JavaScript Event Loop – One Page Revision Notes

1. JavaScript Execution Model
• JavaScript is single-threaded
• Executes all synchronous code first
• Synchronous code runs in the Call Stack

2. setTimeout
• setTimeout does not execute immediately
• Timer is handled by Web APIs
• Callback is queued only after delay finishes

3. setTimeout(0)
• 0ms means minimum delay, not instant execution
• Callback goes to Macrotask (Callback) Queue
• Executes only after Call Stack and Microtasks are empty

4. Event Loop Priority
Execution order:
1. Call Stack (Synchronous code)
2. Microtask Queue (Promises)
3. Macrotask Queue (setTimeout)

5. Multiple setTimeouts
• Timers run in parallel
• Execution depends on which timer finishes first
Example: 2s → 5s → 10s

6. Same Timeout Values
• If delays are equal, insertion order is followed

7. Key Interview Line
JavaScript executes synchronous code first, then microtasks like Promises, and finally macrotasks like setTimeout based on timer completion.