# Chapter 13: Best Practices and Optimization

## 1 Introduction

Is chapter mein humne React apps ke liye **best practices** aur **optimization techniques** seekhe, jaise `useMemo`, `useCallback`, lazy loading, aur error boundaries. Yeh performance aur maintainability improve karte hain.

## 2 13.1 React Best Practices

- **Component Structure**: Chhote, reusable components.
- **Naming Conventions**: Meaningful names, e.g., `CartItem`.
- **Props Destructuring**: Readability ke liye.

```
function CartItem({ name }) {
  return <div>{name}</div>;
}
```

- **Minimal State**: Derived data compute karo.
- **File Organization**: Components, hooks, contexts alag folders mein.

### 2.1 Real-World Example

Food delivery app mein `Header`, `MenuItem`, `CartContext` alag rakho for clean structure.

## 3 13.2 Performance Optimization with useMemo and useCallback

`useMemo` values memoize karta hai, `useCallback` functions.

### 3.1 Code Example: Optimizing with useMemo and useCallback

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Optimization with useMemo and useCallback</title>
  <script
     src="https://cdn.jsdelivr.net/npm/react@18.2.0/umd/react.development.js
  <script
     src="https://cdn.jsdelivr.net/npm/react-dom@18.2.0/umd/react-dom.develo
  <script
     src="https://cdn.jsdelivr.net/npm/@babel/standalone@7.20.6/babel.min.js
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div id="root"></div>
  <script type="text/babel">
```

```
    // CartItem Component
    function CartItem({ item, removeItem }) {
      console.log(`Rendering CartItem: ${item.name}`);
      return (
        <li className="text-lg">
          ${item.name} - Rs ${item.price}
          <button
            className="ml-2 text-red-500"
            onClick={() => removeItem(item.id)}
          >
            Remove
          </button>
        </li>
      );
    }

    // Memoized CartItem
    const MemoizedCartItem = React.memo(CartItem);

    // Cart Component
    function Cart() {
      const [items, setItems] = React.useState([
        { id: 1, name: "Chai", price: 10 },
        { id: 2, name: "Samosa", price: 20 }
      ]);
      const [counter, setCounter] = React.useState(0);

      // useMemo for total price
      const totalPrice = React.useMemo(() => {
        console.log("Calculating total price");
        return items.reduce((sum, item) => sum + item.price, 0);
      }, [items]);

      // useCallback for removeItem
      const removeItem = React.useCallback((id) => {
        setItems((prevItems) => prevItems.filter((item) =>
          item.id !== id));
      }, []);

      return (
        <div className="text-center p-4">
          <h1 className="text-3xl font-bold
            text-blue-600">Cart</h1>
          <button
            className="bg-blue-500 text-white px-4 py-2 m-2
              rounded"
            onClick={() => setCounter(counter + 1)}
          >
            Counter: ${counter}
          </button>
```

```
            <h2 className="text-xl font-bold">Total: Rs
               ${totalPrice}</h2>
            <ul className="list-disc list-inside">
              {items.map((item) => (
                <MemoizedCartItem
                  key={item.id}
                  item={item}
                  removeItem={removeItem}
                />
              ))}
            </ul>
          </div>
        );
      }

      const root =
         ReactDOM.createRoot(document.getElementById('root'));
      root.render(<Cart />);
    </script>
</body>
</html>
```

### 3.2 Output

Browser mein yeh dikhega:

- Heading: "Cart" (large, bold, blue).
- Button: "Counter: 0" (blue, increments on click).
- Subheading: "Total: Rs 30".
- List: "Chai - Rs 10 [Remove]", "Samosa - Rs 20 [Remove]" (red buttons).
- **Behavior**: Counter click pe total nahi recalculate hota, Remove pe item delete aur total update.

### 3.3 Explanation

- `useMemo`: `totalPrice` jab `items` change ho.
- `useCallback`: `removeItem` memoized.
- `React.memo`: `CartItem` ko unnecessary renders se bachata.

## 4 13.3 Lazy Loading and Code Splitting

`React.lazy` aur `Suspense` se components dynamically load hote hain.

### 4.1 Code Example: Lazy Loading a Component

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <title>Lazy Loading</title>
  <script
     src="https://cdn.jsdelivr.net/npm/react@18.2.0/umd/react.development.js
  <script
     src="https://cdn.jsdelivr.net/npm/react-dom@18.2.0/umd/react-dom.develo
  <script
     src="https://cdn.jsdelivr.net/npm/@babel/standalone@7.20.6/babel.min.js
  <script
     src="https://cdn.jsdelivr.net/npm/react-router-dom@6.3.0/dist/umd/react
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div id="root"></div>
  <script type="text/babel">
    const { BrowserRouter, Routes, Route, Link } =
       ReactRouterDOM;

    // Lazy-loaded Menu Component
    const Menu = React.lazy(() => {
      return new Promise((resolve) => {
        setTimeout(() => {
          resolve({
            default: () => (
              <div className="text-center p-4">
                <h2 className="text-xl font-bold
                   text-blue-600">Menu</h2>
                <ul className="list-disc list-inside">
                  <li>Chai - Rs 10</li>
                  <li>Samosa - Rs 20</li>
                </ul>
              </div>
            )
          });
        }, 1000);
      });
    });

    // Main App Component
    function App() {
      return (
        <BrowserRouter>
          <div className="p-4">
            <nav className="flex justify-center space-x-4">
              <Link to="/" className="text-blue-500
                 hover:underline">Home</Link>
              <Link to="/menu" className="text-blue-500
                 hover:underline">Menu</Link>
            </nav>
```

```
            <React.Suspense fallback={<div
               className="text-center p-4">Loading...</div>}>
              <Routes>
                <Route path="/" element={<h1
                   className="text-center text-3xl font-bold
                   text-blue-600 p-4">Home</h1>} />
                <Route path="/menu" element={<Menu />} />
              </Routes>
            </React.Suspense>
          </div>
        </BrowserRouter>
      );
    }

    const root =
       ReactDOM.createRoot(document.getElementById('root'));
    root.render(<App />);
  </script>
</body>
</html>
```

### 4.2 Output

Browser mein yeh dikhega:

- **Nav Bar**: "Home", "Menu" links (blue, hover pe underline).
- **URL: /**: Heading "Home".
- **URL: /menu**: Initially "Loading..." (1s), then heading "Menu", list: "Chai - Rs 10", "Samosa - Rs 20".

### 4.3 Explanation

- `React.lazy`: `Menu` dynamically load.
- `Suspense`: Loading UI.

## 5 13.4 Error Boundaries

Error boundaries errors catch karte hain, app crash se bachate hain.

### 5.1 Code Example: Error Boundary

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Error Boundary</title>
  <script
     src="https://cdn.jsdelivr.net/npm/react@18.2.0/umd/react.development.js
```

```
  <script
     src="https://cdn.jsdelivr.net/npm/react-dom@18.2.0/umd/react-dom.develo
  <script
     src="https://cdn.jsdelivr.net/npm/@babel/standalone@7.20.6/babel.min.js
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div id="root"></div>
  <script type="text/babel">
    // Error Boundary Component
    class ErrorBoundary extends React.Component {
      state = { hasError: false };

      static getDerivedStateFromError(error) {
        return { hasError: true };
      }

      render() {
        if (this.state.hasError) {
          return <div className="text-center p-4
             text-red-500">Something went wrong!</div>;
        }
        return this.props.children;
      }
    }

    // Buggy Component
    function BuggyComponent() {
      const [crash, setCrash] = React.useState(false);
      if (crash) {
        throw new Error("Crashed!");
      }
      return (
        <div className="text-center p-4">
          <h2 className="text-xl font-bold text-blue-600">Buggy
             Component</h2>
          <button
            className="bg-red-500 text-white px-4 py-2 m-2
               rounded"
            onClick={() => setCrash(true)}
          >
            Crash Me
          </button>
        </div>
      );
    }

    // Main App Component
    function App() {
      return (
        <div className="p-4">
```

```
53            <h1 className="text-3xl font-bold text-blue-600">Error
                 Boundary Demo</h1>
54            <ErrorBoundary>
55              <BuggyComponent />
56            </ErrorBoundary>
57          </div>
58        );
59      }
60
61      const root =
           ReactDOM.createRoot(document.getElementById('root'));
62      root.render(<App />);
63    </script>
64  </body>
65  </html>
```

### 5.2 Output

Browser mein yeh dikhega:

- Heading: "Error Boundary Demo".
- Subheading: "Buggy Component".
- Button: "Crash Me" (red).
- **Behavior**: Button click pe "Something went wrong!" (red text).

### 5.3 Explanation

- `ErrorBoundary`: Errors catch karta hai.
- `BuggyComponent`: Error throw karta hai.

## 6 Common Mistakes

- **No `React.memo`**: Unnecessary renders.
- **Overusing `useMemo`/`useCallback`**: Sirf jab zaroori ho.
- **No Error Boundaries**: App crash risk.
- **Messy Structure**: Components alag rakho.

## 7 Interview Tips

- **Optimization kaise karte hain?**
  - `useMemo`, `useCallback`, `React.memo`, lazy loading.
- **`useMemo` vs `useCallback`?**
  - `useMemo`: Values, `useCallback`: Functions.

- **Error boundaries kaise kaam karte hain?**
  - Class components mein errors catch.

# 8 Assignment: Practice Time

## 8.1 Task 1: Optimized Cart

- `CartApp` banao jisme cart items aur total.
- `useMemo` se total, `useCallback` se `addItem`, `removeItem`.
- `React.memo` se `CartItem`.
- Tailwind se style.

## 8.2 Task 2: Lazy-Loaded Pages

- `FoodApp` banao jisme `/home`, `/menu` routes.
- `Menu` ko `React.lazy` se load.
- `Suspense` se loading UI.
- Tailwind se style.

## 8.3 Task 3: Error Boundary with API

- `MenuApp` banao jo JSONPlaceholder se posts fetch.
- `ErrorBoundary` se API errors catch.
- Fallback UI.
- Tailwind se style.