Event loop interview questions

> provide a list of questions of 15 theory/concept based questions 5 easy 5 meadium, 5 hard of event loop, task queue, micro task queue concepts questions mostly asked in interviews.

Here's a **complete interview-style list of 15 theory/concept-based questions** about the **Event Loop**, **Task Queue**, and **Microtask Queue** — categorized by **difficulty level (Easy, Medium, Hard)** — along with short hints and what interviewers usually look for 👇

🟢 EASY LEVEL (Basic Understanding)

1. What is the Event Loop in JavaScript?

- **Concept Tested:** Core mechanism of JS concurrency model.
- **Hint:** Explain how it manages execution of synchronous and asynchronous code.

---

2. Why is JavaScript called a single-threaded language?

- **Concept Tested:** Threading model of JS.
- **Hint:** Only one call stack — one piece of code executes at a time.

---

3. What is the difference between synchronous and asynchronous code?

- **Concept Tested:** Blocking vs non-blocking execution.
- **Hint:** Synchronous → executes line by line; Asynchronous → deferred execution via callbacks, promises, etc.

---

4. What is the role of the Call Stack in the Event Loop?

- **Concept Tested:** How functions are executed.
- **Hint:** Stack structure (LIFO), executes functions, blocks event loop until empty.

---

5. What is the Task Queue (or Callback Queue)?

- **Concept Tested:** Macrotask queue basics.
- **Hint:** Stores callbacks from `setTimeout`, `setInterval`, etc. executed after stack clears.

🟡 MEDIUM LEVEL (Deeper Understanding)

6. What is the Microtask Queue, and how is it different from the Macrotask Queue?

- **Concept Tested:** Order of execution and prioritization.
- **Hint:** Microtasks (Promises, queueMicrotask, MutationObserver) are executed before next macrotask.

---

7. Explain the order of execution between `setTimeout` and `Promise.then`.

- **Concept Tested:** Event loop execution order.
- **Hint:** `Promise.then` (microtask) executes before `setTimeout` (macrotask).

---

8. How does the Event Loop handle multiple Promises chained together?

- **Concept Tested:** Microtask queue re-queuing.
- **Hint:** Each `.then()` adds a new microtask; all microtasks run before next macrotask.

---

9. What is the difference between the Job Queue and the Microtask Queue?

- **Concept Tested:** Terminology used in ECMAScript specification.
- **Hint:** They are effectively the same — "Job Queue" is the spec name, "Microtask Queue" is the browser term.

---

10. How does the Event Loop differ between browsers and Node.js?

- **Concept Tested:** Environment-specific behavior.
- **Hint:** Node.js uses libuv — it has separate phases (timers, I/O callbacks, check, close, microtasks after each phase).

🔴 HARD LEVEL (Advanced and Tricky)

11. Can a Promise callback delay a `setTimeout` from executing? Explain why or why not.

- **Concept Tested:** Event loop starvation.
- **Hint:** If microtasks keep scheduling new microtasks, macrotasks (like setTimeout) can be delayed indefinitely.

---

12. What happens if you queue a microtask inside another microtask?

- **Concept Tested:** Recursive scheduling behavior.
- **Hint:** It executes in the same microtask phase before moving to the next macrotask.

---

13. Explain what happens step by step in this code:

```
js

console.log("A");
setTimeout(() => console.log("B"), 0);
Promise.resolve().then(() => console.log("C"));
console.log("D");
```

- **Concept Tested:** Execution order reasoning.
- **Expected Output:** A, D, C, B
- **Hint:** Stack → microtasks → macrotasks.

---

14. Why does adding too many microtasks slow down the browser, and how can it lead to UI blocking?

- **Concept Tested:** Event loop responsiveness.
- **Hint:** Microtasks run before rendering — if many are queued, rendering (macrotasks) is delayed.

---

15. Describe in detail what happens in this sequence:

```
js

setTimeout(() => console.log("T1"), 0);
Promise.resolve().then(() => {
  console.log("P1");
  setTimeout(() => console.log("T2"), 0);
});
Promise.resolve().then(() => console.log("P2"));
console.log("S");
```

- **Concept Tested:** Deep event loop trace and ordering.
- **Expected Output:**
  S
  P1
  P2
  T1
  T2
- **Hint:** Microtasks before macrotasks, but inner macrotasks (T2) go to a new macrotask cycle.

✅ Summary Table

Level	Focus Area	Example Concept
Easy	Basic terms: Event Loop, Stack, Queue	Definition-level questions
Medium	Ordering logic	Microtask vs Macrotask behavior
Hard	Real-world event loop flow	Timing, starvation, nested queues

Would you like me to make a **visual diagram** showing how the **call stack, microtask queue, and macrotask queue** interact during execution?
(It really helps understand these 15 questions deeply.)