**<간단한 웹 게임 생성을 통해 배우는 React>**

**1. Why we use React?**

1. 사용자 경험의 질 상승
2. 재사용을 위한 코드 컴포넌트화
3. 데이터-화면 일치

**2. State**

What is the state?

- value 등이 바뀔 여지가 있는 부분 ex) button 태그 내 text 등

**# index.html**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**# Web Page**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

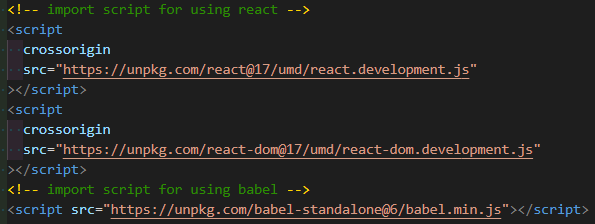
- button 클릭 시 state { liked: false -> true } 로 변경되고 따라서 button의 text인 unliked -> liked로 역시 변경.

**3. Babel**

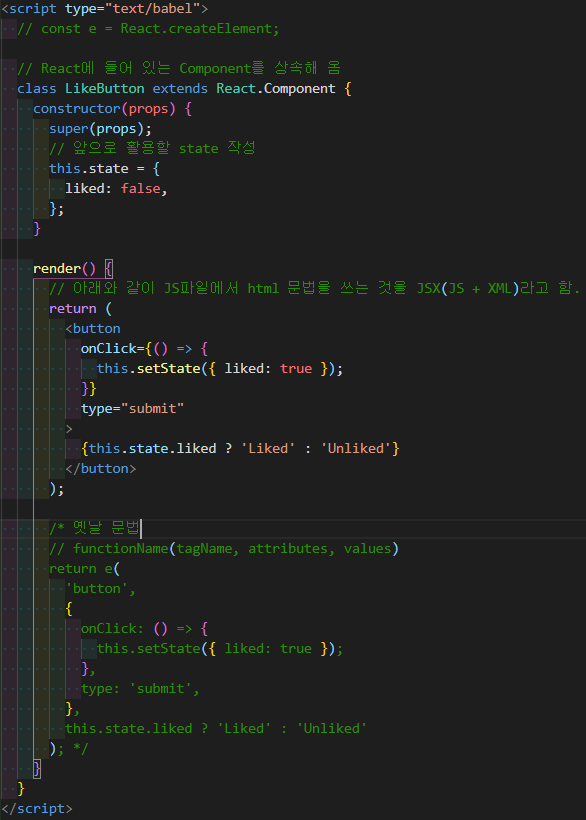
- JS파일에서 html 형식의 코드를 React.createElement로 변환하여 읽을 수 있게 해주는 기능.

**# index.html**

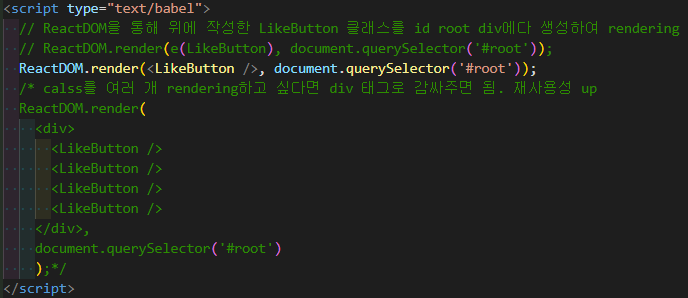
**- head**

****

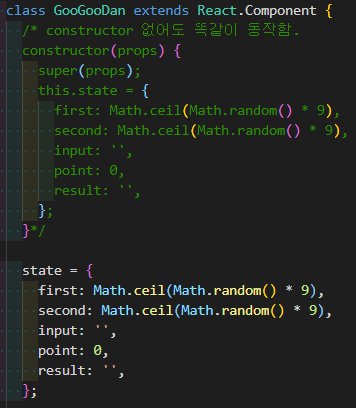
**- first script tag**

****

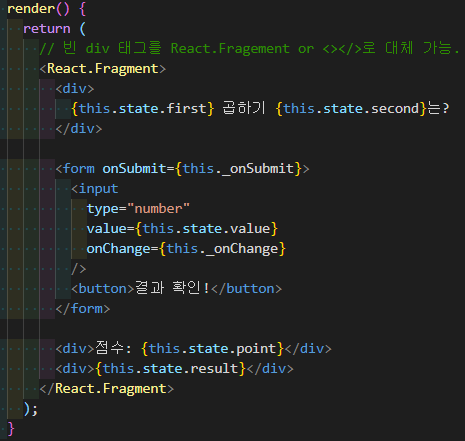
**- second script tag**

****

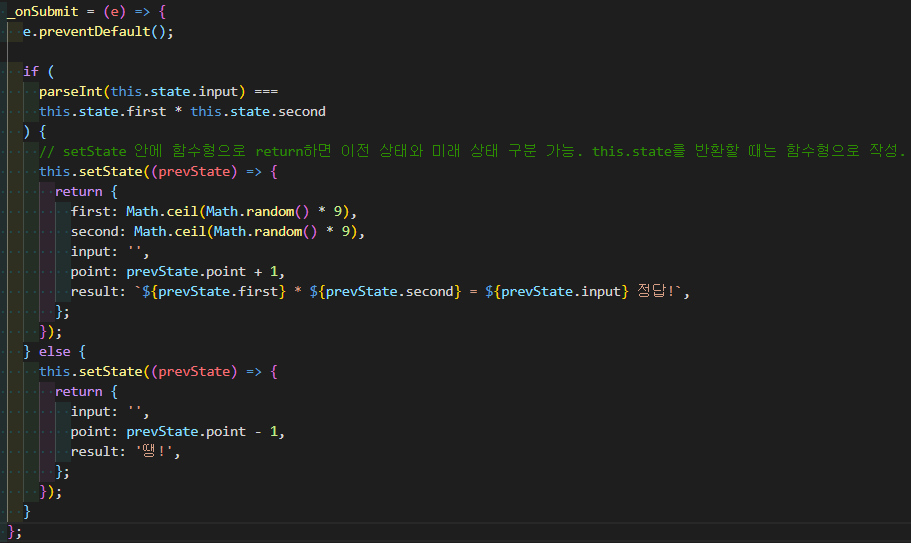
**4. 구구단 by React**

****

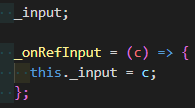
****

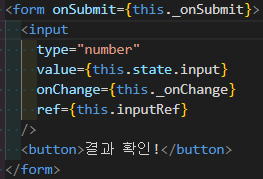
****

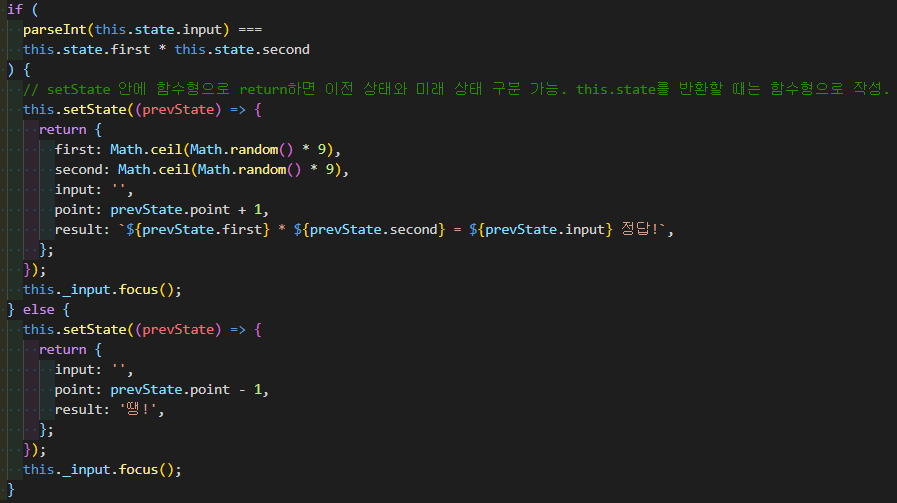
**5. \_onSubmit method 내 setState 함수형으로 변경하기**

****

**# input autofocus 구현**

****

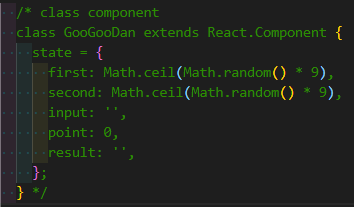
****

****

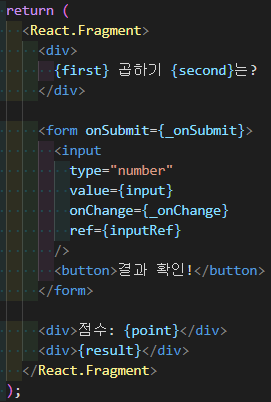
- react에서 setState에 변경이 일어날 때마다 render()가 새로 실행되므로 input 태그에 함수를 하드 코딩할 경우 작업 속도가 느려짐. 따라서 위와 같이 class 내 메서드로 따로 지정하여 활용.

**6. React Hooks**

- functional component에 useState, useEffect, useRef 기능 등을 추가한 것.







**7. WebPack**

- 여러 개의 components JS파일을 하나(app.js)로 합쳐서 유지 보수 용이하게 해주는 기능.

- webpack을 활용하려면 node.js를 먼저 알아야 한다. 참고로 node.js는 server 구축 도구가 아니라 단순한 JS 실행기임.

**# example/**

**installation**

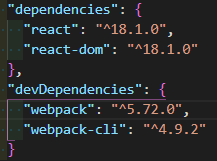
- create package.json: npm init

- install react, react-dom: npm i react react-dom

- install webpack: npm i -D webpack webpack-cli (\*-D는 개발용으로만 쓰겠다는 의미)

**Basic Settings**

**# package.json**

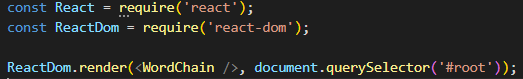
****

- dependencies는 서비스에 쓰이는 libraries, devDependencies는 개발에만 쓰이는 libraries.

**# example/webpack.config.js**



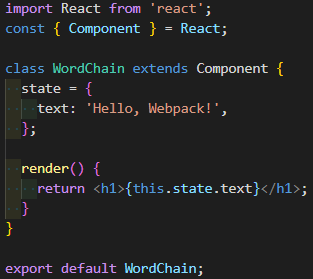
**# example/client.jsx**

****

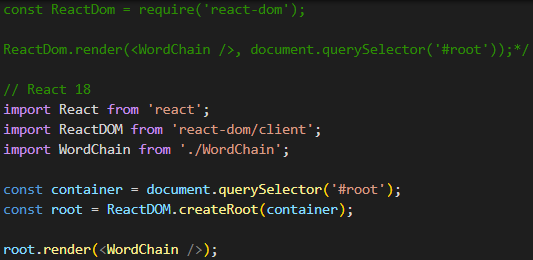
**8. Word Chain Game**

**webpack settings**

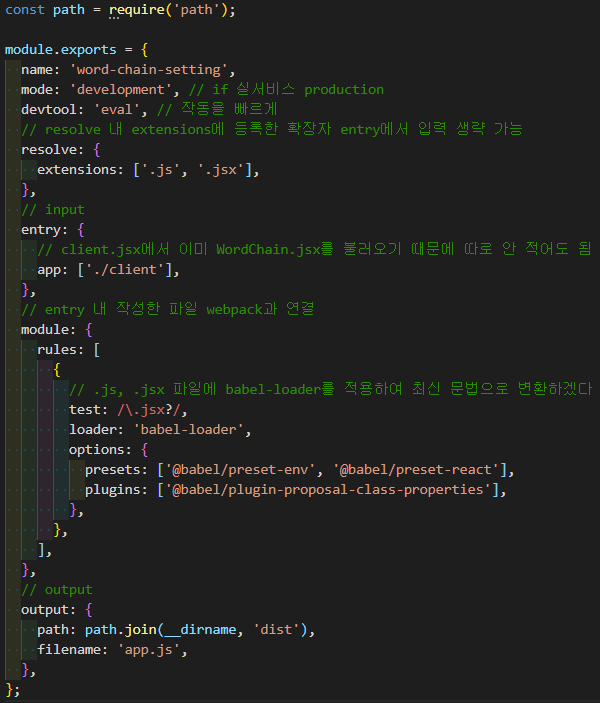
**# example/WordChain.jsx**

****

**# example/client.jsx**

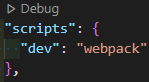
****

**# example/webpack.config.js**

****

**webpack 생성 2가지 방법**

**1. # example/package.json**

****

- npm run dev 입력 on terminal

**2. npx webpack 입력 on terminal**

**Babel**

**installation**

- npm i -D @babel/core @babel/preset-env @babel/preset-react babel-loader @babel/plugin-proposal-class-properties

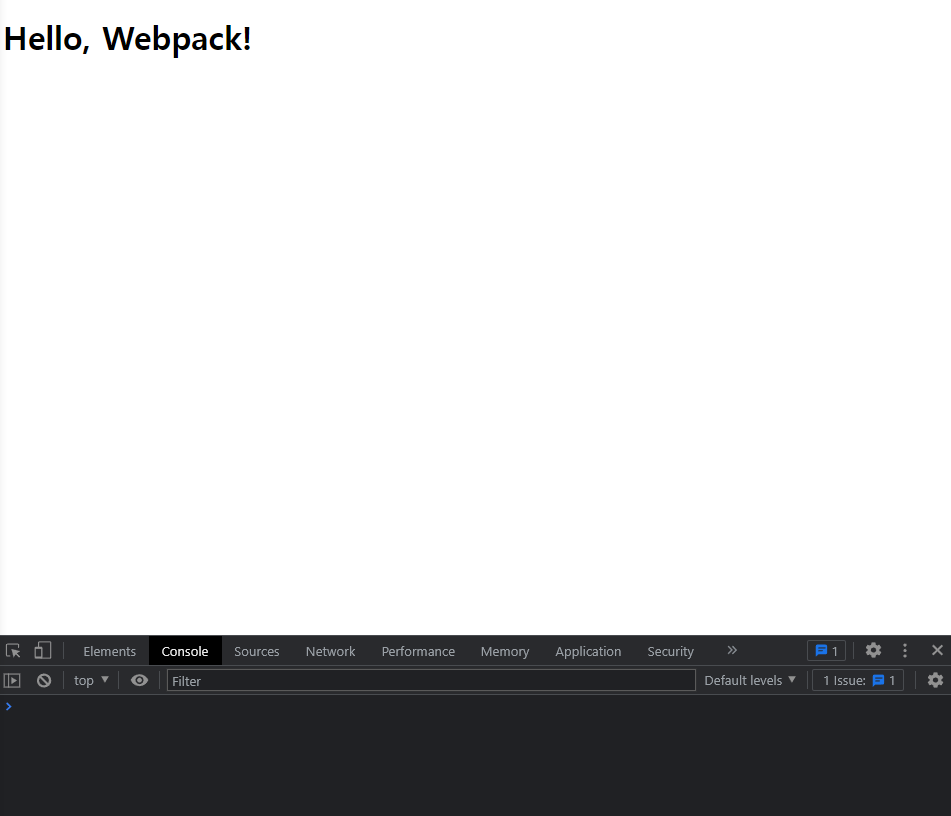
- @babel/core: 기본적인 babel. 작성한 코드를 최신 문법으로 변환.

- @babel/preset-env: 사용자 브라우저에 맞게 작성한 코드를 최신 문법으로 변환.

- @babel/preset-react: jsx파일 지원.

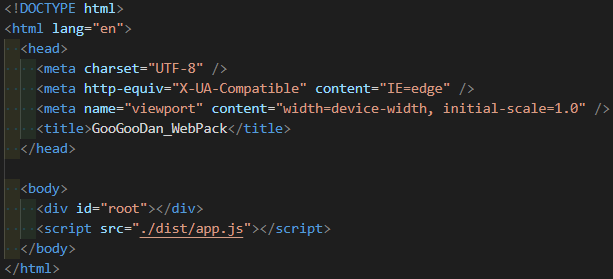
- babel-loader: babel과 webpack 연결.

**# WebPage**

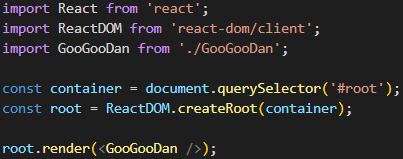
****

**9. 구구단 WebPack ver.**

**# googoodan\_webpack.html**

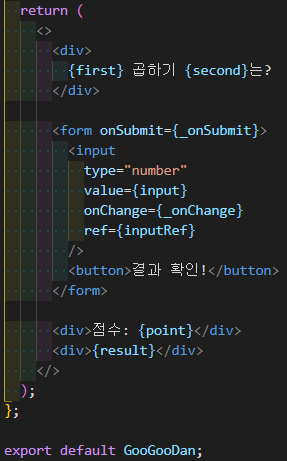
****

**# client.jsx**

****

**# GooGooDan.jsx**

****

****

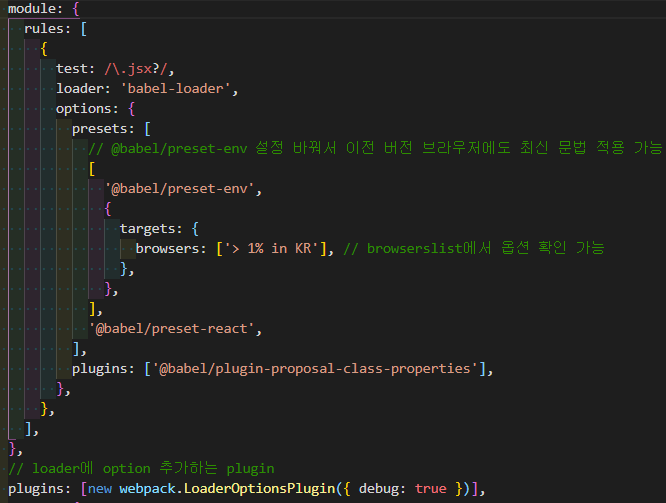
**# webpack.config.js**

****

**10. @babel/preset-env와 Plugins**

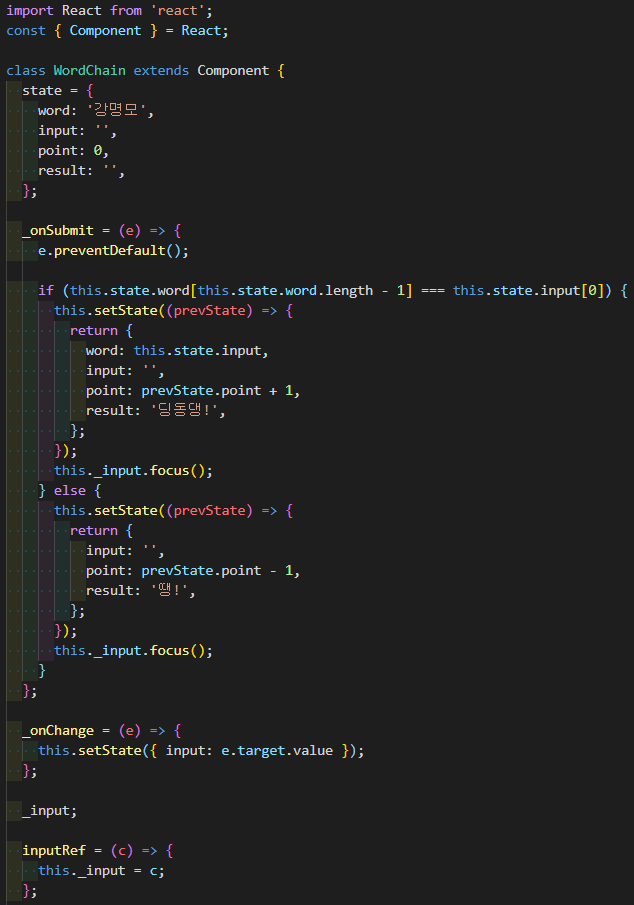
**# webpack.config.js**

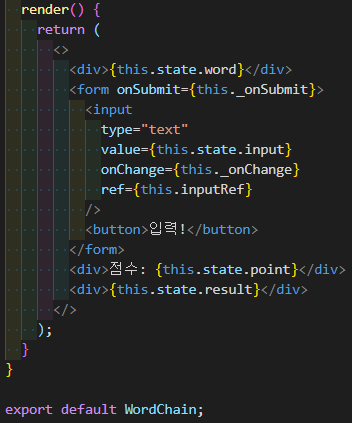
****

****

**11. Word Chain Game by Class**

**# word\_chain/WordChain.jsx**

****

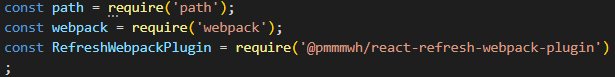
****

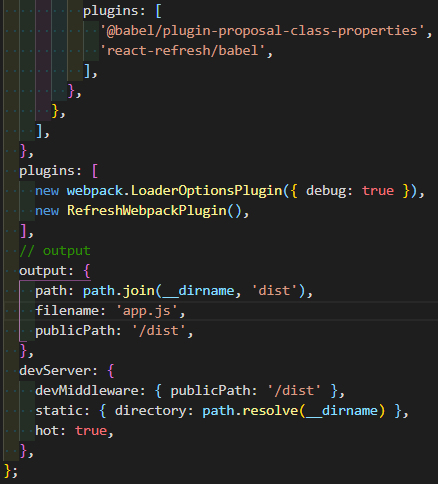
**webpack dev server와 hot reloading**

**installation**

- npm i react-refresh @pmmmwh/react-refresh-webpack-plugin webpack-dev-server -D

**# word\_chain/webpack.config.js**

****

****

**12. React Loops**

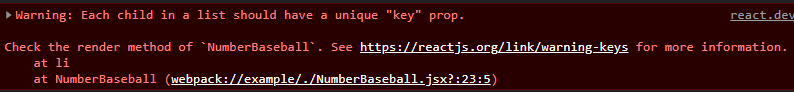
**map**

- react에서 반복문을 쓸 때는 map()을 활용한다.

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명



- 배열과 map 함수만 입력하면 위와 같이 'key'가 필요하다는 err가 뜬다. 따라서 배열 내 요소마다 고유한 key값을 설정해줘야 한다.(실무에서 가장 골치 아픈 부분) 이때 key에다가 map 배열 함수가 제공하는 idx값을 넣으면 성능 최적화 부분에서 문제가 생긴다.

- 가독성을 위해 1차원 배열을 객체로 고쳐 필요한 값들을 프로퍼티에 입력하여 관리하면 보다 가독성이 좋아진다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

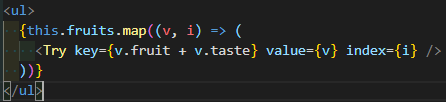
자동 생성된 설명

**Component화**

- 위에서 만든 반복문 요소를 component화하여 코드 관리를 효율적으로 할 수 있다.

- 이때 map 함수에서 탐색하는 배열을 component화한 파일에서 받아오지 못하는 문제가 발생하는데 이를 'props'를 통해 해결한다. props로 부모 component의 요소를 자식 component에 상속하여 아래와 같이 활용 가능 한 것이다.

**# NumberBaseball.jsx(부모 component)**

****

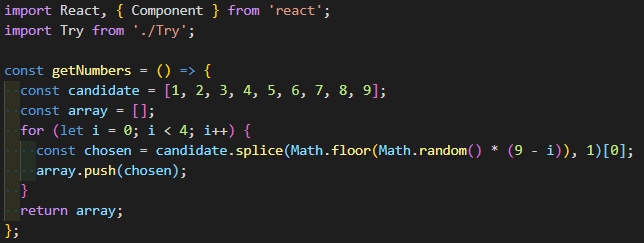
**# Try.jsx(자식 component)**

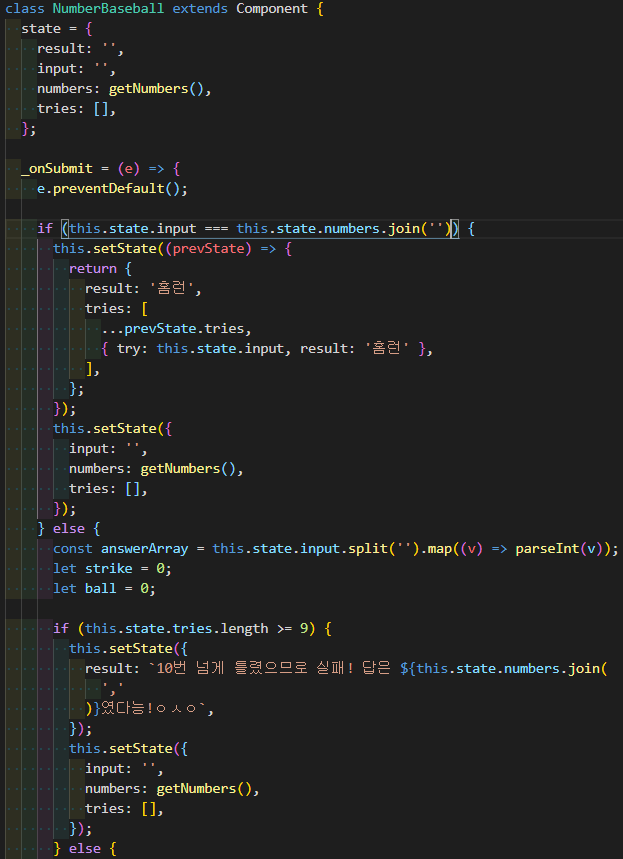
****

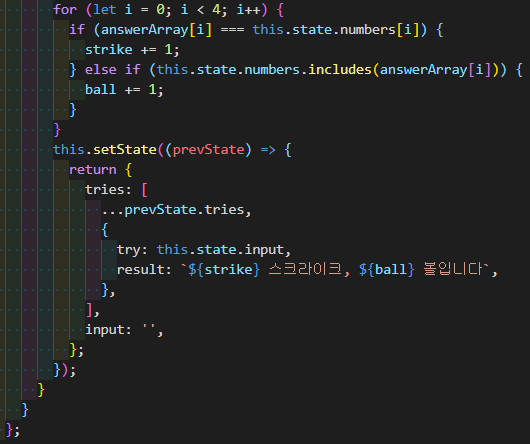
- 부모 component에서 <Try /> 부분에 value와 index로 물려준 값들을 자식 component에서 props로 받아와 활용.

**13-1. Number Baseball Game Class ver.**

**# NumberBaseball.jsx**

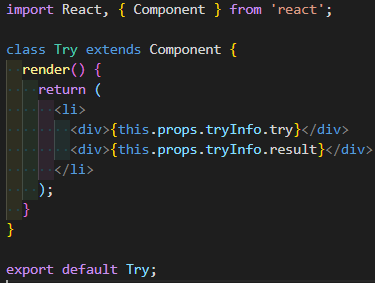
****

****

****

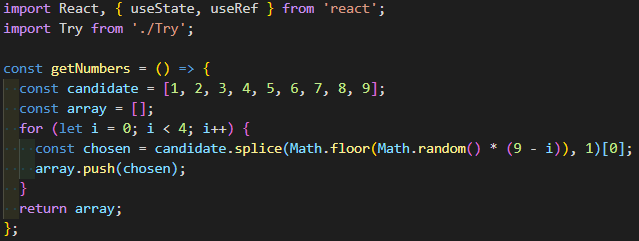
****

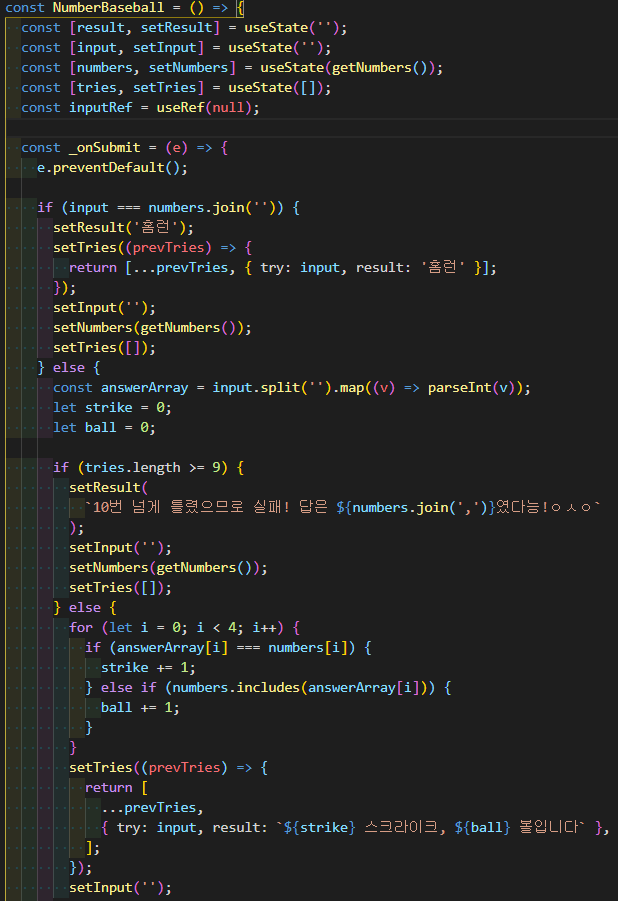
**# Try.jsx**

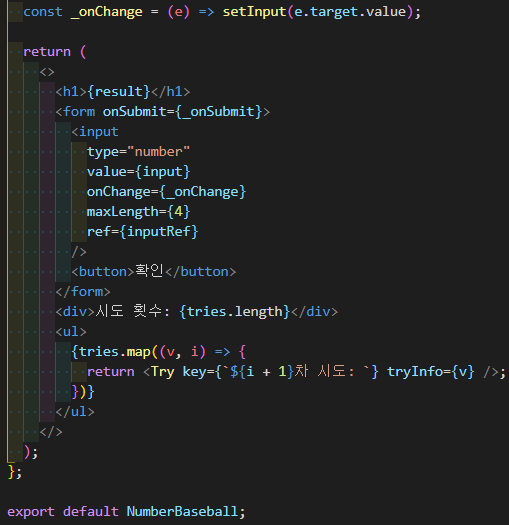
****

**13-2. Number Baseball Game Hooks ver.**

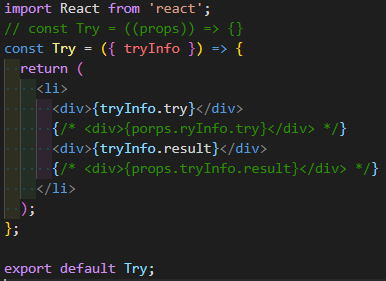
**# NumberBaseball.jsx**

****

****

****

**# Try.jsx**

****

**14. Solving Problems related to Rendering**

- props나 state가 변경될 때마다 새롭게 rendering됨.

- react dev tool을 활용하면 rendering될 때마다 색깔 표시. 빨간색에 가까워질수록 작업 과부하.



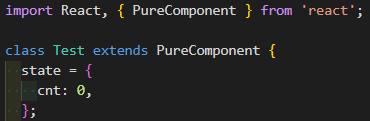
- input창에 값만 입력했는데 나머지 element들도 전부 rendering됨. 따라서 application 크기가 커질수록 문제 발생.

- 아래와 같은 방법들로 이 같은 문제를 해결할 수 있음.

**# shouldComponentUpdate() method**

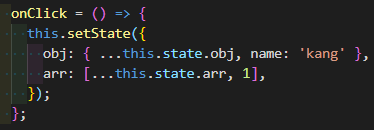
****

**# pureComponent**

****

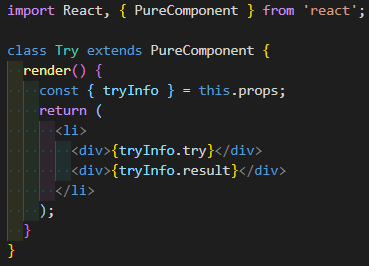
- PureComponent를 쓰면 state 내의 값과 변경된 값을 자동으로 비교해줘서 변경이 일어날 때만 rendering해줌.

- obj나 arr를 생성할 때는 아래와 같이 작성해야 PureComponent가 변경 여부를 인지 가능.

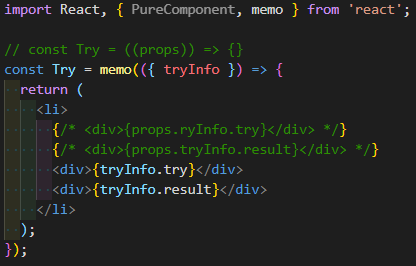


**# Change Number Baseball Game**

**# Try.jsx Class ver. with PureComponent**

****

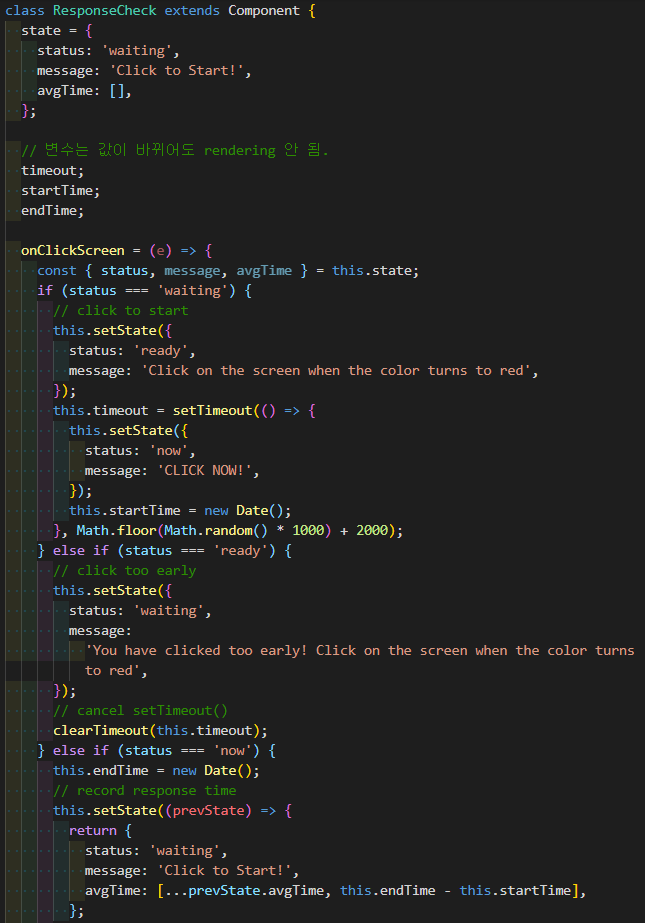
**# Try.jsx Hooks ver. with Memo**

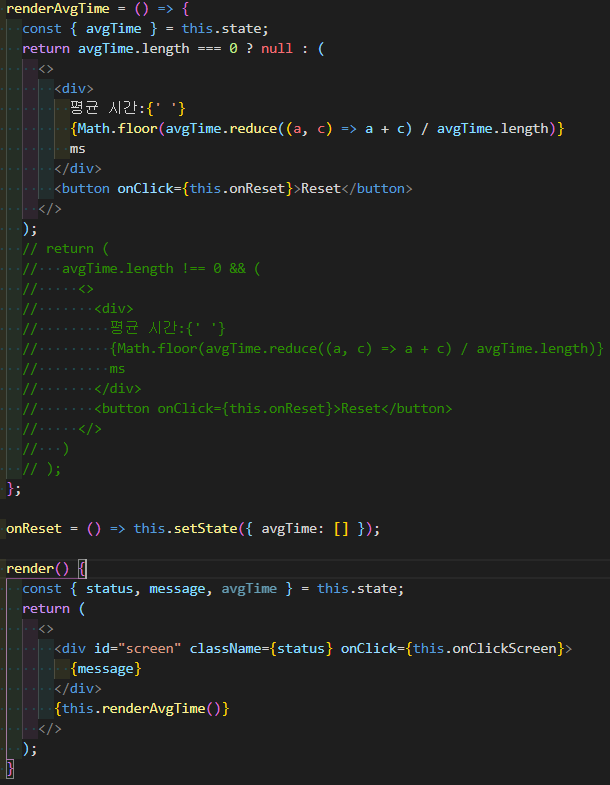
****

- child component에 PureComponent 혹은 memo를 적용하였다면 parent component에도 적용.

**15-1. ResponseCheck.jsx Class ver.**

**# ResponseCheck\_class.jsx**

****

****

- jsx 내에서 if문을 직접 입력 할 수 없기 때문에 따로 빼서 삼항 연산자 등으로 조건문을 작성한다.

**15-2. Response Check Game Hook ver.**

- 값이 바뀌어도 rendering 안 되게 하려면 useRef() 쓰면 됨.

- useRef() 적용한 변수는 current를 붙여서 활용.

**#** **ResponseCheck\_hooks.jsx**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**15-3. change renderAvgTime() function to components**

**# class**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**# hooks**

**텍스트이(가) 표시된 사진

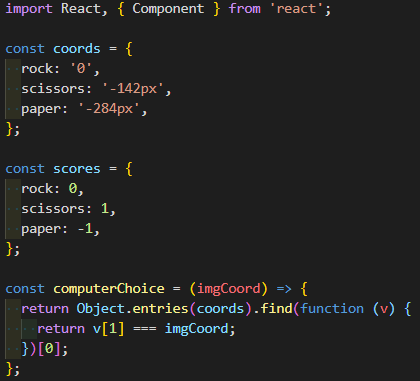
자동 생성된 설명**

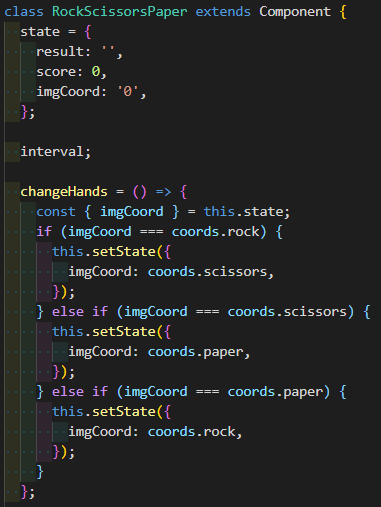
**16. Rock! Scissors! Paper!**

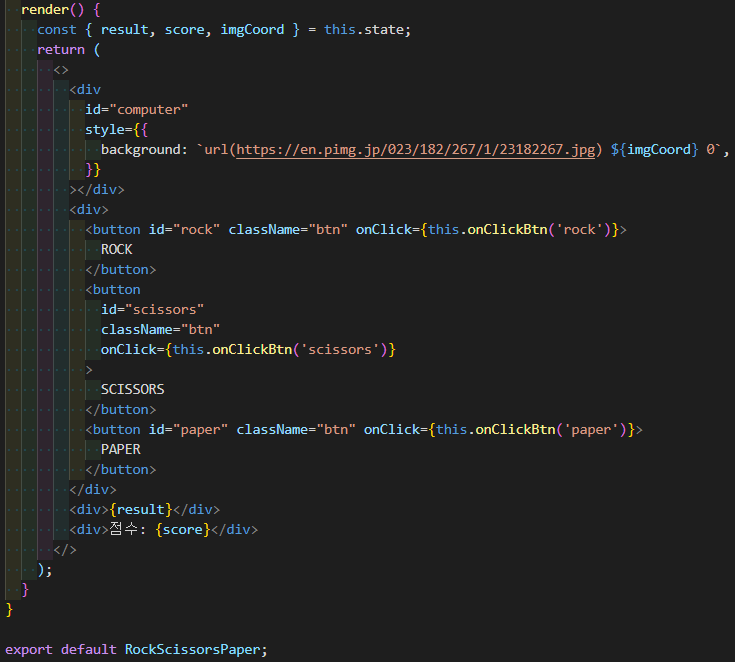
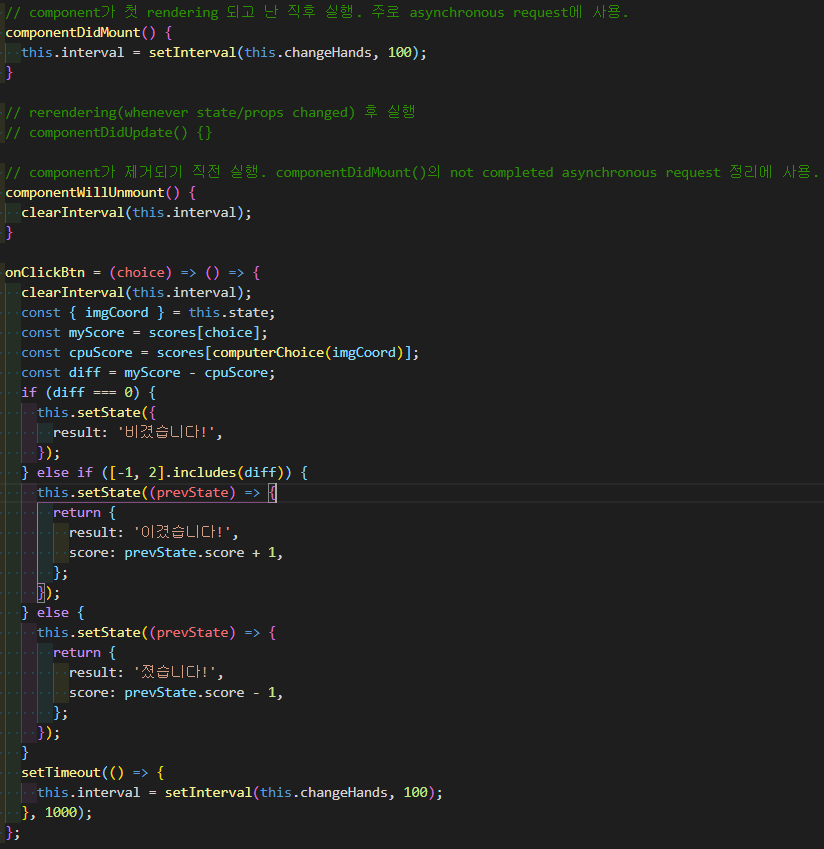
**Component's Life Cycle(if class)**

constructor → render → ref → componentDidMount → (whenever setState/props changed → shouldComponentUpdate → render → componentDidUpdate) → when parent component removes child component → componentWillUnmount → extinction

**# RockScissorsPapre\_class.jsx**

****

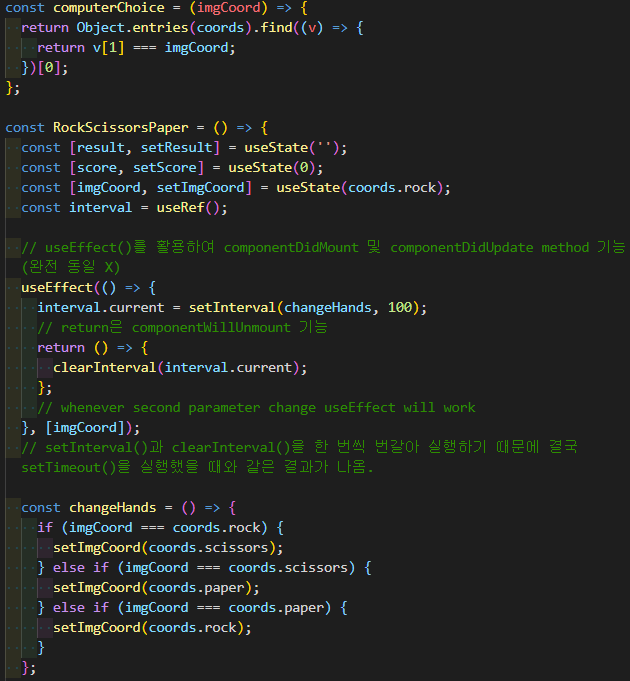
****

****

**# RockScissorsPaper\_hooks.jsx**

- Hooks에서는 component life cycle 관련 method를 제공하지 않기 때문에 useEffect()를 활용하여 해당 기능을 구현하는 것이 관건이다.

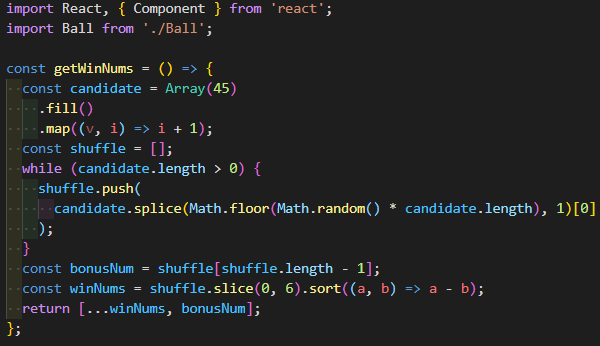


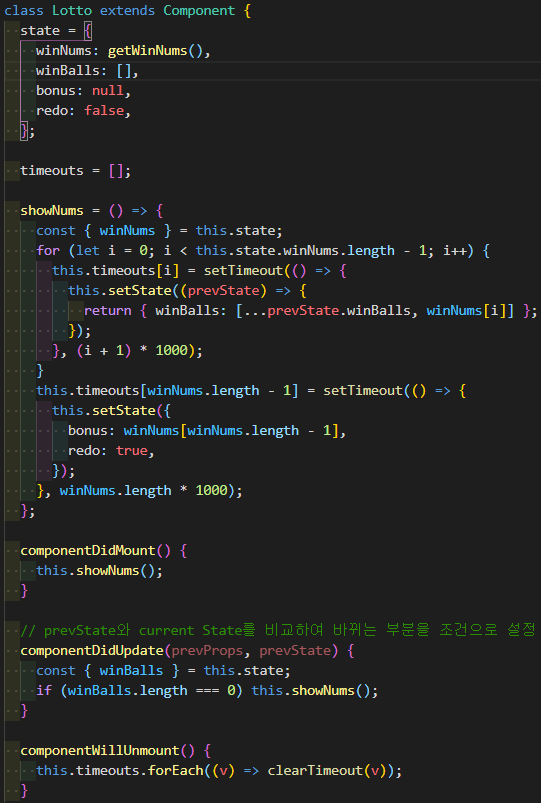


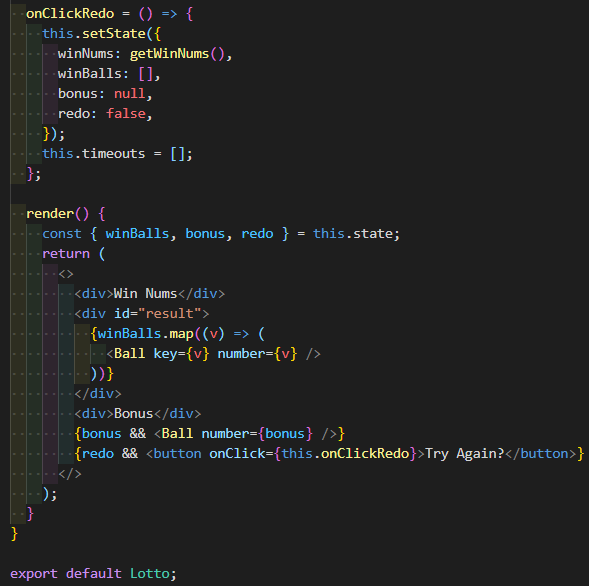


**17. Lotto**

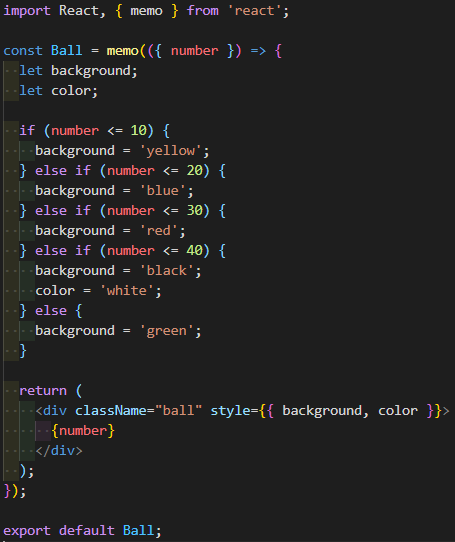
**# Lotto\_class.jsx**

****

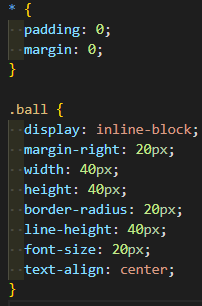
****

****

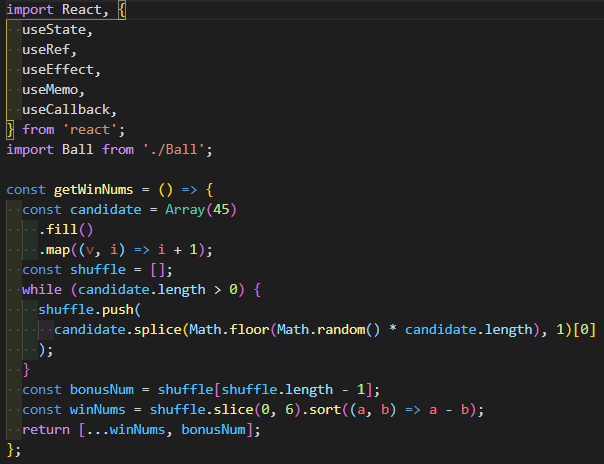
**# Ball.jsx**

****

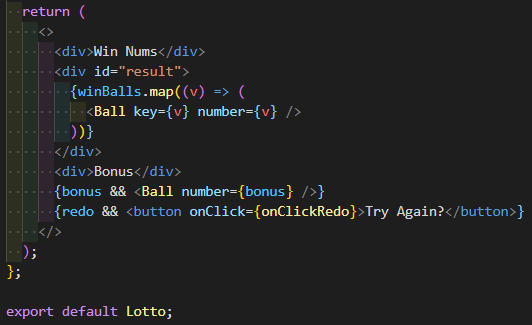
**# style.css**

****

**# Lotto\_hooks.jsx**

****

****

****

- useMemo: 함수 return값 기억

- useCallback: 함수 자체 기억. 전체 함수 재실행되어도 기억해둔 함수는 재실행 X

- 함수 실행 시간 길 때 해결 방법

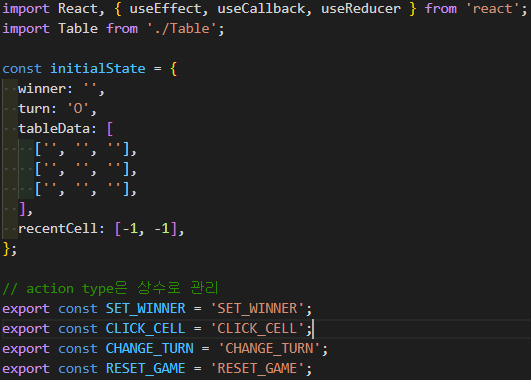
- 두 번째 parameter에 담은 값이 변경될 때마다 재실행. 아무 요소도 배열에 입력하지 않으면 최초 상태를 계속 기억하여 문제 발생

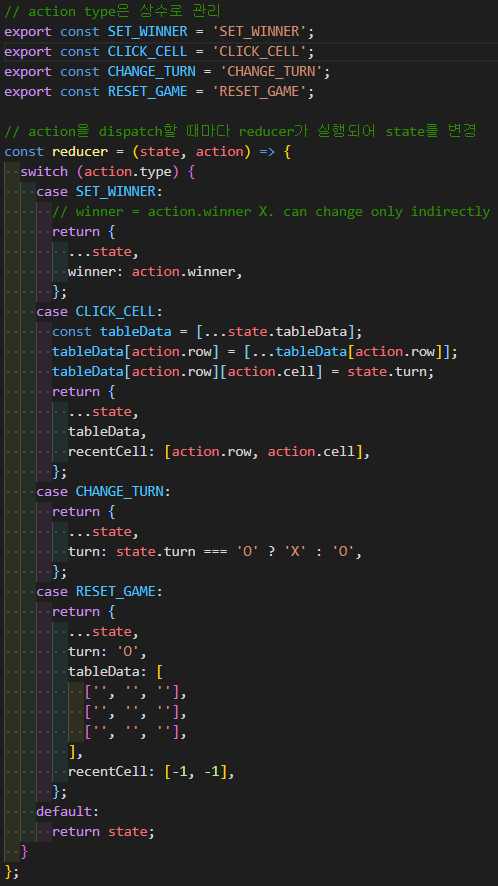
- child component의 props로 function을 넘길 때 useCallback 필수. 불필요한 rerendering 방지

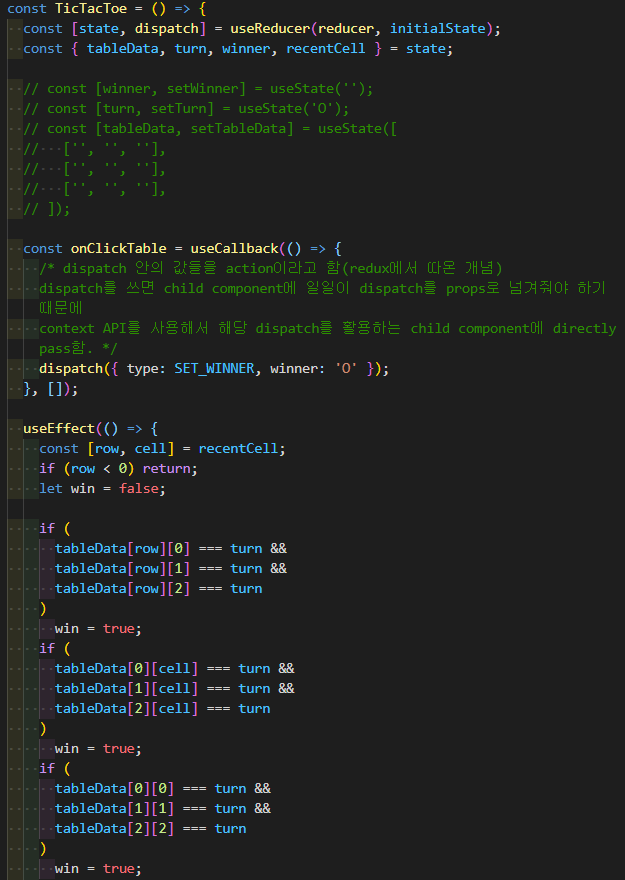
- useRef: 일반적인 값 기억

**17. Tic Tac Toe**

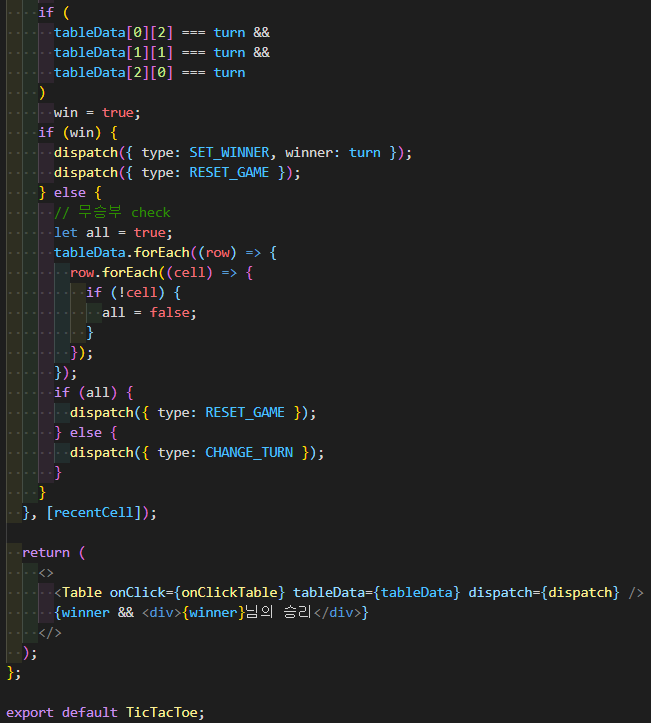
**# TicTacToe.jsx**



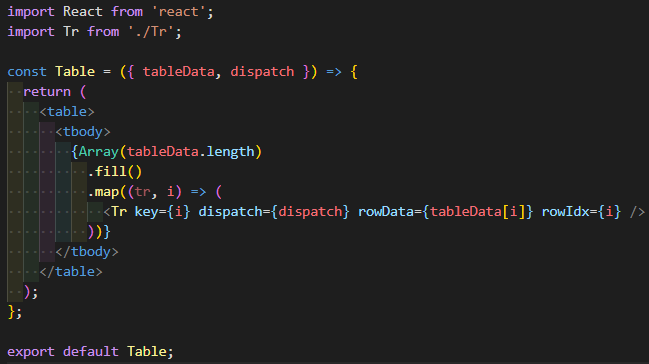




- useReducer(): redux에서 차용한 개념으로 여러 개의 state를 관리하기 편하게 하나로 묶고, state가 변경될 때마다 action을 dispatch해서 변경 사항 적용. 다만, redux는 state들이 동기적으로 변경되는 반면, useReducer()로는 비동기적으로 변경된다.



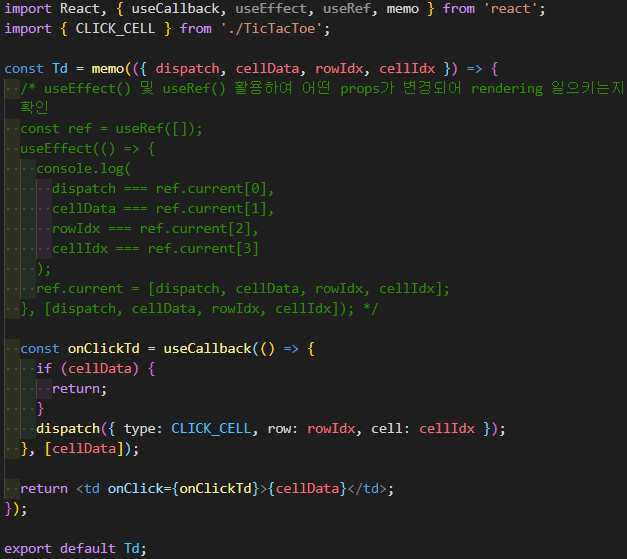
**# Table.jsx**

****

**# Tr.jsx**

****

**# Td.jsx**

****

**18. Minesweeper**

**# Minesweeper.jsx**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**# Setting.jsx**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**# Table.jsx**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**# Td.jsx**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**# Tr.jsx**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

- useContext()를 사용하면 rerendering issue가 발생하여 성능 저하가 초래되므로 memo, useMemo 등으로 성능 최적화를 잘 해줘야 함.