연습문제 09 풀이

> 이 예제는 Route를 처리할 필요가 없으며, Spinner와 Table 컴포넌트 및 useMountedRef의 코드는 수업시간에 사용된 것과 동일하지 별도로 제시하지 않겠습니다.

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <React.StrictMode>
        <App />
    </React.StrictMode>
);
```

App.js

드롭다운 선택 항목 제시하기

1) 14개의 빈 칸을 갖는 배열 만들기

`new Array(숫자)`는 주어진 칸 만큼의 빈 배열을 생성합니다. (Javascript 수업 참고바람)

```
new Array(2018 - 2005 + 1)
```

2) 빈 배열을 비구조 문법으로 복사하기

생성자를 통해 생성된 객체는 즉시 메서드를 사용하지 못합니다. (문법적 제약) 그렇기 때문에 아래의 방법은 동작하지 않습니다.

```
(new Array(2018 - 2005 + 1)).map( ... );
```

하지만 `new Array(숫자)` 방식으로 생성한 배열을 비구조 문법으로 복사하면 탐새 관련 메서드를 사용할 수 있습니다.

```
[...new Array(숫자)].map( ... )
```

3) 복사한 배열을 탐색하여 반복 수행하기

어차피 복사된 배열은 원소가 모두 undefined입니다. 그렇기 때문에 여기서는 0부터 시작되는 인덱스 값에 2005를 더한 값을 사용해서 2005부터 2018까지의 반복 처리를 구현해 내었습니다.

```
<select name='year' onChange={onSelectChange}>
    <option value="">-- 년도 선택 --</option>
    {[...new Array(2018 - 2005 + 1)].map((v, i) => {
        return (<option key={i} value={2005 + i}>{2005 + i}년도</option>)
    })}
</select>
```

전체 코드

```
import React from 'react';
import styled from 'styled-components';
import Spinner from './components/Spinner';
import Table from './components/Table';

// Axios 기능 제공 hook
import useAxios from 'axios-hooks';
// 페이지의 마운트 여부를 확인하기 위한 hook
import useMountedRef from './hooks/useMountedRef';

/** 드롭다운을 배치하기 위한 박스 */
const SelectContainer = styled.div`
     position: sticky;
     top: 0;
     background-color: #fff;
     border-top: 1px solid #eee;
     border-bottom: 1px solid #eee;
     padding: 10px 0;
     margin: 0;

     select {
         margin-right: 15px;
         font-size: 16px;
         padding: 5px 10px;
     }
 `;

// 접속할 백엔드의 URL
const URL = "http://localhost:3001/traffic_acc";

const App = () => {
    const [{ data, loading, error }, refetch] = useAxios(URL);
    const [year, setYear] = React.useState('');
    // 이 컴포넌트가 화면에 마운트 되었는지를 확인하기 위한 hook
    const mountedRef = useMountedRef();

    /** 드롭다운 선택 변경시 호출되는 이벤트 */
    const onSelectChange = React.useCallback(e => {
```

```
        e.preventDefault();

        // 드롭다운의 입력값 취득
        const current = e.target;
        const value = current[current.selectedIndex].value;
        setYear(value);
    }, []);

    /** state 상태값이 변경되었을 때 실행될 hook */
    React.useEffect(() => {
        // 컴포넌트가 화면에 마운트 된 이후에만 동작하도록 한다.
        if (mountedRef.current) {
            // 상태값 중에서 빈값이 아닌 항목들을 옮겨담는다.
            const params = {};

            if (year) {
                params.year = parseInt(year);
            }

            // Ajax 재요청
            refetch({
                params: params
            });
        }
        // hook함수 안에서 다른 상태값을 사용할 경우 해당 상태값을 모니터링 해야 한다.
    }, [mountedRef, refetch, year]);

    /** 에러가 발생했다면 에러 메시지를 표시한다. */
    if (error) {
        console.error(error);

        // 컴포넌트 자체가 함수이고, 함수가 실행도중 리턴을 하므로
        // 이 내용을 화면에 표시하고 컴포넌트의 실행은 중단된다.
        return (
            <div>
                <h1>Oops~!!! {error.code} Error.</h1>
                <hr />
                <p>{error.message}</p>
            </div>
        )
    }

    /** 메인 화면 구성 */
    return (
        <div>
            <h1>Exam 10</h1>

            {/* 로딩바 */}
            <Spinner visible={loading} />

            {/* 검색 조건 드롭다운 박스 */}
            <SelectContainer>
                <select name='year' onChange={onSelectChange}>
                    <option value="">-- 년도 선택 --</option>
```

```
                    {[...new Array(2018 - 2005 + 1)].map((v, i) => {
                        return (<option key={i} value={2005 + i}>{2005 +
i}년도</option>)
                    })}
                </select>
            </SelectContainer>

            {data && (
                <Table>
                    <thead>
                        <tr>
                            <th>번호</th>
                            <th>년도</th>
                            <th>월</th>
                            <th>교통사고 건수</th>
                            <th>사망자 수</th>
                            <th>부상자 수</th>
                        </tr>
                    </thead>
                    <tbody>
                        {data.map(({ id, year, month, accident, death,
injury }, i) => {
                            return (
                                <tr key={i}>
                                    <td>{id}</td>
                                    <td>{year}년</td>
                                    <td>{month}월</td>
                                    <td>{accident.toLocaleString()}건</td>
                                    <td>{death.toLocaleString()}명</td>
                                    <td>{injury.toLocaleString()}명</td>
                                </tr>
                            );
                        })}
                    </tbody>
                    <tfoot>
                        <tr>
                            <th colSpan="3">합계</th>
                            <th>{data.map((v, i) => v.accident).reduce((p,
c) => p + c, 0).toLocaleString()}건</th>
                            <th>{data.map((v, i) => v.death).reduce((p, c)
=> p + c, 0).toLocaleString()}명</th>
                            <th>{data.map((v, i) => v.injury).reduce((p,
c) => p + c, 0).toLocaleString()}명</th>
                        </tr>
                    </tfoot>
                </Table>
            )}
        </div>
    );
};

export default React.memo(App);
```