🔍 Key Areas to Prepare (React + Ecosystem)

**✅ 1. Core React Concepts**

* JSX, Virtual DOM, Reconciliation
* Functional components vs Class components
* React Hooks: useState, useEffect, useMemo, useCallback, useRef, useContext, useReducer
* Component lifecycle (both class and hook equivalents)
* Controlled vs uncontrolled components

**✅ 2. Component Design & Architecture**

* Atomic design principles
* Lifting state up and prop drilling
* Composition vs inheritance
* Higher-Order Components (HOCs)
* Render props

**✅ 3. State Management**

* Local state vs global state
* Redux (or Zustand, Jotai, Recoil)
  + Actions, Reducers, Middleware (Redux Thunk / Saga)
* Context API and comparison with Redux

**✅ 4. Performance Optimization**

* Memoization (React.memo, useMemo, useCallback)
* Lazy loading, Code splitting
* React Profiler
* Avoiding unnecessary re-renders
* List rendering optimizations (keys, React.memo, useMemo)

**✅ 5. Testing**

* Unit testing with Jest and React Testing Library
* Mocking props/services
* Snapshot testing
* Coverage reports

**✅ 6. Routing**

* React Router v6
  + Dynamic routes
  + Nested routes
  + Route guards (private/public routes)

**✅ 7. Forms**

* Controlled forms using native input handling
* Libraries: Formik / React Hook Form
* Validation using Yup or custom logic

**✅ 8. Styling**

* CSS Modules, Styled-components, Emotion
* TailwindCSS
* Responsive design, media queries

**✅ 9. Tooling & DevOps Awareness**

* Webpack, Babel basics
* Linting (ESLint), Prettier
* Git best practices
* Environment variables, .env setup

**✅ 10. TypeScript (if applicable)**

* Typing props and state
* Interfaces vs types
* Generics in functions/hooks
* Discriminated unions

**✅ 11. API Communication**

* Fetch / Axios
* Handling loading/error states
* Retry logic, caching (React Query, SWR)

**✅ 12. Accessibility and SEO**

* ARIA attributes
* Semantic HTML
* Meta tags and SSR awareness (Next.js)

**✅ 13. System Design / Architecture**

* Folder structure
* Code splitting strategies
* Modular and scalable component systems
* Micro frontends (if applicable)

**💻 Commonly Asked Coding Questions (React + DSA)**

**🔹 React-focused Coding Questions**

1. Build a custom useDebounce hook
2. Create an infinite scroll component
3. Build a modal/tooltip using portals
4. Tabs component with dynamic children
5. Build a type-ahead / autocomplete input
6. Render a nested comment structure (recursive)
7. Create a form wizard with dynamic steps
8. Build a pagination component with navigation
9. Render a list of items with filter/search/sort
10. Build a custom dropdown (with keyboard support)

**🔹 JavaScript & DSA Questions**

*These are commonly asked even in React roles to test problem-solving:*

1. **Array & String manipulation**
   * Reverse a string
   * Remove duplicates from an array
   * Flatten nested arrays
   * Group anagrams
   * Find first non-repeating character
2. **Data Structures**
   * Map vs Set vs Object
   * LRU cache implementation
   * Custom stack/queue with methods
3. **Sorting & Searching**
   * Custom sort functions
   * Binary search
   * Sorting objects by keys/values
4. **Recursion**
   * Factorial, Fibonacci
   * Tree traversal (used in rendering nested comments, menus, etc.)
5. **Promises & Async**
   * Create a custom Promise.all
   * Debounce and throttle functions
   * Retry failed fetch with exponential backoff

**🗣️ Behavioral + System Design Rounds**

* Be ready to explain:
  + A tough bug you solved
  + How you improved app performance
  + How you approached code refactoring
  + How you handled component reusability
  + How you mentor juniors or collaborate with backend/devops

**🧠 Tip: Use STAR for Scenario-Based Questions**

**S**ituation → **T**ask → **A**ction → **R**esult