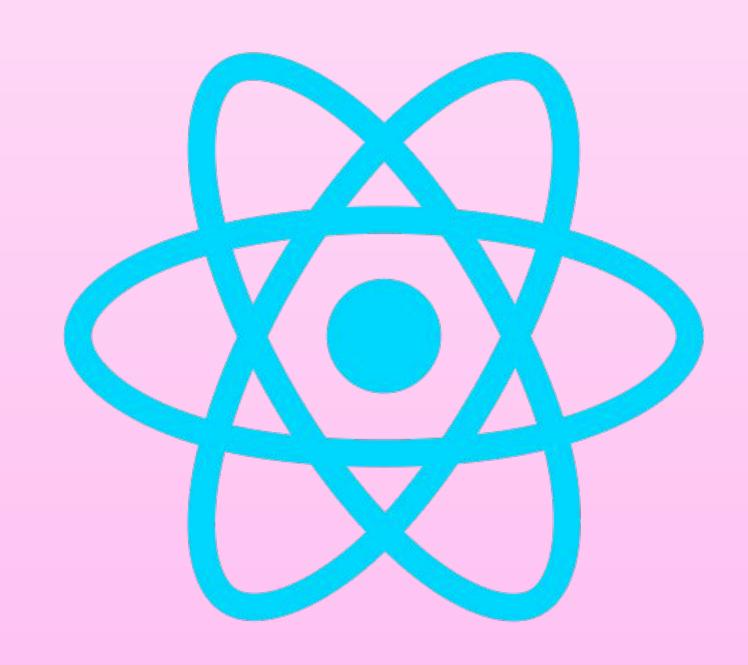
Immer 쉬운 불변성 유지

Immer

독일어로 'Always(항상)' 라는 뜻

immer가 나오게 된 배경



불변성을 지켜야하는 상태관리

불변성을 지키기 위한 방법

```
Table — type: 'table'
            - rowEdges: TableEdge[][] ---- TableEdge ---- x1, y1, x2, y2: number
---- style: StrokeStyle ---- stroke
                                                                            - strokeWidth
                                                                            strokeDasharray
            — colEdges: TableEdge[][] — (same as rowEdges)
6
            8
                                         - borders: CellBorders ---- top: TableEdge
9
                                                                  - bottom
10
                                                                  ·left
11
                                                                  - right
12
                                         — style: CellStyle —— isVisible
13
                                                               fill
14
                                                               textBox?
15
                                         — merge: CellMerge — isMerged
16
                                                             — rowSpan, colSpan
17
                                                              - start? { row, col }
18
19
                                        L split: CellSplit — isSplit
20
                                                             — splitType?
21
                                                             - splitLine?
22
                                                             └─ splitedCells[] ──── SplitedCell
23
24
                                                                                 - fill
25
26
                                                                               └─ textBox?
27
             - selectedCells: Set<string>
            - selectedRowEdges: Set<string>
28
            L selectedColEdges: Set<string>
29
30
```

불변성을 지키기 위한 방법

만약 병합된 셀의 정보를 변경한다면?

```
1 const updatedTable = {
    ...table,
     cells: table.cells.map((rowCells, rIdx) =>
       rowCells.map((cell, cIdx) => {
         if (rIdx === row && cIdx === col) {
           return {
             ...cell,
             merge: {
               ...cell.merge,
10
               start: { row: newRow, col: newCol },
11
             },
12
13
         return cell
14
      })
15
16
```

불변성을 지키기 위한 방법

기존 방법

```
1 const updatedTable = {
 2 ...table,
     cells: table.cells.map((rowCells, rIdx) =>
       rowCells.map((cell, cIdx) => {
         if (rIdx === row && cIdx === col) {
           return {
             ...cell,
             merge: {
               ...cell.merge,
               start: { row: newRow, col: newCol },
 10
 11
             },
 12
 13
         return cell
 14
 15
       })
 16
 17 }
```

immer 적용

```
1 import { produce } from 'immer'
2
3 const updatedTable = produce(table, (draft) => {
4   const cell = draft.cells[row][col]
5   if (cell.merge.isMerged) {
6    cell.merge.start = { row: newRow, col: newCol }
7   }
8 })
9
```

immer 사용방법 - produce(기본)

```
1 import { produce } from "immer";
2
3 const baseState = { count: 0 };
4
5 const nextState = produce(baseState, (draft) => {
6 draft.count += 1; // 직접 수정하는 것처럼 동작
7 });
8
9 console.log(baseState); // { count: 0 }
10 console.log(nextState); // { count: 1 }
```

immer 사용방법 - uselmmer(with react)

```
1 import React, { useCallback } from "react";
 2 import { useImmer } from "use-immer";
 4 const TodoList = () => {
     const [todos, setTodos] = useImmer([
  6
         id: "React",
         title: "Learn React",
  8
         done: true
 9
 10
       },
11
12
       id: "Immer",
13
         title: "Try Immer",
         done: false
14
15
     ]);
16
17
```

```
1 const handleToggle = useCallback((id) => {
     setTodos((draft) => {
    const todo = draft.find((todo) => todo.id === id)
      todo.done = !todo.done
 6 }, [])
 8 const handleAdd = useCallback(() => {
     setTodos((draft) => {
       draft.push({
         id: 'todo_' + Math.random(),
12
       title: 'A new todo',
13
         done: false,
      })
14
15
16 }, [])
17
```

immer - set, map

```
1 import { enableMapSet } from 'immer';
 2 import Router from './Router';
 4 // immer에서 Map과 Set을 사용하기 위해 활성화
 5 enableMapSet();
  6
 8 /**
 9 * 에셋 저작도구 최상단 컴포넌트
10 */
11 export default function App() {
       return <Router />;
12
13 }
 14
```

immer 장점

간편하게 불변성 유지

spread operator를 중첩적으로 사용하 지 않고도 간편하게 불변성 유지

자동 동결

한번 immer를 적용한 객체는 자동으로 내부 속성이 object.freeze() 적용되어 실수 방지

가독성

간결한 코드로 인해 가독성 향상

zustand middleware

```
1 import { create } from 'zustand'
 2 import { immer } from 'zustand/middleware/immer'
 4 type State = {
 5 count: number
 6 }
 8 type Actions = {
     increment: (qty: number) => void
     decrement: (qty: number) => void
11 }
12
13 export const useCountStore = create<State & Actions>()(
     immer((set) => ({
15
       count: 0,
 16
       increment: (qty: number) =>
         set((state) => {
17
18
           state.count += qty
19
         }),
       decrement: (qty: number) =>
20
21
         set((state) => {
           state.count -= qty
     }),
    })),
25 )
```