



# The Ultimate 6-Week / 49-Day Elite Frontend Roadmap

 15th July – 1st September 2025


 Your mission: Go from developer → FAANG-ready engineer

---







✨ *"Discipline is the bridge between goals and accomplishment."*

— Jim Rohn

---


 Goal: Crack high-paying frontend roles at FAANG, Unicorns, or Product Companies

 Profile: React Developer (3 years exp, working professional)

 Format:  Must-Do,  Should-Prepare |  Beginner |  Intermediate |  Advanced



---

## WEEK 1: React Core Foundations

 15th July – 21st July 2025

---

## 15th July – JSX + Functional Components

 Beginner +  Must-Do

1. JSX is a syntax extension that allows you to write UI in a HTML-like format in JavaScript.
2. It is the foundation of all React UIs and used in every component.



 Checklist:

- ☐ Create a `ProfileCard` component using JSX
- ☐ Compile JSX using Babel to understand transformation

 Q: What does JSX compile to behind the scenes?

---

## 16th July – Props vs State

 Beginner +  Must-Do

1. Props are inputs to a component; state is managed inside the component.
2. Knowing the difference is vital to building dynamic, reusable components.



 Checklist:

- ☐ Create a `Greeting` component with name passed via props
- ☐ Build a `Counter` using `useState`

 Q: How do you decide whether to use props or state?

---

## 17th July – Controlled vs Uncontrolled Components

 Beginner +  Must-Do

1. Controlled components rely on state; uncontrolled components use DOM refs.
2. Important when dealing with forms and input validation.



 Checklist:

- ☐ Build a login form using `useState`
- ☐ Rebuild the same using `useRef`

 Q: What are pros/cons of controlled vs uncontrolled inputs?

---

## 17 18th July – `useEffect` + Event Handling

 Beginner +  Must-Do

1. `useEffect` handles side effects like data fetching or subscriptions.
2. Mastery of this is essential for data-driven UIs.

 Checklist:

☐ Add an event listener inside `useEffect`

☐ Clean up the listener on unmount

 Q: What causes `useEffect` to rerun?

---

## 17 19th July – Rendering + Reconciliation

 Intermediate +  Must-Do

1. React's virtual DOM re-renders based on diffing changes.
2. Helps prevent unnecessary UI updates and bugs.

 Checklist:

☐ Create a list using `.map()` with dynamic keys

☐ Replace keys with index and observe the impact

 Q: Why should you avoid using index as a key?

---

## 17 20th July – `useEffect` Deep Dive

 Intermediate +  Must-Do

1. Advanced use of `useEffect` includes handling stale closures and debounce.
2. Many bugs arise from incorrect dependency usage.

 Checklist:

☐ Implement debounced input

☐ Simulate stale closure using outdated state in effects

 Q: What is a stale closure and how do you avoid it?

---

## 17 21st July – Mini Project: Movie Search

🟢/🟡 Mixed + ✅ Must-Do

1. Builds core concepts with real-world use.
2. Shows initiative and UI completeness.

🎯 Checklist:

- ☐ Build a search UI using OMDb API
- ☐ Add loading, error, and empty states

📖 Q: How would you handle error boundaries in this project?

---

## ✅ WEEK 2: Intermediate React Skills

📅 17 22nd July – 28th July 2025

---

### 📅 17 22nd July – useRef + useCallback

🟡 Intermediate + ✅ Must-Do

1. `useRef` holds mutable DOM references or values across renders without triggering re-renders. `useCallback` memoizes function instances between renders.
2. Useful for preventing unnecessary re-renders in lists, child components, and forms.

🎯 Checklist:

- ☐ Auto-focus an input with `useRef`
- ☐ Prevent function recreation with `useCallback` inside a parent component

📖 Q: When should you use `useCallback`?

---


### 📅 17 23rd July – useMemo + React.memo

🔴 Advanced + ✅ Must-Do

1. Memoization prevents unnecessary recalculations ( `useMemo` ) or re-renders ( `React.memo` ).
2. Optimizing component and computation performance is key in large-scale apps and interviews.

### Checklist:

- ☐ Render a large filtered list using `useMemo`
- ☐ Wrap child components in `React.memo` and track when they re-render

 Q: When can memoization actually hurt performance?

---

## 17 **24th July – Forms with React Hook Form / Formik**

 Intermediate +  Must-Do

1. RHF and Formik are libraries to manage form state, validation, and input bindings efficiently.
2. Common in dashboards, onboarding forms, and feature-rich apps.

### Checklist:

- ☐ Build signup form using React Hook Form
- ☐ Add schema validation with Yup

 Q: How do RHF and Formik differ?

---


## 17 **25th July – Custom Hooks**

 Intermediate +  Must-Do

1. Custom hooks allow you to extract and reuse stateful logic across components.
2. Encouraged for maintaining DRY code in professional React codebases.

### Checklist:

- ☐ Build `useFetch`, `useDarkMode`, `useLocalStorage`
- ☐ Use them in a form or widget

 Q: What rules must a custom hook follow?


---

## 17 **26th July – React Router v6**

 Intermediate +  Must-Do

1. React Router handles routing, nested routes, and dynamic parameters in React SPAs.
2. Required in nearly all apps; also common in interviews.

### Checklist:


- ☐ Build Home, About, Profile pages with `Routes`
  - ☐ Add nested layout with `Outlet`
  -  Q: How does `useParams` work in nested routes?
- 

## 17 **27th July – Route Guards + Lazy Routing**




 Advanced +  Should-Prepare

1. Route guards restrict access to private pages; lazy routing helps reduce load time.
2. Both are essential for auth-based production SPAs.

### Checklist:


- ☐ Create `PrivateRoute` that checks auth context
  - ☐ Lazy load dashboard pages using `React.lazy` and `Suspense`
  -  Q: How do you protect private routes in a React SPA?
- 

## 17 **28th July – Project: Netflix UI Clone**


  Mixed +  Must-Do

1. A UI project that combines routing, cards, search, and auth simulation.
2. Demonstrates frontend polish and ability to build real-world UIs.

### Checklist:

- ☐ Clone Netflix homepage layout with responsive movie cards
  - ☐ Add search, routing, and dummy login using context
  -  Q: How would you scale this with a real API and state management?
- 

## **WEEK 3: Context, Global State, Architecture**

 17 29th July – 4th August 2025

---

## 17 **29th July – Context API**

 Intermediate +  Must-Do

1. Context provides a way to share state between components without prop drilling.
2. Ideal for themes, language, and auth flows; essential for scalable UIs.

🎯 Checklist:

- ☐ Create a `ThemeContext` to toggle light/dark mode
- ☐ Wrap components with provider and consume via `useContext`

📖 Q: When should you avoid using Context?

---

## 17 **30th July – useReducer + Global State**

🟡 Intermediate +  Must-Do

1. `useReducer` manages complex state logic using reducer patterns, similar to Redux.
2. Important when dealing with structured UI logic like carts or filters.

🎯 Checklist:

- ☐ Create a `CartReducer` for add/remove/update items
- ☐ Combine `useReducer` with Context to build global state

📖 Q: How does `useReducer` compare with `useState`?

---

## 17 **31st July – Redux Toolkit / Zustand**

🔴 Advanced +  Should-Prepare

1. Redux Toolkit simplifies Redux patterns; Zustand is a lightweight alternative to Redux.
2. Useful in large-scale apps where shared state crosses many components.


🎯 Checklist:

- ☐ Build a counter app with Redux Toolkit
- ☐ Build the same with Zustand's `create()`

📖 Q: When would you prefer Zustand over Redux?

---

## 17 **1st August – Atomic Design + Reusability**

🔴 Advanced +  Must-Do

1. Atomic Design divides UI into Atoms (Button), Molecules (FormGroup), Organisms (CardList), etc.
2. Encouraged in large-scale design systems and component libraries.


🎯 Checklist:

- ☐ Refactor components into `atoms/`, `molecules/`, `organisms/`
- ☐ Build reusable `Button`, `Input`, `Card`, `Navbar` components

📖 Q: How does Atomic Design enable scalable UI systems?

---

## 17 **2nd August – Folder Structure & Project Patterns**

🔴 Advanced +  Must-Do

1. Proper folder structure improves maintainability and team productivity.
2. Common interview topic for architecture questions.

🎯 Checklist:

- ☐ Refactor project into `/features/`, `/shared/`, `/lib/`, `/hooks/`, `/contexts/`
- ☐ Setup an alias system for imports ( `@/features/...` )

📖 Q: How would you structure a scalable frontend codebase?

---

## 17 **3rd August – Headless Components**

🔴 Advanced +  Should-Prepare

1. Headless components expose only logic (not UI), allowing full design flexibility.
2. Popular in UI libraries like Radix UI and Downshift.


🎯 Checklist:

- ☐ Build a `useAccordion` or `useSelect` hook
- ☐ Pass render functions (render props or children as function)

📖 Q: What are the benefits of headless components?

---

## 17 **4th August – Project: E-Commerce UI**

🟢🔴 Mixed +  Must-Do

1. Practical use of context, reducer, reusable components, and routing.



2. Great portfolio project that tests architectural decisions and UI depth.

🎯 Checklist:

- ☐ Product list with filters, sort, and categories
- ☐ Global cart with Context + Reducer
- ☐ Responsive layout with Tailwind or custom CSS

📖 Q: How would you persist cart state between sessions?

---

## ✅ WEEK 4: UI Frameworks, Performance, Tooling

📅 17 5th August – 11th August 2025

---

### 📅 17 5th August – TailwindCSS + ShadCN

🟡 Intermediate + ✅ Must-Do

1. TailwindCSS is a utility-first CSS framework for rapid, responsive styling. ShadCN offers beautiful prebuilt components built on top of Tailwind + Radix.
2. Popular in modern startups and production-grade UI systems.

🎯 Checklist:

- ☐ Setup Tailwind in your React project
- ☐ Use ShadCN's `Button`, `Alert`, and `Dialog` in a demo

📖 Q: What are pros/cons of Tailwind vs CSS Modules?

---

### 📅 17 6th August – Accessibility (a11y)

🟡 Intermediate + ✅ Must-Do

1. Accessibility ensures apps are usable by people with disabilities (screen readers, keyboard users).
2. Critical for inclusive design, and often evaluated in frontend interviews.

🎯 Checklist:

- ☐ Add semantic HTML (e.g., `label`, `fieldset`, `main`)
- ☐ Add ARIA roles + focus traps in modals

📖 Q: How do you make a modal accessible?

---

## 17 7th August – Memoization + React DevTools

🔴 Advanced + ✅ Must-Do

1. Memoization ( `useMemo` , `useCallback` , `React.memo` ) helps avoid unnecessary calculations and renders.
2. Improves app performance and prevents wasteful re-renders.

🎯 Checklist:

- ☐ Use React DevTools Profiler to inspect re-renders
- ☐ Optimize an expensive computation with `useMemo`
- ☐ Prevent child re-renders using `React.memo`

📖 Q: When is memoization counterproductive?

---

## 17 8th August – Lazy Loading + Code Splitting

🔴 Advanced + ⚠️ Should-Prepare

1. Lazy loading defers code execution for faster initial load; code-splitting separates parts of your app.
2. Helps with time-to-interactive and large-bundle scaling.

🎯 Checklist:

- ☐ Implement route-based code-splitting using `React.lazy`
- ☐ Wrap lazy components with `Suspense` and fallback

📖 Q: What is the difference between lazy loading and dynamic imports?

---

## 17 9th August – ESLint + Prettier + Husky


🟡 Intermediate + ✅ Must-Do

1. ESLint enforces code quality; Prettier handles code formatting; Husky sets up Git hooks.
2. Helps maintain consistent code and prevent broken commits.

🎯 Checklist:

- ☐ Setup `.eslintrc` , `.prettierrc` , `.husky` and `lint-staged`

☐ Add Git pre-commit hook to format code

 Q: How do lint-staged and Husky help in CI pipelines?

---

## 17 **10th August – Vite + TurboRepo Setup**


 Intermediate +  Must-Do

1. Vite is a fast dev server and build tool; TurboRepo enables efficient monorepo management.
2. Improves DX and lets teams split frontend into multiple composable packages.

 Checklist:



☐ Create a Vite app inside `/apps/`

☐ Setup `packages/ui` for shared UI with TurboRepo

 Q: How does TurboRepo speed up builds compared to traditional monorepos?

---

## 17 **11th August – Project: Admin Dashboard**

 Advanced +  Must-Do

1. A CRUD-based admin UI with table filters, modal forms, and global state — perfect for portfolios.
2. Demonstrates component abstraction, performance, and data handling.

 Checklist:

☐ Create a user table with filters, sort, and pagination


☐ Add editable modal with validation

☐ Use Zustand or Context for state

 Q: How would you scale this dashboard to handle 10k+ records?

---

## **WEEK 5: Backend Integration & Fullstack Support**

 17 **12th August – 18th August 2025**

---

## 17 12th August – REST API + Axios/Fetch

🟡 Intermediate + ✅ Must-Do

1. REST APIs expose data over HTTP. Axios/Fetch are tools to call these endpoints from the frontend.
2. All real-world frontend apps interact with APIs — critical for interviews and take-home projects.

🎯 Checklist:

- ☐ Fetch users or posts using Axios/Fetch
- ☐ Handle loading, error, and success states in UI

📖 Q: How would you cancel a fetch request in React?

---

## 17 13th August – GraphQL Basics

🟡 Intermediate + ⚠️ Should-Prepare

1. GraphQL is a query language that lets you request only the fields you need.
2. Used at Meta, GitHub, Shopify — learning it shows adaptability to modern APIs.

🎯 Checklist:

- ☐ Setup Apollo Client and fetch data from GitHub GraphQL API
- ☐ Use `useQuery`, show loading/error states

📖 Q: What are the benefits of GraphQL over REST?

---

## 17 14th August – Auth (JWT / OAuth)

🟡 Intermediate + ✅ Must-Do

1. JWT (JSON Web Token) is used for stateless user authentication; OAuth is for delegated login (e.g. Google).
2. Essential for any app with user accounts and route protection.

🎯 Checklist:

- ☐ Simulate login + store token in `localStorage`
- ☐ Create `PrivateRoute` to protect dashboard


📖 Q: How do you securely manage auth tokens in frontend?

---


## 17 15th August – Express.js & Node Basics

 Intermediate +  Must-Do

1. Express is a minimalist web server used to build REST APIs in Node.js.
2. Helps you understand and test full-stack features independently.

 Checklist:

- ☐ Setup `/api/products` in Express
- ☐ Enable CORS and return JSON list

 Q: How do you protect an API route in Express?

---

## 17 16th August – PostgreSQL + Prisma ORM

 Advanced +  Should-Prepare

1. PostgreSQL is a relational DB; Prisma is a type-safe ORM for querying it from JavaScript.
2. Helps you build production-ready full-stack features with minimal setup.



 Checklist:

- ☐ Setup a `User` + `Product` model
- ☐ Run migration and seed dummy data

 Q: Why is Prisma preferred over raw SQL in TypeScript apps?

---

## 17 17th August – Fullstack Mini Project (Auth + DB)

 Advanced +  Must-Do

1. Combines frontend, backend, and database into a mini full-stack app.
2. Shows end-to-end understanding — a strong hiring signal in take-home tests.

 Checklist:

- ☐ Create signup/login with Express + JWT
- ☐ Use Prisma for user storage and authentication
- ☐ Create protected React dashboard that shows current user


 Q: How would you structure this project for team collaboration?

---

## 17 **18th August – BONUS: TypeScript API Integration**

 Intermediate +  Should-Prepare

1. Add strong typing to API requests and responses with Axios + TypeScript.
2. Boosts developer confidence and helps avoid runtime bugs.


 Checklist:

- ☐ Add TypeScript types to API responses
- ☐ Create custom hook `useUser()` with typed return

 Q: How would you infer types from your backend responses?

---

## **WEEK 6: Testing, DevOps, System Thinking**

 17 **19th August – 25th August 2025**

---

### 17 **19th August – Unit Testing with Vitest / Jest**

 Intermediate +  Must-Do

1. Unit tests check individual components or functions in isolation.
2. Companies expect at least basic testing knowledge from frontend engineers.

 Checklist:

- ☐ Write unit tests for a `Button`, `Counter`, and `Card` component
- ☐ Mock props and test conditionally rendered UI

 Q: What is the difference between unit and integration testing?

---

### 17 **20th August – E2E Testing (Cypress / Playwright)**

 Intermediate +  Should-Prepare

1. End-to-End tests simulate real user flows in a browser.
2. Used in CI pipelines to validate critical paths (e.g., login, checkout).

 Checklist:

- ☐ Write Cypress test for login + dashboard access

☐ Take a screenshot on test failure

 Q: What's the trade-off between E2E and unit testing?

---

## 17 **21st August – GitHub Actions (CI/CD)**

 Intermediate +  Must-Do

1. GitHub Actions enables automation for tests, builds, and deployments on every commit or PR.
2. Almost every team now uses CI/CD — shows engineering maturity.

 Checklist:



☐ Setup `.github/workflows/test.yml` to run Jest on PR

☐ Add linting + formatting to workflow

 Q: How would you auto-deploy preview builds on PRs?

---

## 17 **22nd August – SSR vs CSR vs ISR**


 Advanced +  Must-Do

1. CSR (Client-side rendering), SSR (Server-side rendering), and ISR (Incremental static regen) define when your page gets rendered.
2. Core concept for Next.js, SEO, and performance-sensitive apps.

 Checklist:

☐ Build blog using `getStaticProps`, `getServerSideProps`, and ISR

☐ Use Vercel to test revalidation behavior

 Q: When would you choose SSR over CSR?

---

## 17 **23rd August – Monitoring (Sentry / LogRocket)**

 Intermediate +  Should-Prepare

1. Monitoring tools log frontend crashes, network issues, and user behavior.
2. Product teams use them to reduce bugs and improve UX.

 Checklist:

☐ Integrate Sentry in your React app

☐ Add breadcrumb logs and capture errors manually

📖 Q: How would you debug a client-only crash that QA can't reproduce?

---

## 17 **24th August – Docker Basics**

🟡 Intermediate + ⚠️ Should-Prepare

1. Docker packages your full app (frontend + backend + DB) into isolated containers.
2. Enables consistency between local, dev, staging, and prod environments.

🎯 Checklist:

- ☐ Dockerize your full-stack app
- ☐ Use `docker-compose` to run PostgreSQL + Node + React together

📖 Q: What are the benefits of Docker over local dev environments?

---

## 17 **25th August – Deployment (Vercel / Railway / Fly.io)**

🟡 Intermediate + ✅ Must-Do

1. Deployment platforms help ship your project live with CI/CD pipelines and autoscaling.
2. Vital for showing real apps to recruiters and interviewers.


🎯 Checklist:

- ☐ Deploy frontend to Vercel (or Netlify)
- ☐ Deploy backend + DB to Railway or Fly.io

📖 Q: How would you scale a Node app behind multiple React frontends?

---

## ✅ **WEEK 7: Soft Skills, Mock Interviews, Final Polish**

 17 **26th August – 30th August 2025**

---

## 17 **26th August – Product Thinking**

🟡 Intermediate + ✅ Must-Do

1. Product thinking is the ability to understand *why* you're building something — not just how.



2. FAANG companies value engineers who solve business/user problems, not just implement tickets.

🎯 Checklist:

☐ Choose a past feature and list how it improved user experience or business goals

☐ Redesign an existing flow to reduce steps while keeping value

📖 Q: Tell me about a time you simplified a feature without losing value.

---

## 17 **27th August – System Thinking**

🟡 Intermediate + ☒ Must-Do

1. System thinking means seeing the whole — how data, users, APIs, and UI interconnect.
2. Critical for debugging, feature planning, and interview scenarios.

🎯 Checklist:

☐ Draw a flowchart of how data flows through your app (e.g., login → dashboard → fetch profile)

☐ Debug one “what-if” scenario (e.g., token expires mid-fetch)

📖 Q: Tell me about a time you discovered a system flaw or bottleneck.

---

## 17 **28th August – Communication + Collaboration**

🟡 Intermediate + ☒ Must-Do

1. Clear communication (verbal, written, async) improves team performance and clarity.
2. Most companies filter candidates on soft skill + collaboration culture fit.

🎯 Checklist:

☐ Record a Loom or walkthrough video explaining your app

☐ Write a review or leave feedback on someone else’s code/design

📖 Q: Describe a time you resolved a miscommunication or helped a team move forward.

---

## 17 **29th August – Ownership & Accountability**

### 🟡 Intermediate + ✅ Must-Do

1. Taking ownership means going beyond your tasks — owning quality, delivery, and long-term impact.
2. Mid-to-senior level roles *demand* strong ownership stories.

#### 🎯 Checklist:

☐ Write a short report describing a bug, refactor, or feature you led end-to-end

☐ Identify one place in your project where you added long-term value

📖 Q: Describe a time you took ownership of delivery and drove it to completion.

---

## 📅 17 30th August – Mock Interview Day

### 🔴 Advanced + ✅ Must-Do

1. Practicing interviews helps you prepare under pressure, tighten communication, and build confidence.
2. This is your test run before the real one.

#### 🎯 Checklist:

☐ 30-minute live coding challenge (e.g., tabs component, controlled form, sortable table)

☐ 5 behavioral questions in STAR format (record or write answers)

📖 Q: What feedback did you get? What would you improve?

---

## 🎉 Bonus: Optional Final Wrap-up (31st August – 1st September)

*These two days are for polish, rest, and transition to job hunt.*

---

## 📅 17 31st August – Portfolio + Resume Polish

### 🟡 Intermediate + ✅ Must-Do

#### 🎯 Checklist:

☐ Add 2–3 projects with clean README, feature list, and live demo

☐ Focus your resume on **impact** and **metrics** ("increased X by Y%")

📖 Q: What project are you most proud of and why?

---

## 1st September – Final Review + Strategy Reset

 Intermediate +  Must-Do

 Checklist:

- ☐ Review all React concepts, performance patterns, and testing
- ☐ Create 2-week job strategy: referrals, mock calls, applications
- ☐ Rest, reflect, and recharge 