – Older versions of Edge (now deprecated)

What is a JavaScript Engine?

A **JS Engine** is **not hardware**, it's a **program** written in **low-level languages (C/C++)** .


It takes in **high-level JS code** and converts it into **machine code**  .

How JavaScript Code Runs Inside the Engine

JS execution goes through **3 core stages**:

1. Parsing

Breaking code into understandable chunks for the engine.

 Example:

```
let a = 7;
```

Becomes tokens:



- `let, a, =, 7`

 Then, the **Syntax Parser** converts these tokens into an **AST (Abstract Syntax Tree)** 

 Use  [AST Explorer](#) to visualize it.

It's like a `package.json` for your line of JS code.

2. Compilation (JIT - Just-In-Time)





Combines benefits of interpreter  and compiler 

 **Old days:**

JS was only interpreted.


 **Now:**

JS is **JIT compiled**:





-  AST  Interpreter → creates **Bytecode**
-  **Compiler (TurboFan)** also works in parallel to optimize code
-  JS compiles **during execution**!

✓ Yes, **JavaScript *does* compile!** 

3. Execution

The moment JS code actually *runs* 

Needs:

-  **Memory Heap** – for storing variables, objects
 -  **Call Stack** – stack of function calls (from previous episodes)
 -  **Garbage Collector** – uses **Mark & Sweep**  to free up unused memory
-

V8 Architecture (Google's JS Engine)

Google's V8 is one of the **most powerful JS engines**, powering Chrome & Node.js 

 Components:

-  **Ignition** – Interpreter

- 🏭 **TurboFan** – Optimizing Compiler
- ✂️ **Orinoco** – Smart Garbage Collector

🔄 Flow:



🔧 Summary with Emojis:

Concept	Emoji	Meaning
JS Engine	⚙️	Core software that runs JS
Parsing	🔍	Tokenizing + AST generation
JIT Compilation	🏭	Compile + Interpret during runtime
Execution	💧	Running the bytecode
Memory Heap	🔄	Stores variables, objects, etc.
Call Stack	📞	Tracks function execution
Garbage Collection	✂️	Cleans unused memory (Mark & Sweep)
Ignition	🔍	Interpreter in V8
TurboFan	🚀	Optimizing compiler in V8
Orinoco	♻️	Garbage collector in V8

💡 Fun Fact:

Different companies build different JS engines to **optimize performance**, but all must follow ECMAScript 📄 standards!