ERROR BOUNDARIES

Alphin K Sajan (287643)

Introduction

- Error boundaries are React components that catch JavaScript errors anywhere in their child component tree.
- They prevent the entire app from crashing and provide a fallback UI in case of errors.
- Error boundaries only work in class components, not in functional components.

When to use Error Boundaries

- Use error boundaries to handle rendering errors in parts of the UI.
- Ideal for isolating broken sections of the UI while leaving other parts functional.
- Typical use cases:
  - A component that renders data from external APIs.
  - A component that relies on third-party libraries.
  - Any section of the app that may potentially throw runtime errors.

Basic Structure

- **State**: Tracks whether an error occurred.
- **Lifecycle methods**:
  - getDerivedStateFromError: Updates state if an error occurs.
  - componentDidCatch: Catches errors and logs them.
- **Fallback UI**: Renders a custom fallback UI when an error is caught.

Key Methods in Error Boundaries

- **getDerivedStateFromError():**
  - A lifecycle method to update the state based on the error.
  - It ensures that the component re-renders with the fallback UI after an error.
  - Example:
    - ```
      static getDerivedStateFromError(): ErrorBoundaryState {
        return { hasError: true };
      }
      ```
      ```

Key Methods in Error Boundaries

- **componentDidCatch():**
  - Handles error logging and additional side effects like reporting.
  - Receives two arguments: the error and additional info about the error.
  - Example:
    - ```
      componentDidCatch(error: Error, info: React.ErrorInfo) {
        console.log('Error:', error, info);
      }
      ```
      ```

How to use Error Boundaries

- Wrap critical components inside the error boundary.
- Only catches errors inside the child component tree, not in event handlers or async code.
- This ensures that errors in MyComponent or any of its children don’t crash the entire app.
-

```
<ErrorBoundary>
  <MyComponent />
</ErrorBoundary>
```

Limitations

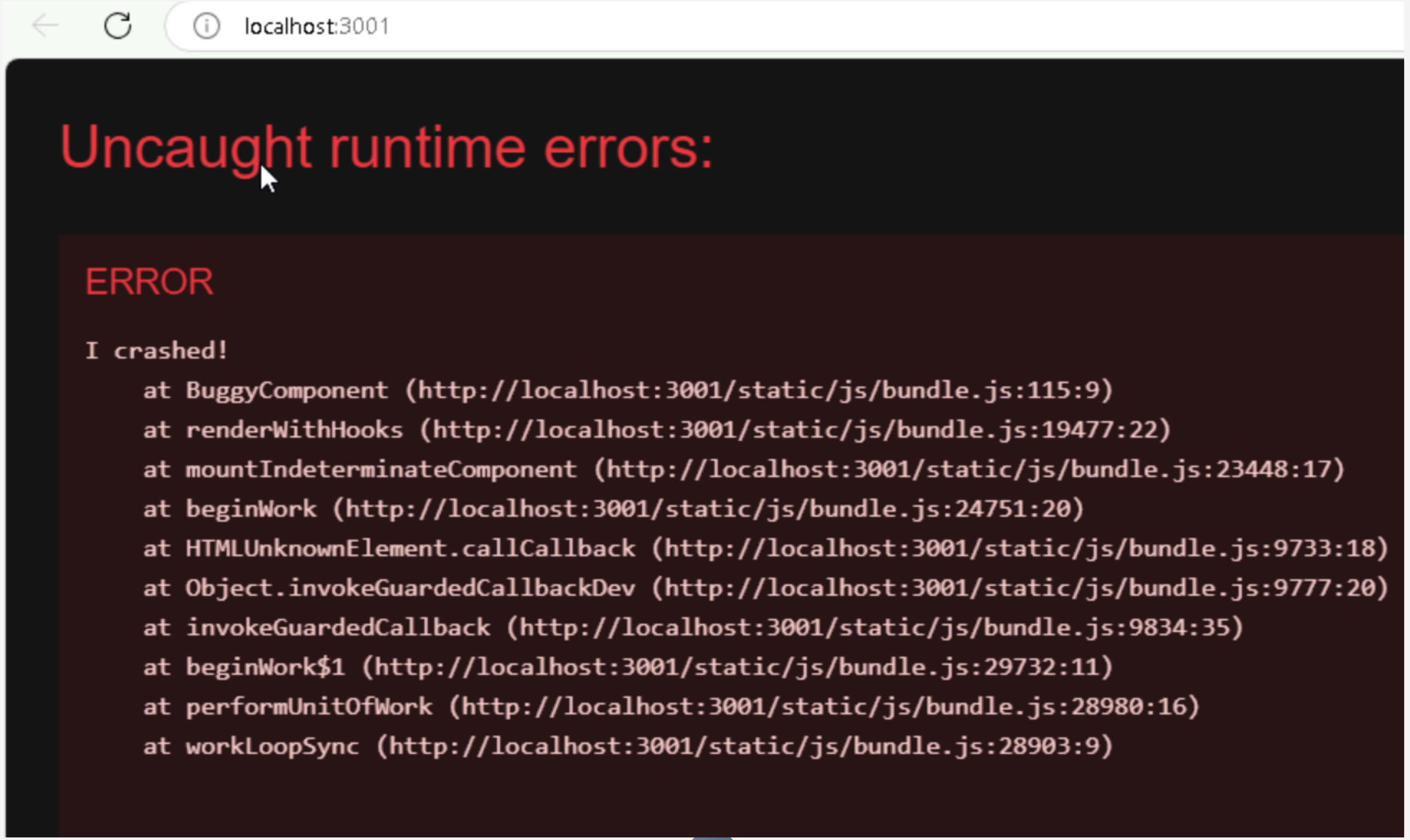
- Error boundaries don't catch errors in:
  - Event handlers (e.g., button clicks).
  - Asynchronous code (e.g., setTimeout, promises).
  - Server-side rendering.
  - Errors thrown inside the error boundary itself.

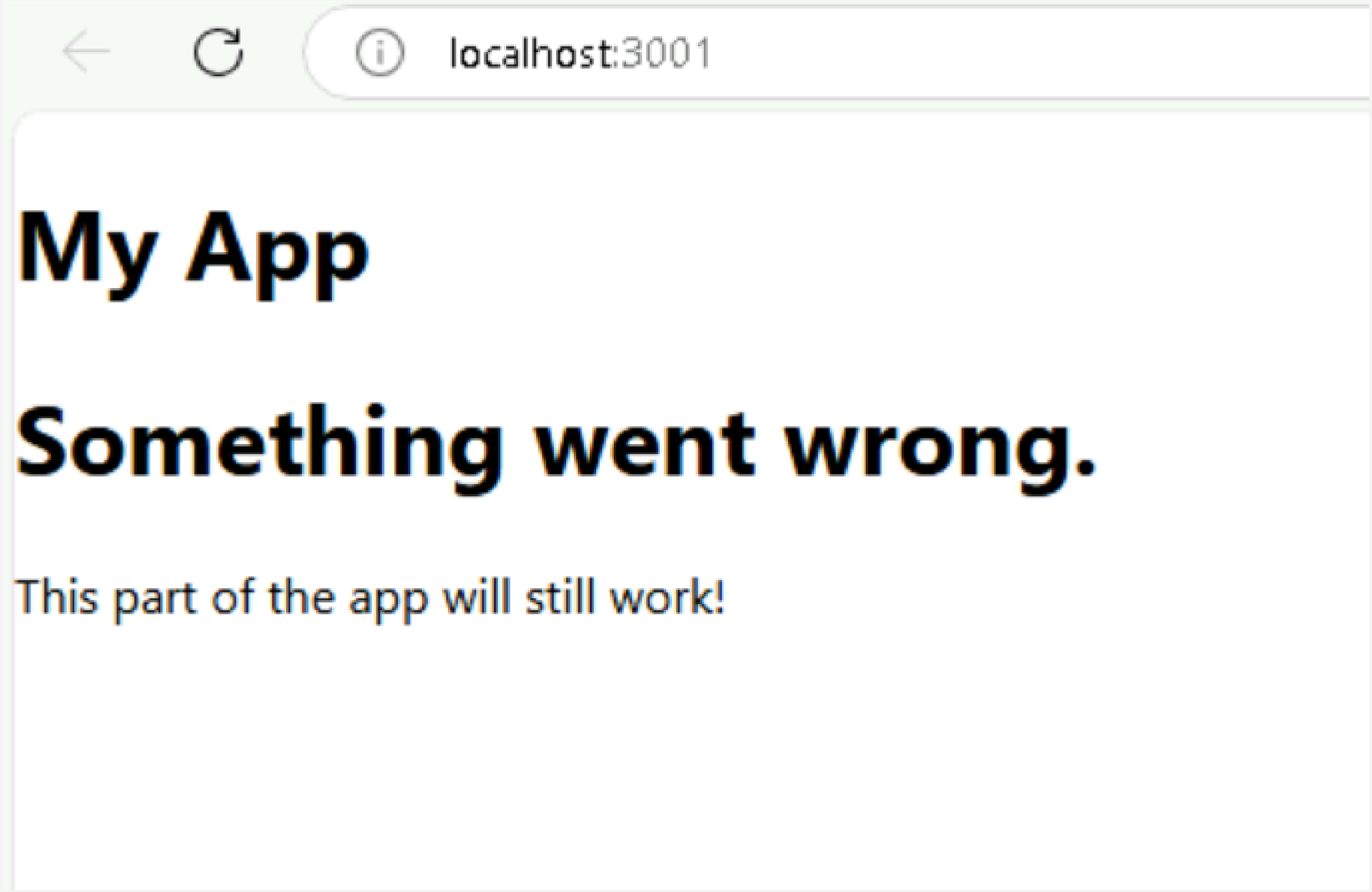
CODE SAMPLES

```tsx
// App.tsx
import React from 'react';
import ErrorBoundary from './ErrorBoundary';
import BuggyComponent from './BuggyComponent';

const App: React.FC = () => {
  return (
    <div>
      <h1>My App</h1>
      {/* <ErrorBoundary> */}
        <BuggyComponent />
      {/* </ErrorBoundary> */}
      <p>This part of the app will still work!</p>
    </div>
  );
};

export default App;
```

```tsx
// BuggyComponent.tsx
import React from 'react';

const BuggyComponent: React.FC = () => {
  throw new Error('I crashed!');
};

export default BuggyComponent;
```

```tsx
// ErrorBoundary.tsx
import React, { Component, ErrorInfo } from 'react';
interface Props {  children: React.ReactNode; }
interface State {  hasError: boolean; }
class ErrorBoundary extends Component<Props, State> {
  constructor(props: Props) {
    super(props);
    this.state = { hasError: false };
  }
  static getDerivedStateFromError(error: Error) {
     return { hasError: true };  // Update state so the next render shows the fallback UI
  }
  componentDidCatch(error: Error, errorInfo: ErrorInfo) {
     console.error("Error caught in ErrorBoundary:", error, errorInfo);
  } // Log the error to an error reporting service
  render() {
    if (this.state.hasError)
        return <h1>Something went wrong.</h1>; // Fallback UI
    return this.props.children;
  }
}
export default ErrorBoundary;
```


localhost:3001
Uncaught runtime errors:
ERROR
I crashed!
at BuggyComponent (http://localhost:3001/static/js/bundle.js:115:9)
at renderWithHooks (http://localhost:3001/static/js/bundle.js:19477:22)
at mountIndeterminateComponent (http://localhost:3001/static/js/bundle.js:23448:17)
at beginWork (http://localhost:3001/static/js/bundle.js:24751:20)
at HTMLUnknownElement.callCallback (http://localhost:3001/static/js/bundle.js:9733:18)
at Object.invokeGuardedCallbackDev (http://localhost:3001/static/js/bundle.js:9777:20)
at invokeGuardedCallback (http://localhost:3001/static/js/bundle.js:9834:35)
at beginWork$1 (http://localhost:3001/static/js/bundle.js:29732:11)
at performUnitOfWork (http://localhost:3001/static/js/bundle.js:28980:16)
at workLoopSync (http://localhost:3001/static/js/bundle.js:28903:9)


localhost:3001
My App
Something went wrong.
This part of the app will still work!

Conclusion

- Error boundaries catch rendering errors and provide fallback UI.
- Implemented as class components using TypeScript to ensure type safety.
- Key methods:
  - getDerivedStateFromError()
  - componentDidCatch()
- Limitations: Does not catch errors in event handlers, async code, or server-side rendering.
- Usage of TypeScript enhances the robustness of error handling.

THANK YOU