

Context

Synchronous code runs first: The JavaScript engine starts with the main thread, executing all the synchronous code line by line. This is where it all begins—right in the call stack!

Asynchronous code runs later: Anything asynchronous—like `setTimeout`, Promises, or AJAX calls—has to wait. The main thread must finish its work first, and only then does the event loop step in to handle these tasks.

The Event Loop waits for the call stack to be empty: The event loop is patient. It watches the call stack, waiting for it to empty. As long as the call stack is busy, the event loop does nothing, meaning it will not watch neither microtask or macrotask queues.

Microtasks have priority over macrotasks: The event loop doesn't jump straight into handling macrotasks (like `setTimeout`). No, it respects the microtask queue first. Until every microtask is dealt with, macrotasks are ignored.

Microtasks are executed one-by-one: To clear the microtask queue, the event loop pulls tasks from it one by one, pushing them onto the call stack. It is a meticulous process, ensuring each microtask is executed to completion.

Macrotasks have to wait: Once the microtask queue and call-stack is empty, the event loop turns its attention to the macrotask queue. It takes just one macrotask, pushes it onto the call stack, and then—before moving on to the next macrotask—it checks again if call-stack or any microtasks are pending. If there are, call-stack and microtasks take precedence once more.

`console.log()` is synchronous: The humble `console.log()` is a synchronous call. It runs immediately and does not wait for anything or anyone.

Callbacks in `then` and `catch` are async and go to the microtask queue: When a Promise is resolved or rejected, its associated callbacks—the ones inside `then` and `catch`—are queued in the microtask queue. They are executed as soon as the call stack is clear.

`setTimeout` callbacks are async and go to the macrotask queue: Callbacks inside a `setTimeout` do not rush. They calmly wait in the macrotask queue, ready to be picked up only after all synchronous code and microtasks have had their turn.