12. Error Handling & Error Boundaries

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

📘 Why Error Handling Is Important

In a React application, runtime errors can break the UI or cause unexpected behavior. Error boundaries provide a way to catch and handle these errors gracefully, ensuring that the app doesn't crash and can display fallback UI instead.

🧠 Key Concepts

✅ What Are Error Boundaries?

Error boundaries are React components that catch JavaScript errors anywhere in their child component tree, log those errors, and display a fallback UI.

```
class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false, error: null };
  }

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

  componentDidCatch(error, errorInfo) {
    console.log(error, errorInfo); // Log error details
  }

  render() {
    if (this.state.hasError) {
      return <h1>Something went wrong: {this.state.error.message}</h1>;
    }
    return this.props.children;
  }
}
```

✅ How Error Boundaries Work

- They catch errors during rendering, in lifecycle methods, and in constructors of the whole tree below them.
- If an error occurs, it falls back to the UI defined in the `ErrorBoundary` .

⚙️ Key Points

- **Static `getDerivedStateFromError`** : This method is used to update state when an error occurs.
- **`componentDidCatch`** : This lifecycle method allows you to log error information.
- You should use error boundaries in parts of the app where unexpected failures are likely (e.g., data fetching or dynamic components).

📊 Visual Overview

```
graph LR
  A[Component with Error] --> B[Error Boundary]
  B --> C[Catch Error]
  C --> D[Display Fallback UI]
  D --> E[Log Error Information]
```

💡 Guidelines

- Wrap components that could throw errors in error boundaries to ensure graceful error handling.
- Use fallback UI that can provide helpful information to the user when something goes wrong.
- Ensure that error boundaries are placed at higher levels to catch errors in deep child components.

📝 Practice Exercises

1. Create a basic error boundary and wrap it around a component that intentionally throws an error.
2. Build an error handling component that displays a retry button after an error occurs.
3. Integrate logging of errors with external services (like Sentry or LogRocket).

❓ Quiz Questions

1. What does an error boundary do in React?
a) It automatically fixes errors in the app
b) It prevents all errors from occurring
✅ **c) It catches errors in components and displays fallback UI**
d) It only catches network errors

2. Where should you place an error boundary?
a) Around the whole app to catch all errors
✅ **b) Around specific components that are prone to errors**
c) In the state of a component
d) In the context API

3. What is the method used to log errors inside an error boundary?
a) `componentDidMount`
b) `getDerivedStateFromError`
✅ **c) `componentDidCatch`**
d) `useEffect`