JavaScript & Typescript

PHASE 1 — JavaScript Execution & Fundamentals (GATE 1)

You **cannot** move ahead unless you can *reason about execution*.

Must Know Topics

- JavaScript runtime architecture
    - Call Stack
    - Heap
    - Web APIs
    - Callback Queue
- Single-threaded model (what it really means)
- Synchronous vs asynchronous execution
- Blocking vs non-blocking code
- Stack overflow & memory allocation basics

You're ready if you can:

- Explain **how JS executes a program line by line**
- Explain **why async code doesn't block**
- Debug infinite recursion mentally

📦 PHASE 2 — Variables, Scope & Data Types (GATE 2)

Interviewers *love* trick questions here.

Must Know Topics

- `var`, `let`, `const` (actual runtime behavior)
- Hoisting (functions vs variables)
- Temporal Dead Zone (TDZ)
- Global scope pollution
- Primitive vs reference types
- `typeof` quirks
- Pass-by-value vs pass-by-reference (correct explanation)

You're ready if you can:

- Predict outputs with hoisting
- Explain **why `const` doesn't mean immutable**
- Explain why objects behave differently in functions

🔢 PHASE 3 — Arrays (HIGH WEIGHT) (GATE 3)

If arrays are weak → rejection risk is high.

Must Know Topics

- Core methods

    - `map`, `filter`, `reduce`
    - `forEach` vs `map`
- `slice` vs `splice`
- `sort` pitfalls
- Mutating vs non-mutating methods
- Time complexity of array operations

Must-Do Patterns

- Frequency counting
- Deduplication
- Grouping
- Chunking
- Sliding window
- Flattening nested arrays

You're ready if you can:

- Write **polyfills** for `map`, `reduce`
- Solve array problems without libraries
- Explain performance trade-offs

🧩 PHASE 4 — Objects (HIGH WEIGHT) (GATE 4)

Objects test **design thinking**, not syntax.

Must Know Topics

- Object creation patterns
- Iteration (`for…in`, `Object.keys/values/entries`)
- `in` vs `hasOwnProperty`
- Shallow vs deep copy
- `Object.assign` vs spread
- `freeze`, `seal`, `preventExtensions`

Must-Do Tasks

- Deep clone object
- Flatten nested objects
- Transform object structure
- Merge objects safely

You're ready if you can:

- Avoid accidental mutation
- Explain reference sharing bugs
- Design clean object shapes

🔁 PHASE 5 — Functions & Closures (GATE 5)

This is where **average devs fail**.

Must Know Topics

- Function declaration vs expression
- Arrow functions (this behavior)
- IIFE
- Default params
- Rest & spread operators

Closures (NON-NEGOTIABLE)

- Lexical scope
- Memory retention
- Practical use cases

You're ready if you can:

- Explain closures without using buzzwords
- Identify memory leaks caused by closures
- Implement data hiding using closures

🧱 PHASE 6 — OOP & Prototypes (GATE 6)

JavaScript OOP ≠ Java OOP.

Must Know Topics

- Prototype chain
- `__proto__` vs `prototype`
- Constructor functions
- `class` syntax (what it abstracts)
- Inheritance
- Composition vs inheritance

You're ready if you can:

- Trace property lookup in prototype chain
- Explain how `class` works internally
- Justify composition in real systems

⏳ PHASE 7 — Async JavaScript & Event Loop (GATE 7)

One wrong answer here = rejection in many interviews.

Must Know Topics

- Event loop (step-by-step)
- Microtasks vs macrotasks
- Promise queue
- `setTimeout(0)` reality
- Callback hell & how promises fix it

Promises

- States
- Chaining
- Error propagation
- `all`, `race`, `any`, `allSettled`

Async/Await

- How it’s built on promises
- Sequential vs parallel execution
- Proper error handling

You’re ready if you can:

- Predict execution order
- Explain why a promise error wasn’t caught
- Write performant async code

🧠 PHASE 8 — `this`, Context & Binding (GATE 8)

Frequently used to **filter candidates**.

Must Know Topics

- `this` in:
  - Global scope
  - Object methods
  - Arrow functions
  - Callbacks
- `call`, `apply`, `bind`
- Loss of context bugs

You’re ready if you can:

- Predict `this` value in any situation
- Fix context issues confidently
- Write your own `bind` polyfill

🧪 PHASE 9 — Polyfills & JS Internals (GATE 9)

Signals **senior-level depth**.

Must Know Topics

- Polyfills for:
  - `map`
  - `reduce`
  - `bind`
  - `debounce`
  - `throttle`

- Garbage collection basics
- Memory leaks
- Performance pitfalls

You're ready if you can:

- Implement polyfills from scratch
- Identify memory leaks in code
- Optimize hot paths

🟦 PHASE 10 — TypeScript (FINAL GATE)

TypeScript is judged by **design**, not syntax.

Must Know Topics

- Type inference
- `any` vs `unknown`
- `never`, `void`
- Union & intersection
- Interfaces vs types

Advanced TS

- Generics
- Utility types
- Readonly & immutability
- Enums vs const enums

Real-World TS

- Typing Express APIs
- DTOs
- Zod + TS
- Migrating JS → TS

You're ready if you can:

- Design type-safe APIs
- Reduce runtime bugs using types
- Explain why TS improves large codebases

🎯 FINAL RULE (Important)

Before moving to **Express / System Design**:

You should be able to:

- Debug JS mentally
- Predict async output
- Write clean, immutable code
- Explain *why*, not just *what*