componentDidCatch로 에러 잡아내기

App.js 에서

function App(){

Return(

<ErrorBoundary>

<User/>

</ErrorBoundary>

);

} 로 감싸주고

errorBoundary.js

Import React, { component } from ‘react’;

Class ErrorBoundary extends Component {

State = {

Error : false

};

componentDidCatch(error, info) {

Console.log(‘에러가 발생했습니다.’);

Console.log({

Error,

Info

});

This.setState({

Error : true

});

}

Render(){

If(this.state.error){

Return <h1>에러 발생!</h1>;

}

Return this.props.children;

}

}

Export default ErrorBoundary;

//componentDidCatch 메서드에는 두개의 파라미터를 사용하게 되는데, 첫번째 파라미터는 에러의 내용, 두번째 파라미터에서는 에러가 발생한 위치를 알려줍니다.

2장 리액트 컴포넌트 스타일링하기

1. Sass

* 복잡한 작업을 쉽게 할 수 있게 해주고, 코드의 재활용성을 높여줄 뿐만 아니라, 코드의 가독성을 높여주어 유지보수를 쉽게 해줍니다.
* 스타일 파일에서 사용 할 수 있는 변수를 선언 할 수도 있고 함수를 사용 할 수도 있습니다.
* <button className={[‘Button’,size].join(‘’)}>{children}</button>
* 아니면 className={`Button ${size}`}
* classnames라는 라이브러리를 사용하는 것이 훨씬 편합니다. ( 조건부 스타일링을 할 떄 함수의 인자에 문자열, 배열, 객체 등을 전달하여 손쉽게 문자열을 조합할 수있습니다. )
* .button{ &:hover{ background} } 그리고 .button{ &.large } 는 .button.large{} 라는 뜻
* ...rest props 전달하기
* 필요한 이벤트가 있을 때마다 props로 넘겨주는 절차가 귀찮은데, 이러한 문제를 해결 해 줄수 있는 문법은 spread 와 rest입니다. 이 문법은 주로 배열과 객체, 함수의 파라미터, 인자를 다룰 때 사용하는데, 컴포넌트에서도 사용할 수있습니다.
* Ex) <button className={classNames(‘Button’,size, color, {outline, fullWidth})}
  + {...rest}
  + > {children}
  + </button>
  + -> 이렇게 ...rest를 사용해서 우리가 지정한 props를 제외한 값들을 rest라는 객체에 모아주고, <button> 태그에 {...rest}를 해주면, rest 안에 있는 객체 안에 있는 값들을 모두 <button>태그에 설정을 해준다.
  + Spread와 rest
    - Spread : 기존 배열은 건드리지 않으면서, 새로운 배열에 항목을 추가적으로 넣을수도있다.
    - Ex) const animals = [‘개’,’고양이’,’참새’];
    - Const anotherAnimals = [...animals, ‘비둘기’];

Console.log(animals);

Console.log(anotherAninals);

-> 새로운 anotherAnimals 배열에 animals가 가지고 있는 내용을 모두 집어넣고, ‘비둘기’라는 항목을 추가적으로 넣었습니다.

* Rest
  + Rest생김새는 spread랑 비슷한데, 역할이 매우 다릅니다.
  + Rest는 객체, 배열 그리고 함수의 파라미터에서 사용가능
  + Ex) const purpleCuteSlime = {
    - Name : ‘슬라임’,
    - Attribute : ‘cute’,
    - Color : ‘purple’
  + };
  + Const { color, ...rest } = purpleCuteSlime;
  + Console.log(color);
  + Console.log(rest);
  + -> 이렇게 하면 rest 안에 color 값을 제외한 값이 들어있습니다.
  + Rest는 객체와 배열에서 사용할 때는 이렇게 비구조화 할당 문법과 함께 사용됩니다. 주로 사용할 때는 위와 같이 rest라는 키워드를 사용하게 되는데, 추출한 값의 이름이 꼭 rest일 필요는 없습니다.
  + Ex) const { color, ...cuteSlime } = purpleCuteSlime;

1. CSS module

* 리액트 프로젝트에서 컴포넌트를 스타일링 할 때 CSS module 이라는 기술을 사용하면, css 클래스가 중첩되는 것을 완벽히 방지 할 수있습니다.
* Css 파일의 확장자를 .module.css로 하면된다.
* Box.module.css
* Ex) .Box {
  + Background : black;
  + Color: white;
  + Padding : 2rem;
* } 이런식으로 지정하면 된다.
* Import styles from “./Box.module.css”;
* 사용은 return <div className={styles.Box}>{styles.Box}</div>;

1. 멋진 투두리스트 만들기
2. ContextAPI를 활용한 상태 관리

* TodoContext.js 에서
* State와 dispatch를 context 통하여 다른 컴포넌트에서 바로 사용할 수있게 해줄건데, 두개의 context를 만들어서 따로따로 넣어줄 것입니다.
* 이렇게 하면 dispatch만 필요한 컴포넌트에서 불필요한 렌더링을 방지 할 수있습니다.
* TodoStateContext = createContext();
* TodoDispatchContext = createContext();
* Export function useTodoState(){
  + Return useContext(TodoStateContext);
* }
* 이렇게 해주면 나중에 이렇게 사용할 수있음
* Const state = useTodoState();
* 실습폴더 참조