React.js Notes: Complete Guide for Beginners to Advanced

Ashwini Kumar

September 2025

Build Awesome Apps with React!

Arre bhai, React.js ek mast library hai! Web apps banane ka maza hi alag hai isse. Jaise Lego ke blocks se ghar banate ho, waise hi React ke components se pura UI banao. Yeh fast hai, easy to learn, aur bilkul developer-friendly! Ek baar padho, sab samajh mein aa jayega. Chalo, dive karte hain aur React master karte hain!

Contents

1 Installation (Setup Kaise Karo)

1.1 Kya Hai?

Installation ka matlab hai React project ko setup karna taaki coding shuru ho sake. Create React App ek tool hai jo automatically React aur uske dependencies (jaise Babel, Webpack) install karta hai, aur ek ready-made project structure deta hai.

1.2 Kyu Use?

- Time save hota hai kyunki manual setup mein boilerplate code likhna padta. - Development server, hot reloading, aur build tools milte hain. - Beginners ke liye easy, kyunki sab pre-configured hota hai.

1.3 Kaise Kaam Karta Hai?

Node.js install karo, phir terminal mein `npx create-react-app my-app` run karo. Yeh command ek naya folder banayega with all necessary files (jaise `src`, `public`). `npm start` se app browser mein chalega at `localhost:3000`.

1.4 Real-Life Example

Ek to-do list app banana hai? Pehle setup karo:

1. Node.js install karo.
2. Terminal mein `npx create-react-app todo-app` run karo.
3. `cd todo-app` aur `npm start` karo.
4. Browser mein default React app dikhega!

```
npx create-react-app todo-app
cd todo-app
npm start
```

Setup Flow: Install → Run → Browser

Tip: Node.js latest version use karo, warna errors aa sakte hain.
Fun Fact: Create React App ko Facebook ne banaya tha!

2 Components (Building Blocks)

2.1 Kya Hai?

Components React ke dil hain! Yeh chhote, reusable code pieces hain jo UI ka ek hissa banate hain, jaise button, header, ya form. Do types hote hain: Functional (modern) aur Class (old-school).

2.2 Kyu Use?

- Code modular hota hai, easy to maintain. - Reuse kar sakte ho multiple jagah. - Ek component ek specific kaam karta hai, toh debugging simple hoti hai.

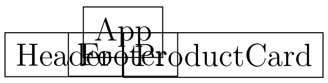
2.3 Kaise Kaam Karta Hai?

- **Functional Components**: Simple functions jo JSX return karte hain. Modern aur hooks ke saath powerful. - **Class Components**: `React.Component` extend karte hain, state aur lifecycle methods ke liye purana tareeka. - JSX use karke HTML-like syntax likho, jo React DOM mein convert karta hai.

2.4 Real-Life Example

Ek shopping app mein ek `ProductCard` component banao jo product ka name, price, aur image show kare.

```
function ProductCard() {
  return (
    <div>
      <h2>iPhone 13</h2>
      <p>Price: $799</p>
      <img src="iphone.jpg" alt="iPhone" />
    </div>
  );
}
```



Component Tree: Jaise Ek App ka Blueprint!

Motivational Quote: “Chhote components, bade sapne!” - Ashwini Kumar

3 JSX (JavaScript XML)

3.1 Kya Hai?

JSX ek special syntax hai jo HTML jaisa dikhta hai lekin JavaScript ke saath kaam karta hai. Yeh React mein UI define karne ka tareeka hai.

3.2 Kyu Use?

- Code readable hota hai kyunki HTML-like syntax familiar hai. - JavaScript ke saath mix karke dynamic UI bana sakte ho. - Errors early catch hote hain compile time pe.

3.3 Kaise Kaam Karta Hai?

JSX ko Babel transpile karta hai `React.createElement` calls mein. Har JSX element ek JavaScript object ban jata hai jo React DOM mein render hota hai. Example: `<h1>Hello</h1>` becomes `React.createElement('h1', null, 'Hello')`.

3.4 Real-Life Example

Ek user greeting: Username dynamically show karo.

```
const name = 'Ashwini';
const element = <h1>Hello, {name}!</h1>;
```

Tip: Curly braces `{}` mein JavaScript expressions likho, lekin objects ya statements nahi. Common mistake: `{if(true)}` likhna.

4 Props (Properties Pass Karo)

4.1 Kya Hai?

Props (properties) ek component se doosre component mein data pass karne ka tareeka hain. Yeh read-only hote hain aur parent se child ko milte hain.

4.2 Kyu Use?

- Components ko dynamic banata hai, jaise ek button ka text change karna. - Reusability badhata hai kyunki same component alag-alag data ke saath use hota hai. - Code organized rehta hai.

4.3 Kaise Kaam Karta Hai?

Props ko attributes ki tarah pass karo: `<Child name="Ashwini" />`. Child component mein `props` object se access karo. Functional components mein `props` parameter hota hai.

4.4 Real-Life Example

Ek e-commerce app mein product details pass karo:

```
function ProductCard(props) {
  return (
    <div>
      <h2>{props.name}</h2>
      <p>Price: ${props.price}</p>
    </div>
  );
}
// Usage
<ProductCard name="Laptop" price="999" />
```

Analogy: Props jaise gift box hain jo parent apne child components ko dete hain!

5 State (Local Data Store)

5.1 Kya Hai?

State ek component ka private data hota hai jo uski internal state manage karta hai. Jab state change hota hai, component re-render hota hai.

5.2 Kyu Use?

- Interactive UI ke liye, jaise form inputs ya counters. - User actions ke response mein UI update karne ke liye. - Temporary data store karne ke liye jo component ke andar hi chahiye.

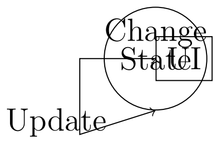
5.3 Kaise Kaam Karta Hai?

Functional components mein `useState` hook use hota hai. Yeh ek array return karta hai: current state aur usko update karne wala function. Example: `const [count, setCount] = useState(0);`.

5.4 Real-Life Example

Ek counter app: Button click pe number badhao.

```
import { useState } from 'react';
function Counter() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```



State Flow: Change → Update UI

Tip: State directly modify nahi karo, hamesha `setState` use karo.

6 Lifecycle Methods (Class Components Ke Liye)

6.1 Kya Hai?

Lifecycle methods class components ke special functions hain jo component ke banne, update hone, ya khatam hone ke time call hote hain. Inka use side effects (jaise API calls) ke liye hota hai.

6.2 Kyu Use?

- Initial setup (jaise data fetch) ke liye. - Cleanup (jaise timers clear) ke liye. - Component ke behavior ko control karne ke liye.

6.3 Kaise Kaam Karta Hai?

- `componentDidMount`: Component DOM mein add hone ke baad call hota hai. - `componentDidUpdate`: Props ya state change hone pe. - `componentWillUnmount`: Component remove hone se pehle.

6.4 Real-Life Example

Ek timer app jo seconds count karta hai:

```
class Timer extends React.Component {
  state = { seconds: 0 };
  componentDidMount() {
    this.timerID = setInterval(() => this.setState({ seconds: this.state.
   seconds + 1 }), 1000);
  }
  componentWillUnmount() {
    clearInterval(this.timerID);
  }
  render() {
    return <p>Seconds: {this.state.seconds}</p>;
  }
}
```

Note: Modern React mein hooks (`useEffect`) prefer karo. Common mistake: Cleanup bhoolna, memory leaks ho sakte hain.

7 Hooks (Modern Magic)

7.1 Kya Hai?

Hooks special functions hain jo functional components mein state aur lifecycle features add karte hain. Yeh React 16.8 se aaye aur classes ki jagah le rahe hain.

7.2 Kyu Use?

- Code simpler aur chhota hota hai. - Classes ke complexity (this binding) se bachao. - Reusable logic banane mein madad.

7.3 Kaise Kaam Karta Hai?

- `useState`: Local state manage karta hai. - `useEffect`: Side effects (jaise API calls) ke liye. - `useContext`, `useReducer`, `useMemo`, etc., specific kaam ke liye.

7.4 Real-Life Example

Ek app jo API se user data fetch karta hai:

```
import { useState, useEffect } from 'react';
function UserList() {
  const [users, setUsers] = useState([]);
  useEffect(() => {
    fetch('https://api.example.com/users')
      .then(res => res.json())
      .then(data => setUsers(data));
  }, []); // Empty array: Mount pe ek baar
  return <ul>{users.map(user => <li key={user.id}>{user.name}</li>)}</ul>;
}
```

Hook	Description
useState	Component mein local state add karta hai
useEffect	Side effects (API calls, timers) handle karta hai
useContext	Context ka data access karta hai
useReducer	Complex state logic ke liye Redux jaisa
useMemo	Expensive calculations ko cache karta hai
useCallback	Functions ko memoize karta hai

Table 1: Hooks Overview

Analogy: Hooks jaise superhero powers hain jo functional components ko supercharged banate hain!

8 Conditional Rendering

8.1 Kya Hai?

Conditional rendering ka matlab hai UI ka koi hissa condition ke basis pe show ya hide karna, jaise if-else logic.

8.2 Kyu Use?

- Dynamic UI banane ke liye, jaise login ke baad dashboard show. - User experience improve hota hai. - Code flexible banta hai.

8.3 Kaise Kaam Karta Hai?

Ternary operators (`condition ? <True/> : <False/>`) ya logical AND (`&&`) use karo. JSX mein conditions directly likh sakte ho.

8.4 Real-Life Example

Ek login system: User logged in hai toh dashboard, nahi toh login form.

```
function App() {
  const isLoggedIn = true;
  return (
    <div>
      {isLoggedIn ? <Dashboard /> : <LoginForm />}
    </div>
  );
}
```

Tip: Complex conditions ke liye separate function banao.

9 Lists and Keys

9.1 Kya Hai?

Lists mein arrays ko render karna hota hai, aur `key` prop uniquely identify karta hai har item ko.

9.2 Kyu Use?

- Efficient rendering, React ko pata hota hai kya change hua. - Dynamic data (jaise todo items) display karne ke liye. - Reordering ya deletion smooth hota hai.

9.3 Kaise Kaam Karta Hai?

`map()` function se array iterate karo, har element ko unique `key` do (usually ID).

9.4 Real-Life Example

Todo list app mein tasks render karo:

```
function TodoList({ todos }) {
  return (
    <ul>
      {todos.map(todo => (
        <li key={todo.id}>{todo.text}</li>
      ))}
    </ul>
  );
}
```

Tip: Index as key avoid karo if list reorder hoti hai.

10 Forms (User Input Handle)

10.1 Kya Hai?

Forms mein user input ko manage karna, usually `useState` ke saath controlled components banake.

10.2 Kyu Use?

- Input validation aur error handling. - Form submission ko control karna. - Dynamic UI updates (jaise live preview).

10.3 Kaise Kaam Karta Hai?

`value` attribute se input state se bind hota hai, aur `onChange` se state update hota hai.

10.4 Real-Life Example

Ek simple login form:

```
function LoginForm() {
  const [email, setEmail] = useState('');
  return (
    <form>
      <input
        type="email"
        value={email}
        onChange={e => setEmail(e.target.value)}
        placeholder="Enter email"
      />
    </form>
  );
}
```

Analogy: Forms jaise diary hain jahan user apne thoughts likhta hai!

11 Lifting State Up

11.1 Kya Hai?

Jab do ya zyada components ko same state chahiye, toh state ko unke common parent mein move karo.

11.2 Kyu Use?

- Siblings ke beech data share karna. - State ko centralized rakhta hai. - Code maintainable banta hai.

11.3 Kaise Kaam Karta Hai?

Parent mein state rakho, props se children ko pass karo, aur callbacks se parent state update karo.

11.4 Real-Life Example

Do inputs jo ek doosre se sync hon (jaise currency converter):

```
function Converter() {
  const [amount, setAmount] = useState(0);
  return (
    <div>
      <input value={amount} onChange={e => setAmount(e.target.value)} />
      <p>Converted: {amount * 2}</p>
    </div>
  );
}
```

12 Composition vs Inheritance

12.1 Kya Hai?

Composition ka matlab hai components ko nest karke UI banana. Inheritance (class extend) React mein rare hai.

12.2 Kyu Use?

- Composition flexible aur reusable hai. - Inheritance se code complex ho jata hai. - React ka design composition pe based hai.

12.3 Kaise Kaam Karta Hai?

Components ko `children` prop ya props ke through combine karo.

12.4 Real-Life Example

Ek modal component:

```
function Modal({ children }) {
  return <div className="modal">{children}</div>;
}
<Modal><h1>Hello</h1></Modal>
```

13 Context API (Global Data Share)

13.1 Kya Hai?

Context API se data globally share karo bina props drilling ke, jaise theme ya user info.

13.2 Kyu Use?

- Deep nested components mein data pass karna easy. - Boilerplate code kam hota hai. - Global state manage karne ke liye.

13.3 Kaise Kaam Karta Hai?

`createContext` se context banao, `Provider` se value set karo, aur `useContext` se consume karo.

13.4 Real-Life Example

Theme toggle app:

```
const ThemeContext = createContext('light');
function App() {
  return (
    <ThemeContext.Provider value="dark">
      <Child />
    </ThemeContext.Provider>
  );
}
function Child() {
  const theme = useContext(ThemeContext);
  return <p>Theme: {theme}</p>;
}
```

Analogy: Context jaise family group chat hai, sabko ek hi message milta hai!

14 Refs (Direct DOM Access)

14.1 Kya Hai?

Refs se DOM elements ya component instances ko directly access karo, jaise input focus karna.

14.2 Kyu Use?

- DOM manipulation (jaise focus, scroll). - Animations ya third-party libraries ke liye. - State ke bina direct control.

14.3 Kaise Kaam Karta Hai?

`useRef` hook ek mutable object return karta hai jiska `.current` property access deta hai.

14.4 Real-Life Example

Input focus on button click:

```
function TextInput() {
  const inputRef = useRef(null);
  return (
    <div>
      <input ref={inputRef} />
      <button onClick={() => inputRef.current.focus()}>Focus</button>
    </div>
  );
}
```

15 Portals (Outside DOM Render)

15.1 Kya Hai?

Portals se React components ko DOM ke alag node mein render karo, jaise modals ke liye.

15.2 Kyu Use?

- Root DOM se bahar render karna (jaise modals, tooltips). - Z-index ya styling conflicts avoid karna. - Cleaner DOM structure.

15.3 Kaise Kaam Karta Hai?

`ReactDOM.createPortal(child, domNode)` use karo.

15.4 Real-Life Example

Modal popup:

```
function Modal() {
  return createPortal(
    <div className="modal">Hello</div>,
    document.getElementById('modal-root')
  );
}
```

16 Error Boundaries

16.1 Kya Hai?

Error boundaries components hain jo JavaScript errors catch karte hain aur fallback UI dikhate hain.

16.2 Kyu Use?

- App crash prevent karna. - User ko better experience dena. - Debugging easy hota hai.

16.3 Kaise Kaam Karta Hai?

Class components mein `componentDidCatch` aur `static getDerivedStateFromError` use karo.

16.4 Real-Life Example

Error fallback:

```
class ErrorBoundary extends React.Component {
  state = { hasError: false };
  static getDerivedStateFromError() {
    return { hasError: true };
  }
  render() {
    if (this.state.hasError) return <h1>Something went wrong!</h1>;
    return this.props.children;
  }
}
```

17 Higher-Order Components (HOC)

17.1 Kya Hai?

HOC ek function hai jo component leta hai aur enhanced component return karta hai.

17.2 Kyu Use?

- Code reuse (jaise authentication logic). - Cross-cutting concerns (logging, analytics). - Component logic ko separate karna.

17.3 Kaise Kaam Karta Hai?

Ek function banao jo component wrap kare aur extra props ya logic add kare.

17.4 Real-Life Example

Auth HOC:

```
function withAuth(Component) {
  return function WrappedComponent(props) {
    const isAuthenticated = checkAuth();
    return <Component {...props} isAuthenticated={isAuthenticated} />;
  };
}
```

18 Render Props

18.1 Kya Hai?

Render prop ek prop hai jo function hota hai aur rendering logic provide karta hai.

18.2 Kyu Use?

- Shared logic ke liye, jaise mouse position tracking. - Flexible aur reusable code. - HOC ka alternative.

18.3 Kaise Kaam Karta Hai?

Component ek function prop accept karta hai jo UI render karta hai.

18.4 Real-Life Example

Mouse tracker:

```
function MouseTracker({ render }) {
  const [position, setPosition] = useState({ x: 0, y: 0 });
  return <div onMouseMove={e => setPosition({ x: e.clientX, y: e.clientY })}>
    {render(position)}
  </div>;
}
```

19 Routing with React Router

19.1 Kya Hai?

React Router ek library hai jo single-page apps mein client-side routing handle karta hai, jaise page navigation.

19.2 Kyu Use?

- URL-based navigation without page reload. - Dynamic routes aur nested routes support. - User-friendly app navigation.

19.3 Kaise Kaam Karta Hai?

`BrowserRouter`, `Routes`, aur `Route` components use karo. React Router v7 React 19 ke saath better types aur performance deta hai.

19.4 Real-Life Example

Ek blog app mein Home, About, aur Post pages:

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/post/:id" element={<Post />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Tip: Dynamic routes ke liye `:id` use karo.