Errors

- Loss of revenue
- Unhappy customers
- Unhappy developers

Cost of Errors

- Bugs cost economy $312 billion per year
- Developers spend 50% of their programming time finding and fixing bugs

Source: Cambridge University

What is an Error?

- Incorrect result
- Unintended behaviour
- Inconsistent state

Error Handling

- Give a feedback to the users
- Discover errors quickly
- Store informations to reproduce problems

Types of Errors

- Unexpected
- Operational

Unexpected Errors

Challenges of Errors in React

- Uncaught errors result in unmounting of the whole React component tree.
- Errors don’t bubble up in components. (Why not try...catch)
- Try...catch suits imperative code, react components are declarative.

Error Boundaries

- Catch JavaScript errors
- Log these errors
- Display Fallback UI

Lifecycle methods

```
componentDidCatch(error, info)
```

```
static getDerivedStateFromError(error)
```

```
componentDidCatch(error, info)
```

- Invoked after an error has been thrown
- Side-effects are permitted
- Should be used for logging errors

Example :-

```
componentDidCatch(error, info) {
  // Catch and Log errors thrown by Child Components
  logErrorToMyBackendService(error, info);
}
```

```
static getDerivedStateFromError(error)
```

- Invoked after an error has been thrown
- Should return a value to update state

```jsx
class SampleErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { error: false };
  }

  static getDerivedStateFromError(error) {
    return { error: true }; // Update state to show the fallback UI on next render.
  }

  componentDidCatch(error, info) {
    logErrorToMyService(error, info);
  }

  render() {
    if (this.state.error) {
      // Fallback UI
      return <h1>Something went wrong.</h1>;
    }

    return this.props.children;
  }
}
```

1

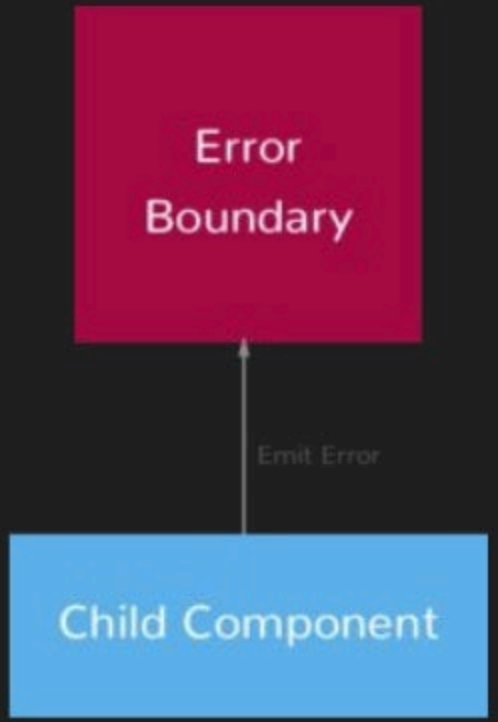
2

3

Used as a regular component

```
<SampleErrorBoundary>
  <Child1 />
  <Child2 />
</SampleErrorBoundary>
```

- Error boundaries catch errors anywhere in their child component tree
- They catch errors during rendering, in lifecycle methods, and in constructors of the whole tree below them. (not in event handlers, async code etc)



Error Boundaries

Best way to use error boundaries is to make a reusable component.

```
const myErrorHandler = (error: Error, componentStack: string) => {
  // Do something with the error
  // E.g. log to an error logging client here
};

const MyFallbackComponent = ({ componentStack, error }) => (
  <div>
    <p><strong>Oops! An error occured!</strong></p>
    <p>Here's what we know…</p>
    <p><strong>Error:</strong> {error.toString()}</p>
    <p><strong>Stacktrace:</strong> {componentStack}</p>
  </div>
);

<ErrorBoundary onError={myErrorHandler} FallbackComponent={MyFallbackComponent} >
  <ComponentThatMayError />
</ErrorBoundary>
```

Demo

codepen link

10.7k

r/UpliftingNews · Posted by u/emitremmus27 7 hours ago

Tennessee becomes first state in the South with hate crime law protecting transgender people

tennessean.com/story/...

1.2k Comments Share Save Give Award ...

27.0k

r/HumansBeingBros · Posted by u/smunozmo 10 hours ago

A mute person communicating with a blind and deaf person

Home

Your personal Reddit frontpage. Come here to check in with your favorite communities.

CREATE POST

CREATE COMMUNITY

Reddit Premium
Ads-free browsing

GET PREMIUM

TRENDING COMMUNITIES

r/shittyaquariums
13,889 subscribers
SUBSCRIBE

r/catfruit
16,635 subscribers
SUBSCRIBE

r/HollowKnight
50,056 subscribers
SUBSCRIBE

r/patientgamers
253,260 subscribers
SUBSCRIBE

r/CatsPlayingDnd

Error Boundaries

- Error Boundaries do not catch Errors thrown in Event handlers, Async Code
- As these don’t occur during rendering cycle
- Use try...catch for event handlers

Operational errors

Global Errors

- Generics Errors which come back from backend
- Like user is not signed in
- Should be stored in outermost stateful component
- Design implementation can be anything

Something went wrong... ×

Handling Errors

Donec ullamcorper nulla non metus auctor fringilla. Donec id elit non mi porta gravida at eget metus. Nulla vitae elit libero, a pharetra augue. Sed posuere consectetur est at lobortis. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Donec sed odio dui. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas faucibus mollis interdum. Vestibulum id ligula porta felis euismod semper. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Praesent commodo cursus magna, vel scelerisque nisl consectetur et.

```
class Application extends Component {
  constructor(props) {
    super(props)
    this.state = {
      error: '',
    }
    this._resetError = this._resetError.bind(this)
    this._setError = this._setError.bind(this)
  }

  render() {
    return (
      <div className="container">
        <GlobalError error={this.state.error} resetError={this._resetError} />
        <h1>Handling Errors</h1>
      </div>
    )
  }
  _resetError() {
    this.setState({ error: '' })
  }
  _setError(newError) {
    this.setState({ error: newError })
  }
}
```

Just pass setError down from Application.js to set a global error

```
_callBackend() {
  axios
    .post('/api/city')
    .then(result => {
      // do something with it, if the request is successful
    })
    .catch(err => {
      if (err.response.data.error === 'GENERIC') {
        this.props.setError(err.response.data.description)
      }
    })
}
```

Bonus

404 Pages

```
render() {
  <Router>
    <div>
      <ul>
        <li><Link to="/">Home</Link></li>
        <li><Link to="/will-match">Will Match</Link></li>
        <li><Link to="/will-not-match">Will Not Match</Link></li>
        <li><Link to="/also/will/not/match">Also Will Not Match</Link></li>
      </ul>

      <Route path="/" exact component={Home} />
      <Route path="/will-match" component={WillMatch} />
    </div>
  </Router>
}
```

404 Pages

```
render() {
  <Router>
    <div>
      <ul>
        <li><Link to="/">Home</Link></li>
        <li><Link to="/will-match">Will Match</Link></li>
        <li><Link to="/will-not-match">Will Not Match</Link></li>
        <li><Link to="/also/will/not/match">Also Will Not Match</Link></li>
      </ul>

      <Switch>
        <Route path="/" exact component={Home} />
        <Route path="/will-match" component={WillMatch} />
        <Route component={NoMatch} />
      </Switch>
    </div>
  </Router>
}
```