React Notes (Basic to Advanced) – Detailed Explanations with Tasks

1. JavaScript Fundamentals

React is built on JavaScript, so mastering JS is mandatory.

Key Concepts:

- let, var, const: Variables with different scoping rules.
- Arrow Functions: Cleaner syntax for functions.
- Array Methods: map, filter, reduce for data transformations.
- Spread & Rest: Useful for passing props and handling arrays/objects.
- Destructuring: Extract values cleanly.
- Promises & async/await: Handle asynchronous API calls.

Tasks:

- Create a custom map() function.
- Implement an async function that simulates a slow API.

2. React Basics

React uses components to build UI. JSX is the syntax extension allowing HTML-like code in JS.

Core Concepts:

- Components: Small reusable UI blocks.
- Props: Inputs to components (read-only).
- State: Mutable data inside components.
- Conditional Rendering: Display content based on conditions.
- Lists & Keys: Render lists efficiently.

Tasks:

- Build a counter component.
- Create a todo app with add/delete functionality.

3. Hooks (Most Important)

Hooks add state and logic to functional components.

Basic Hooks:

- useState: Manage state.
- useEffect: Run side effects like APIs, timers, subscriptions.
- useRef: Access DOM elements or store values without re-render.

Advanced Hooks:

- useMemo: Optimize expensive calculations.
- useCallback: Optimize functions passed as props.
- useReducer: Alternative to managing complex state.
- Custom Hooks: Reusable logic.

Tasks:
- Build a search filter using useEffect.
- Build a stopwatch using useRef.
- Create a custom hook useFetch(url).

4. State Management

React components manage local state, but large apps need global state.

Options:
- Prop Drilling: Passing props down multiple levels.
- Context API: Avoids prop drilling.
- Redux Toolkit: Modern, simplified global state management.

Tasks:
- Build a theme switcher using Context.
- Create a cart system using Redux Toolkit.

5. React Router

Used for navigation in Single Page Applications.

Important Features:
- BrowserRouter: Wraps the entire app.
- Route: Defines a page.
- useNavigate: Navigate programmatically.
- useParams: Read dynamic route parameters.
- Protected Routes: Restrict access.

Tasks:
- Create Home, Products, ProductDetails pages.
- Implement protected routes based on login.

6. API Integration

React apps frequently fetch data from servers.

Topics:

- Fetch & Axios
- Loading state
- Error handling
- Retry logic
- Pagination

Tasks:
- Build a product list using API.
- Add loading & error messages.
- Add pagination & search filters.

7. Performance Optimization

Large apps need optimization.

Techniques:
- React.memo: Prevent unnecessary renders.
- useMemo/useCallback: Cache calculations/functions.
- Lazy Loading: Load components only when needed.
- Code Splitting: Reduce bundle size.

Tasks:
- Optimize a list of expensive components.
- Convert a large page into lazy-loaded chunks.

8. Advanced React Concepts

These topics help in real-world, production-level apps.

Topics:
- Custom Hooks: Reusable logic patterns.
- HOC: Functions that take components and return enhanced components.
- Error Boundaries: Catch runtime errors.
- Portals: Render components outside root.
- Suspense: Handle async loading.

Tasks:
- Build an Error Boundary.
- Create a modal using Portals.
- Use Suspense with lazy loading.

9. Full Project Tasks

These help you become a real React expert.

Projects:

1. E-commerce Mini App

- Products, cart, filters, search.
- API Integration.
- Redux Toolkit.

2. Movie Search App

- Search API.
- Pagination.
- Debouncing.

3. Job Portal

- Authentication.
- CRUD (Add/Edit/Delete Jobs).
- Protected Routes.