16. Advanced Patterns in React

Last updated by | Subramanya Dixit | May 5, 2025 at 7:59 PM GMT+5:30

📘 Why Advanced Patterns Are Important

As your React applications grow in complexity, you'll need to use more sophisticated patterns to manage and share state, handle side effects, and organize your app. Advanced patterns allow you to write more scalable and maintainable React code.

🧠 Key Concepts

✅ Render Props

Render Props is a technique for sharing code between components using a function that returns a React element. This allows you to dynamically inject content into a component.

```
function MouseTracker({ render }) {
  const [position, setPosition] = useState({ x: 0, y: 0 });
  useEffect(() => {
    const updateMousePosition = (e) => {
      setPosition({ x: e.clientX, y: e.clientY });
    };
    window.addEventListener('mousemove', updateMousePosition);
    return () => window.removeEventListener('mousemove', updateMousePosition);
  }, []);

  return render(position);
}

function App() {
  return (
    <MouseTracker render={({ x, y }) => (
      <h1>The mouse position is ({x}, {y})</h1>
    )} />
  );
}
```

✅ Higher-Order Components (HOCs)

A Higher-Order Component (HOC) is a function that takes a component and returns a new component with added props or behavior. It's used to share logic across multiple components.

```
function withLoading(Component) {
  return function WithLoading({ isLoading, ...props }) {
    if (isLoading) {
      return <div>Loading...</div>;
    }
    return <Component {...props} />;
  };
}

const MyComponentWithLoading = withLoading(MyComponent);
```

✅ Compound Components

Compound Components allow multiple components to share state while maintaining a flexible structure. The parent component manages the state, and the child components consume that state via context.

```
const Tab = ({ children }) => (
  <div>
    {children}
  </div>
);

const TabList = ({ children }) => (
  <div>{children}</div>
);

const TabPanel = ({ children }) => (
  <div>{children}</div>
);

function Tabs({ children }) {
  const [selectedTab, setSelectedTab] = useState(0);

  return (
    <Tab>
      {React.Children.map(children, (child, index) =>
        React.cloneElement(child, {
          selectedTab,
          setSelectedTab: () => setSelectedTab(index),
        })
      )}
    </Tab>
  );
}
```

✅ Context API with Custom Hook

Custom hooks can be used in conjunction with the Context API to abstract state management and business logic.

```
const ThemeContext = React.createContext();

function useTheme() {
  return useContext(ThemeContext);
}

function ThemeProvider({ children }) {
  const [theme, setTheme] = useState('light');
  return (
    <ThemeContext.Provider value={{ theme, setTheme }}>
      {children}
    </ThemeContext.Provider>
  );
}

function ThemedComponent() {
  const { theme, setTheme } = useTheme();
  return (
    <div style={{ background: theme === 'light' ? 'white' : 'black' }}>
      <button onClick={() => setTheme(theme === 'light' ? 'dark' : 'light')}>
        Toggle Theme
      </button>
    </div>
  );
}
```

💡 Guidelines

- **Render Props** is useful when you need to share logic between components, but it can lead to prop drilling if overused.

- **Higher-Order Components (HOCs)** help to abstract shared behavior, but they can make code harder to debug if used excessively.
- **Compound Components** work well for scenarios where you need to manage state internally and allow for a flexible structure.
- Use **custom hooks** and **Context API** to avoid prop drilling and manage shared state in a more declarative way.

📝 Practice Exercises

1. Create a component that tracks mouse position using **render props**.
2. Implement a **Higher-Order Component (HOC)** that adds additional behavior to a component (like loading state).
3. Create a **Compound Component** that manages tabs and their content.
4. Refactor an app to use **Context API** with a **custom hook** for theme management.

❓ Quiz Questions

1. What is a render prop?
a) A function that returns a React element
✅ **b) A function that allows sharing code between components**
c) A prop that renders content based on state
d) A prop that is passed to the root component

---

2. What does a Higher-Order Component (HOC) do?
a) Adds new state to a component
b) Returns a component that extends another component's behavior
✅ **c) Adds shared behavior to a component**
d) Provides a wrapper for context

---

3. What is the purpose of Compound Components?
a) To allow multiple components to share state without context
b) To improve the component rendering speed
✅ **c) To allow multiple components to share state and flexibility**
d) To pass props from parent to child
