

# ErrorBoundary

효과적인 에러 관리

# 클래스형 컴포넌트

react v16.8 이전에 작성된 코드라면 대부분 클래스형 컴포넌트

```
import React, { Component } from "react";

class App extends Component {
  render() {
    return (
      <div>
        <h1>Hello, React!</h1>
      </div>
    );
  }
}

export default App;
```

# 클래스형 컴포넌트 vs 함수형 컴포넌트

함수형 컴포넌트는 생명주기 메서드가 없다

: useEffect를 사용해 비슷하게 구현할 수 있지만, 목적의 본질은 다르다

```
class MyComponent extends React.Component {
  componentDidMount() {
    console.log("컴포넌트가 마운트되었습니다.");
  }

  componentDidUpdate(prevProps, prevState) {
    console.log("컴포넌트가 업데이트되었습니다.");
  }

  componentWillUnmount() {
    console.log("컴포넌트가 언마운트됩니다.");
  }

  render() {
    return <div>Hello</div>;
  }
}
```

```
import React, { useEffect } from "react";

function MyComponent() {
  useEffect(() => {
    console.log("컴포넌트가 마운트되었습니다."); // componentDidMount

    return () => {
      console.log("컴포넌트가 언마운트됩니다."); // componentWillUnmount
    };
  }, []);

  return <div>Hello</div>;
}
```

# ErrorBoundary

try-catch 방식이 아니라  
선언적으로 에러를 처리할 수 있는 컴포넌트

# ErrorBoundary

에러가 발생하면 catch하고, 원하는 컴포넌트를 보여준다

```
useEffect(() => {
  dispatch(fetchComments);
}, []);

if (error) {
  // 실제로는 더 많은 정보(페이지 웹에서 사용하는 에러 코드)를 담고 있는 에러 객체입니다.
  throw error;
}

if (isLoading) {
  return <Loading />;
}

return children;
}
```

# ErrorBoundary

불편한점)

1. 공식문서에서 클래스형 컴포넌트로 작성되어 있음
2. 함수형으로 리액트 코드를 작성하는 우리에게는 익숙하지 않고, 일관성도 떨어진다.



그러면 못쓰나요?



# react-error-boundary

함수형으로도 에러바운더리를 구현할 수 있는 라이브러리

```
const Example = () => {  
  return (  
    <ErrorBoundary fallback={ErrorFallback}>  
      <ErrorableComponent />  
    </ErrorBoundary>  
  );  
};  
  
export default Example;
```



# props

## 1. fallback: 에러 발생시 보여줄 컴포넌트

```
import React from "react";
import { ErrorBoundary } from "react-error-boundary";

function BuggyComponent() {
  throw new Error("This is a test error!");
}

function App() {
  return (
    <ErrorBoundary fallback={<div>An error occurred!</div>}>
      <BuggyComponent />
    </ErrorBoundary>
  );
}

export default App;
```

# props

## 2. fallbackRender: 에러 발생시 보여줄 함수

```
<ErrorBoundary
  fallbackRender={({ error, resetErrorBoundary }) => (
    <div>
      <p>Error: {error.message}</p>
      <button onClick={resetErrorBoundary}>Try again</button>
    </div>
  )}
>
  <MyComponent />
</ErrorBoundary>
```

# props

## 3. onError: 에러 발생시 호출되는 콜백함수

```
<ErrorBoundary
  onError={(error, info) => {
    console.error("Error caught:", error, info);
  }}
  fallback={<div>Oops! An error occurred.</div>}
>
  <MyComponent />
</ErrorBoundary>
```