

01:53

→ Javascript

- ↳ Synchronous
- ↳ Single threaded

→ Execution Context

- ↳ execute one line of code at a time

→ console log → 1

→ console log → 2

CALL Stack

Memory Heap

Default

Each operation waits for the last one to complete before executing


05:51

Blocking Code VS Non Blocking code

- ↓
- Block the flow of Program
- ↓
- Read File Sync

- ↳ Does not block execution
- ↓
- Read File Async

12:29



Handwritten diagram illustrating the JavaScript Event Loop and Async/Await mechanism.

JS Engine

- Memory Heap**: A box representing memory.
- CALL STACK**: A stack of function calls. The bottom frame is labeled **Global**. Above it are three frames labeled **fn**. An arrow labeled **call** points from the top **fn** frame to the **Web API** box.

Web API

- DOM API**: A box containing **Set timeout** (with a dashed arrow pointing to **Register CALL BACK**), **Set interval**, and **fetch()**.
- Promise**: A box containing **CB** and **CB**. An arrow labeled **Promise** points from **fetch()** to this box.
- task Queue**: A box containing **CB** and **CB**. An arrow points from the **Promise** box to this queue.
- Register CALL BACK**: A box containing **CB** and **CB**. A dashed arrow points from **Set timeout** to this box.

Event Loop

- A dashed arrow labeled **High Priority** points from the **task Queue** to the **CALL STACK**.
- A dashed arrow labeled **Add to CALL Stack** points from the **task Queue** to the **CALL STACK**.

