20 JavaScript Interview Questions for Frontend Developers in 2025

1. Explain the difference between Promise.all(), Promise.allSettled(), and Promise.any().
2. How does the Nullish Coalescing Operator (??) differ from OR (||)?
3. What are WeakMap and WeakSet, and when would you use them?
4. Explain the concept of Top-Level Await.
5. How do you implement proper error boundaries in JavaScript applications?
6. What happens when you mix async/await with .then()/.catch()?
7. Explain the event loop with microtasks and macrotasks.
8. How would you implement a retry mechanism for failed API calls?
9. What is the difference between debouncing and throttling? Implement both.
10. How does JavaScript garbage collection work, and how can you optimize for it?
11. Explain tree shaking and how it affects your code.
12. What are Web Workers and when would you use them?
13. How do you handle state management without external libraries?
14. Explain the Module Federation pattern.
15. What are JavaScript Proxies and how can they be used?
16. How would you implement a custom hook pattern in vanilla JavaScript?
17. How do you prevent XSS attacks in JavaScript applications?
18. What is Content Security Policy and how does it affect JavaScript?
19. How would you test asynchronous code without external testing frameworks?
20. Explain different types of JavaScript testing (unit, integration, e2e) and their trade-offs.

Some of the concepts I was grilled on:

- Execution Context & Call Stack – how JS runs code step by step.
- Hoisting & Closures – what gets lifted, and the hidden powers of lexical scope.
- Event Loop & Concurrency Model – microtasks vs macrotasks.
- Promises & async/await – handling chains, errors, and tricky pitfalls.
- this keyword & Prototypes – implicit/explicit binding, inheritance.
- Performance & Memory – garbage collection, Big-O basics, leaks.
- Modern ES6+ Features – optional chaining, spread/rest, immutability.
- DOM & Event Delegation – interview classics that never fade.

- What is the difference between var, let, and const?
- How does JavaScript handle asynchronous operations (Promises, async/await)?
- Can you explain event bubbling vs. event capturing?
- What is hoisting in JavaScript?

- How does this keyword work in different contexts?
- Difference between == and ===?
- How do closures work, and why are they useful?
- What are some JavaScript ES6+ features you use often?

The discussion revolved around core and advanced concepts such as:

👉 Event Loop & Call Stack
👉 Closures & Scope
👉 Promises vs Async/Await
👉 Prototypes & Inheritance
👉 Array methods like map(), filter(), reduce()

It reinforced my belief that mastering JavaScript at its core makes frameworks much easier to understand and apply.

👉 JavaScript Code Execution & Global Execution Context
👉 Event Loop, Call Stack & Concurrency
👉 Hoisting (Variables & Functions)
👉 var vs let vs const
👉 Arrow Functions vs Traditional Functions & this
👉 Closures in JavaScript
👉 Classes, Objects & Getters/Setters
👉 Working with JavaScript Objects
👉 Optional Chaining (?.), Nullish Coalescing (??), Logical OR (||)
👉 Object & Array Destructuring
👉 Shallow vs Deep Copy
👉 Array Methods: map(), filter(), reduce(), join(), split(), find(), findIndex(), splice(), slice()
👉 Apply, Bind & Call Methods
👉 Prototype & Inheritance
👉 Boolean Values, Ternary Operator & Short-Circuiting
👉 Spread & Rest Operators
👉 Set & Map
👉 Debouncing & Throttling in JavaScript
👉 Event Delegation & Event Bubbling
👉 Promises & Async/Await

👉 Memoization

👉 ES6 Modules (export vs export default)

🌟 Key takeaway: Frameworks change, but strong JavaScript knowledge never goes out of demand.