React.js Developer Interview Questions and Answers

React.js & Frontend Development

Q: What are the key features of React that make it suitable for enterprise-grade applications?
A: React's component-based architecture, virtual DOM for efficient rendering, unidirectional data flow, and strong ecosystem (Redux, React Query, etc.) make it ideal for large scalable apps.

Q: Explain the concept of virtual DOM.
A: The virtual DOM is a lightweight in-memory representation of the actual DOM. React updates only the changed parts, resulting in faster UI rendering and performance.

Q: Difference between functional and class components.
A: Functional components are stateless and use hooks for state and lifecycle, while class components use lifecycle methods and 'this' context.

Q: How do Hooks improve performance?
A: Hooks like useMemo and useCallback prevent unnecessary re-renders by memoizing data or functions.

Q: How to manage complex state across modules?
A: Use Redux or Context API with reducers to manage global state. Redux Toolkit simplifies boilerplate and supports middleware.

Integration & APIs

Q: How do you integrate RESTful APIs in React?
A: Use fetch or axios inside useEffect hooks. Handle loading and error states with local or global state.

Q: How to handle authentication and authorization?
A: Store tokens in HttpOnly cookies or secure storage and use protected routes with React Router.

Q: How to manage protected routes?
A: Use with conditional rendering or higher-order components to check authentication state.

Q: How to handle API errors gracefully?
A: Show toast or alert messages, fallback UI, and retry logic where appropriate.

AI Integration

Q: How to integrate AI-driven features in the frontend?
A: Consume AI APIs from backend and visualize data using charts, insights, or suggestions dynamically.

Q: What libraries for visualization?
A: Use Recharts, Chart.js, or D3.js for interactive, responsive data visualization dashboards.

UI/UX & Accessibility

Q: How to ensure responsive design?

A: Use CSS frameworks like Tailwind or Bootstrap with media queries and Flex/Grid layout.

Q: How to ensure accessibility?
A: Follow WCAG guidelines, use semantic HTML, ARIA attributes, and keyboard navigation support.

Collaboration & Practices

Q: How to maintain code quality?
A: Use ESLint, Prettier, code reviews, and follow modular structure and reusable components.

Q: How to test React apps?
A: Use Jest and React Testing Library for unit/integration tests, Cypress for E2E testing.

Enterprise Focus

Q: How to design modular front-end architecture?
A: Organize by feature modules; each module has its own components, reducers, and services.

Q: How to handle role-based UI access?
A: Implement role-based routing and conditional rendering based on user role from JWT or state.

Q: How to implement PWA features?
A: Use service workers, manifest.json, and offline caching with Workbox.

Behavioral Questions

Q: Describe a time you optimized a React app's performance.
A: Example: Reduced render lag by implementing lazy loading, memoization, and pagination for large data tables.

Q: How to handle backend API inconsistency?
A: Add validation on frontend, report errors to backend, and show fallback data to users.