Day 5 — Event Loop

1. Event Loop

JavaScript is single-threaded, but still capable of handling asynchronous behavior using the Event Loop.
Key components include:
- Call Stack
- Heap
- Task Queue (Macrotasks)
- Microtask Queue
- Timers
- Promise Jobs

2. Call Stack

The Call Stack is where synchronous code executes. Functions are pushed onto the stack when invoked and popped off when they finish.
Execution follows a Last-In-First-Out model.

3. Heap

The Heap is where objects are stored in memory. Variables reference object locations in the heap from the call stack.

4. Task Queue vs Microtask Queue

Task Queue (Macrotasks): Handles callbacks from setTimeout, setInterval, and I/O events.
Microtask Queue: Handles Promise.then/catch/finally and queueMicrotask.
Microtasks always run before any macrotask once the stack is empty.

5. Timers & setTimeout(0)

setTimeout does not run immediately after its delay. Instead, its callback is placed into the Task Queue.
Even setTimeout(0) waits until:
- The Call Stack is empty
- All Microtasks have finished

6. Promise Jobs

Promise callbacks are placed into the Microtask Queue, giving them higher priority over timers and other macrotasks.

7. Event Loop Cycle

The Event Loop repeatedly performs:
1. Run synchronous code in the Call Stack
2. Execute ALL Microtasks
3. Execute ONE Macrotask
4. Repeat

8. Event-Loop Timeline Example

Consider the following code snippet:

```
console.log("start");

setTimeout(() => console.log("timeout"), 0);

Promise.resolve().then(() => console.log("promise"));

console.log("end");
```

Timeline showing how the Event Loop schedules execution:

Step	Call Stack	Microtask Queue	Task Queue	Output so far
1	global() → console.log('start')	[]	[]	start
2	global() (after setTimeout scheduled)	[]	[] (timer pending)	start
3	global() (after Promise.then registered)	[promise callback]	[] (timer still pending)	start
4	global() → console.log('end')	[promise callback]	[]	start, end
5	promise callback (microtask running)	[]	[timeout callback]	start, end, promise
6	timeout callback (macrotask running)	[]	[]	start, end, promise, timeout