JAVASCRIPT INTERVIEW PREP

JS Concepts You Can't Miss

59 core concepts · 14 polyfills to implement · curated practice links

---

What's inside

1 Scope, Hoisting & Execution Context

1. Scope in JS — global, functional, and block scope
2. Scope chaining
3. var, let and const
4. Temporal Dead Zone (TDZ)
5. Hoisting
6. Variable shadowing
7. Lexical environment
8. Lexical scope vs dynamic scope — which does JavaScript use? UPDATED
9. Deep dive on TDZ — why typeof x throws for let x before declaration, but not for an undeclared variable UPDATED
10. Execution context
11. Execution context: creation phase vs execution phase — what gets stored in the lexical environment? UPDATED
12. Call stack
13. The 'this' keyword in JS
14. Strict mode
15. Garbage collection
16. How JavaScript parses and compiles code, step by step

2 Data Types, Objects & Prototypes

1. Primitive vs non-primitive types
2. Pass by reference vs pass by value
3. undefined vs not-defined vs null
4. Equality: === vs ==
5. Equality comparison deep dive — ==, ===, Object.is, SameValueZero. Why is NaN === NaN false but Object.is(NaN, NaN) true? UPDATED
6. Type coercion vs type conversion
7. How many ways can you create an object in JS?
8. Prototypes in JS
9. Prototype object
10. Prototype chaining
11. Prototype pollution — how unsafe object-merge logic can become a security issue UPDATED
12. What does 'static' mean in a JavaScript class?

3 Functions, Closures & Functional Patterns

1. Closures
2. First class functions
3. Higher order functions

4. Function declaration vs expression vs anonymous vs arrow functions
5. Immediately Invoked Function Expressions (IIFE)
6. call, apply and bind — why we use them
7. Currying in JS
8. The infinite currying problem
9. Memoization in JS
10. Rest parameters
11. Spread operator
12. Generator functions
13. Throttle
14. Debounce
15. mapLimit

4 Asynchronous JavaScript & the Event Loop

1. JavaScript is a single-threaded language
2. Callbacks — why do we need them?
3. Callback hell
4. The event loop
5. Callback queue
6. Microtask queue
7. If a microtask keeps re-queuing itself infinitely, how do you handle it? UPDATED
8. Promises
9. Async / await
10. .then() and .catch()
11. async vs defer (script loading)

5 DOM Events

1. Event propagation
2. Event bubbling
3. Event capturing
4. stopPropagation()
5. Event delegation

6 Polyfills to Practice

Implement each of these from scratch — the fastest way to actually understand the concept.

1. map, reduce, filter, forEach, find
2. call, apply, bind
3. Promise, Promise.all(), Promise.any(), Promise.allSettled(), Promise.race()
4. mapLimit
5. Debounce
6. Throttle
7. Event emitter
8. setInterval polyfill
9. Parallel limit function
10. Deep vs shallow copy
11. Flatten a deeply nested object
12. Memoization / caching
13. Promise.finally()
14. Retry (with backoff)

7 Best Videos & Practice Links

- **Practice — LeetCode 30 Days of JS** V.IMP
  Structured coding-question practice set.
- **Playlist — Roadside Coder**
  Polyfills and interview questions, explained.
- **Playlist — Namaste JavaScript**
  Deep-dive into every core JS concept.
- **Docs — javascript.info**
  Fundamentals, data types, and language reference.

> *Tip: when a concept feels fuzzy, ask an AI assistant to go deeper — curiosity plus a good explainer beats memorizing definitions.*