**Exception Handing in ReactJS**

React 16 has introduced a new error / exception handling concept called Error Boundary that catch JS errors in their child component tree, log the error and display the fallback UI instead of crashed the tree components. A component becomes an error boundary when it defines componentDidCatch(error, info) and it’s reusable through the app. The following is the template of different child components that ‘sheltered’ under the <ErrorBoundary> tag in the parent component.

Property File:

errorID:

errorMessage:

errorDescription:

interestedParty:

cause:

location:

Component\_n

Component\_1

Error Boundary

Component\_2

App.js  
  
**export default class** App **extends** Component {  
 render () {  
 **return** (  
 <div className='App'>  
 <h6>Parent</h6>  
 <ErrorBoundary>  
 <Child />  
 <Header />  
 <CallinComponent />  
 </ErrorBoundary>  
 </div>  
 )  
 }  
}

**ErrorBoundary.js**

**export default class** ErrorBoundary **extends** Component {  
 constructor (props) {  
 **super**(props)  
 **this**.state = { hasError: **false** }  
 }  
 componentDidCatch (error, info) {  
 **this**.setState({ hasError: **true** })  
 }  
 render () {  
 **if** (**this**.state.hasError) {  
 **return**(  
 <div>  
 <HashRouter>  
 <div>  
 <h1>Something went wrong</h1>  
 <ul>  
 <NavLink to="/home">Home</NavLink>  
 <Route exact path='/home' component={Home}/>  
 </ul>  
 <div>  
 </div>  
 </div>  
 </HashRouter>  
 </div>  
 )  
 }  
 **return this**.props.children  
 }  
}

Child.js  
  
**export default class** Child **extends** React.Component {  
 constructor(props) {  
 **super**(props);  
 **this**.state = { counter: 0 };  
 **this**.handleClick = **this**.handleClick.bind(**this**);  
 }  
  
 handleClick() {  
 **this**.setState(({counter}) => ({counter: counter + 1}));  
 }  
  
 render() {  
 **if** (**this**.state.counter === 3) {  
 **return**(  
 /\* Where the error has occurred, It will redirected to the error Boundary component \*/  
 <ErrorBoundary />  
 )  
 }  
 **return** <h1 onClick={**this**.handleClick}>{**this**.state.counter}</h1>;  
 }  
}

/\*This is the sample property file which has a list of error messages that needs to be accessed by all components\*/

PropertyFile.js /\*TODO…. Consider XML for ReactJS\*/

**const** PropertiesFile = {  
 msg: [  
 {errorID: 1, errorMessage: "Empty fields", description: "Please enter your first name.", interestedParty: "End User" },  
 {errorID: 2, errorMessage: "Miss match", description: "Email does not match", interestedParty: "End User"},  
 {errorID: 3, errorMessage: "Miss the requirement", description: "The 5-digit ZIP code you enter should include only numbers.", interestedParty: "Developer"},  
 {errorID: 4, errorMessage: "re-register with the same email ID\n", description: "The email address you entered is already registered", interestedParty: "End User/Developer"},  
 {errorID: 5, errorMessage: "Credential error", description: "Your email and/or password are incorrect. Please try again.", interestedParty: "End User"}  
 ],  
 all: **function**() { **return this**.msg},  
 getOne: **function**(errorId) {  
 **const** isError= p => p.number === errorId  
 **return this**.msg.find(isError)  
 }  
}  
**export default** PropertiesFile

Error Detected/Alien exception

Notify the interestedUser

(Message or log file)

Propagate the Exception

Wrap and Throw Exception

Gathering Error Information

Local Exception Handling