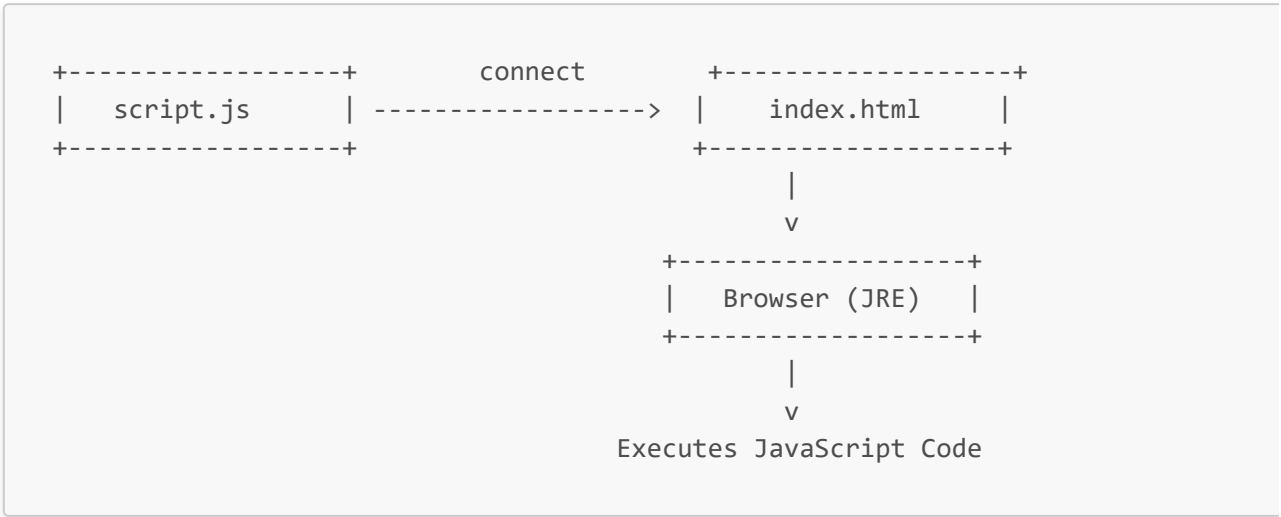
JavaScript: How It Actually Works

> This note is a comprehensive breakdown of JavaScript's working environment, engine, and Node.js evolution.

JavaScript Working Mechanism

1. JavaScript Runtime Environment (JRE)

When `script.js` is connected to `index.html` and opened in a browser, it runs using the **JavaScript Runtime Environment** provided by the browser.



2. Web APIs (provided by Browser)

- DOM
- EVENT
- setTimeout

- setInterval
- PROMISE
- fetch()
- navigator

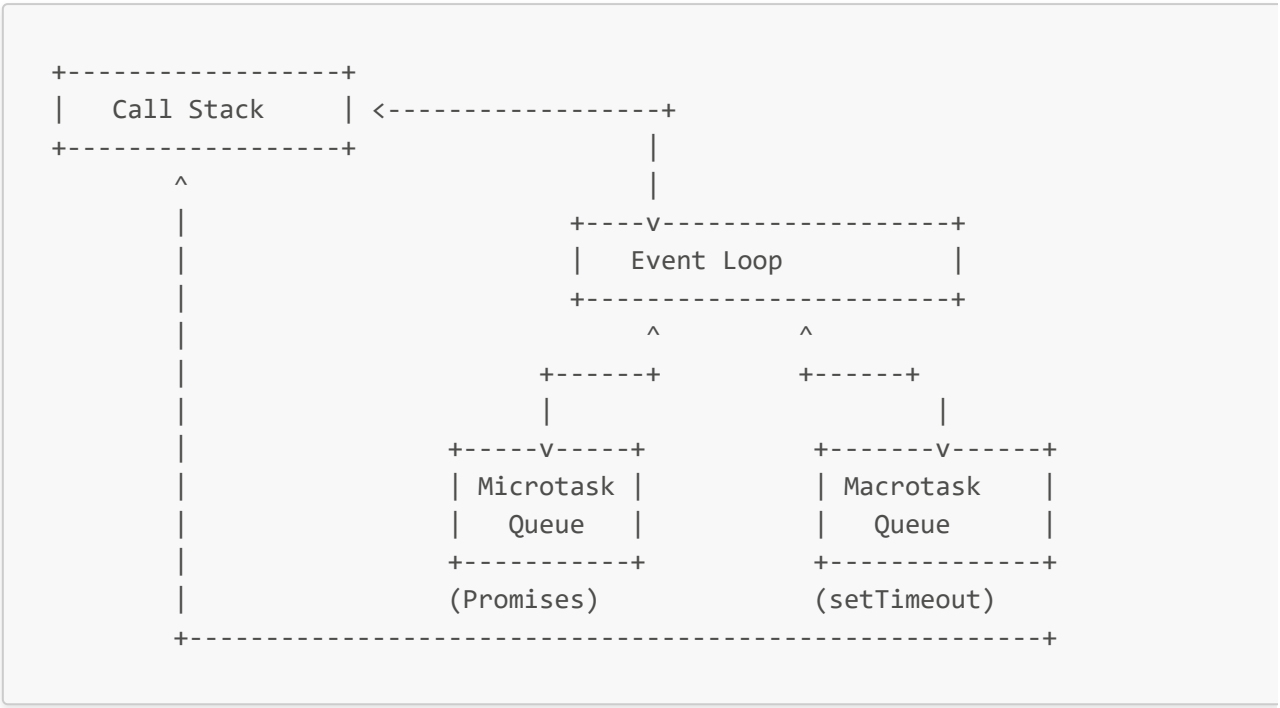
3. Storage APIs

- Local Storage
- Session Storage
- IndexedDB
- Cookies
- Cache

4. JavaScript Engine Components

- Parser – Checks syntax.
- Abstract Syntax Tree (AST) – Represents structure of code.
- Tokenization – Breaks code into tokens.
- Interpreter – Executes code line-by-line.
- JIT Compiler – Converts code to machine language.
- Garbage Collector – Frees memory.
- Call Stack – Executes and manages function calls (LIFO).
- Heap Memory – Allocates memory for objects.

5. Event Loop



6. Execution Flow

```
console.log("start");
let a = "Ravi";
setTimeout(() => console.log("Timeout"), 2000); // Async (Macrotask Queue)
Promise.resolve().then(() => console.log("Promise")); // Async (Microtask
Queue)
console.log(a);
console.log("end");
```

Execution Order:

1. Synchronous code runs first (`start` , variable declarations, `a` , `end`).
2. Microtasks (e.g., Promise).
3. Macrotasks (e.g., setTimeout after 2s).

Browser JS Engines

Browser	JavaScript Engine
Google Chrome	V8
Mozilla Firefox	Spider Monkey
Safari	JavaScriptCore
MS Edge	Chakra
Brave	V8

- If a browser has a JS Engine, it becomes a **JRE**.

Node.js Evolution Year-Wise

2009

- **Ryan Dahl** (Creator of Node.js).
- Took **Spider Monkey** (Mozilla's engine) out of the browser.
- Tried for 2 days, failed. Named it **Web.js**.
- Later used **V8 engine** (from Chrome).
- Created **Node.js** – JavaScript runtime for backend.
- Joined company **Joyent**.
- Initially worked only on **Mac and Linux**.

2010

- **Isaac** (Isaac Schlueter) created **npm** (Node Package Manager).
- Joined **Joyent**.

2011

- **Joyent** collaborated with **Microsoft**.
- Released **Node.js for Windows** support.

2012

- **Ryan Dahl** left Node.js project.
- Project handed over to **Isaac**.

2014

- **Fedor**, co-developer of Node.js, left Joyent.
- Created a **forked version of Node.js** named **IO.js**.

2015

- To resolve confusion between Node.js and IO.js, a foundation was created:
  - **Node.js Foundation** established.
  - Merged **Node.js + IO.js** under this foundation.
- Started regular releases.
- Introduced **LTS (Long Term Support)**:
  - Valid for **30 months / 2.5 years**.
- JS Frontend ecosystem evolved:
  - ES6
  - Frameworks: **React**, **Angular**, **Vue**
  - Feature: **ES Modules** (`import` , `export`)

- Problem: Node.js only supported older CommonJS (`require` , `module.exports`).

2018

- **Node.js v10** released.
- Officially started supporting **ECMAScript Modules (ESM)**.

2019

- **JS Foundation + Node.js Foundation** merged.
- New body formed: **OpenJS Foundation**.

2020

- **Ryan Dahl** returned.
- Introduced **Deno** as a **secure competitor to Node.js**.

Feature	Node.js	Deno
Uses npm	Yes	No
Package Mgr	npm	Built-in (No installation required)

Modern Module System (Side Note)

ES6 Modules (Frontend & Modern JS)

```
import something from 'module';
export const data = ...;
```

CommonJS (Old Node.js)

```
const something = require('module');
module.exports = { ... };
```

Node.js v10+

- Supports both ES6 Modules and CommonJS.
- Extension `.mjs` for ESM or use `"type": "module"` in `package.json`.

Summary Timeline

```
2009 -> Ryan created Node.js using V8 (Joyent)
2010 -> Isaac created npm
2011 -> Node.js Windows support via Microsoft
2012 -> Ryan left, Isaac takes over
2014 -> Fedor forked to IO.js
2015 -> Node.js Foundation merges both
2016 -> LTS support introduced
2018 -> ES Module support (v10)
2019 -> OpenJS Foundation created
2020 -> Ryan returns, introduces Deno
```
