React Cheat Sheet

Why choose React?

Component-Based Architecture, Virtual DOM, Rich Ecosystem, Declarative UI, Flexibility.

Virtual DOM and its benefits

Lightweight copy of real DOM, minimizes direct manipulation, improves performance.

JSX and its advantages

Combines HTML and JS, improves readability, enables powerful tooling.

Reconciliation

Process React uses to update DOM efficiently by comparing Virtual DOM.

State vs Props

State: local, mutable; Props: immutable, passed from parent.

Hooks: useState & useEffect

useState manages local state; useEffect handles side effects.

useMemo vs useCallback

useMemo memoizes values; useCallback memoizes functions.

Code Splitting

Breaks code into chunks for better performance using React.lazy and Suspense.

Accessibility in React

Use semantic HTML, ARIA attributes, keyboard navigation.

Unidirectional Data Flow

Data flows top-down, predictable state management.

Pure Components vs HOC

Pure Component prevents unnecessary re-renders; HOC enhances components.

Security: CSRF & XSRF

Use tokens, validate requests, avoid storing sensitive data in local storage.

useEffect Use Cases

Fetching data, subscriptions, DOM updates.

React Optimization Techniques

Memoization, code splitting, virtualization.

Lazy Loading

Loads components only when needed for better performance.

Class vs Functional Components

Class uses lifecycle methods; Functional uses hooks.

SEO in React

Use SSR or static generation, add meta tags, optimize crawlability.

React Router

Handles client-side routing for SPA navigation.

Context API

Provides global state without prop drilling, alternative to Redux.

Server-Side Rendering (SSR)

Renders React on server for better SEO and performance.

Error Boundaries

Catch runtime errors and prevent app crashes.