

## React Hooks

Hook	Purpose	Notes
useState	State management in functional components	Stores values, triggers re-render
useEffect	Side effects / lifecycle methods	Replace componentDidMount etc.
useContext	Access context values	For global state, e.g., theme or auth
useReducer	Complex state management	Alternative to Redux for local state
useRef	Access DOM elements or persist values	Does not trigger re-render
useMemo	Memoize expensive calculations	Avoid unnecessary re-renders
useCallback	Memoize functions	Useful for passing stable functions to children
useLayoutEffect	Runs after DOM updates, before painting	Rarely needed, advanced
useImperativeHandle	Customize exposed methods for refs	Advanced, rarely used
useDebugValue	For debugging custom hooks	Optional, used in dev tools
useTransition	React 18, for concurrent rendering	Optimizing transitions
useDeferredValue	React 18, defer expensive updates	Optimize large state updates
useId	React 18, generate unique IDs	Useful in forms or lists

useState, useEffect, useContext, useReducer, useRef, useMemo, useCallback

---

## Core React Topics for MERN

Topic	Why important
Functional components	Modern React standard
JSX	How HTML is written inside JS
Props	Passing data to child components
State (useState)	Local component state
Lifecycle (useEffect)	Fetching API data, subscriptions, cleanup
Event handling	Buttons, forms, clicks (onClick, onChange)
Forms & controlled components	Handling user input
Conditional rendering	Show/hide elements based on state
Lists & keys	Rendering arrays (map) efficiently
Context API	Global state without Redux
Hooks combinations	useEffect + fetch for REST APIs
Routing	react-router-dom for multi-page apps
Error boundaries	Catch JS errors in components (advanced)
Higher-order components / custom hooks	Reusable logic (intermediate)

---

## MERN-Specific Frontend Topics

Topic	Notes
Fetch API / Axios	Connecting frontend to backend APIs
JSON handling	Sending & receiving JSON
CORS handling	Enable React to call Express backend
Forms + validation	Input validation before sending to backend
State management patterns	Local state, lifting state, context
Environment variables	VITE_ prefixed env vars in Vite
Async operations	async/await for fetching data
Error handling	Show user-friendly error messages
React + MongoDB flow	State → fetch → backend → database

---