JavaScript Roadmap for Web Engineering (React, Next.js, Node.js, Express)

1. Core JavaScript (Foundations)

• Variables: var, let, const
• Data types (primitive vs reference)
• Operators (arithmetic, comparison, logical)
• Conditionals (if, switch)
• Loops (for, while, for…of, for…in)
• Functions: declarations, expressions, arrow functions, scope, closures
• Objects & Arrays: destructuring, array methods (map, filter, reduce), spread/rest
• DOM: selecting elements, events, manipulating DOM

2. Intermediate JavaScript

• ES6+ Features: template literals, destructuring, modules, async/await, promises, classes
• Error Handling: try/catch, custom errors
• Asynchronous Programming: callbacks, promises, async/await, event loop
• Advanced Array & Object Techniques: reduce, Object.keys/values/entries, deep vs shallow copy

3. Advanced JavaScript (Engineering Concepts)

• Execution Model: call stack, hoisting, closures, this keyword
• Prototypes & OOP: prototype chain, inheritance, classes
• Functional Programming: higher-order functions, pure functions, currying, composition
• Modules & Bundling: ES modules, CommonJS, bundlers
• Event Handling: bubbling vs capturing, custom events
• Memory & Performance: garbage collection, debouncing, throttling

4. JavaScript for React & Next.js

• Destructuring props & state
• Array methods for rendering lists
• Spread/rest operators for immutability
• Event handling & synthetic events
• Closures in hooks (stale state problem)
• Shallow vs deep equality (re-renders)
• Async state updates & batching
• Next.js: async/await for data fetching, server vs client context

5. JavaScript for Node.js & Express

• Node Basics: require vs import, fs, EventEmitter, streams, process env
• Asynchronous Patterns: callbacks, promises, async/await
• Express: middleware functions, req/res objects, routing, async handlers, error handling middleware

6. Engineering Practices

• Type Safety: JSDoc, TypeScript basics
• Code Quality: ESLint, Prettier
• Testing: Jest, Supertest
• Patterns: MVC, component-based design, middleware
• Security: sanitizing input, avoiding callback hell, async best practices

7. Full-Stack JavaScript

• Fetching APIs with fetch/axios
• Understanding CORS
• JWT authentication flow
• Handling cookies & sessions
• Real-time with WebSockets
• File uploads & form-data

Suggested Learning Path Order

• 1. Core JS
• 2. Intermediate JS
• 3. Advanced JS
• 4. React JS
• 5. Next.js
• 6. Node.js
• 7. Express
• 8. Full-stack integration projects