13-movie-rank

#01. 프로젝트 생성 및 초기화

1) 프로젝트 생성

> 프로젝트 이름은 영어 소문자만 사용 가능함.

```
yarn create react-app 11-redux
```

2) 프로젝트 생성 후 기초작업

프로젝트 초기화

대상	작업 내용
src폴더	App.css, App.test.js, index.css, logo.svg, setupTests.js, reportWebVitals.js 삭제
App.js	App.css와 logo.svg에 대한 참조(import) 구문 제거
index.js	index.css와 reportWebVitals.js에 대한 참조(import) 구문 제거 맨 마지막 행에 있는 `reportWebVitals()` 부분 삭제

3) 추가 패키지 설치

패키지 이름	설명
react-router-dom	SPA앱을 만들 때 사용. URL에 따라 실행할 Javascript를 분기한다.
react-helmet-async	`<head>`태그 내의 SEO관련 태그를 설정한다.
sass	sass와 scss 컴파일 기능 제공
styled-components	styled component 지원
styled-components-breakpoints	styled component에서 media query를 쉽게 사용할 수 있게 한다.
dayjs	날짜 처리 기능 제공 (리액트와 직접적인 연관은 없음)
axios	Ajax 요청 라이브러리
react-loader-spinner	로딩바(spinner) 컴포넌트
axios-hooks	Axios 라이브러리 사용을 간소화 시켜 주는 hook
react-redux	리액트에서 redux를 사용할 수 있도록 해주는 컨테이너. redux에 의존한다.
@reduxjs/toolkit	리액트에서 리덕스를 좀 더 간결하게 사용할 수 있도록 하는 최신 패키지

패키지 이름	설명
redux-devtools-extension	리덕스의 상태를 크롬브라우저 개발자도구에 설치된 확장 기능과 연동할 수 있게 해주는 미들웨어
chart.js	그래프 표시 라이브러리 (리액트와 직접적인 연관은 없음)
react-chartjs-2	리액트에 chart.js를 적용하기 위한 wrapper 라이브러리

```
yarn add react-router-dom react-helmet-async sass styled-components
styled-components-breakpoints dayjs axios react-loader-spinner axios-hooks
react-redux @reduxjs/toolkit redux-devtools-extension chart.js react-
chartjs-2
```

4) index.js

import 구문 추가

```
import { BrowserRouter } from 'react-router-dom';
import { Provider } from 'react-redux';
import store from './store';
```

index.js 파일에서 `<App />`을 아래와 같이 변경

```
<Provider store={store}>
    <BrowserRouter>
        <App />
    </BrowserRouter>
</Provider>
```

5) App.js

```
import { NavLink, Routes, Route } from "react-router-dom";
```

혹은

```
import { Link, Routes, Route } from "react-router-dom";
```

6) redux 준비

store.js 템플릿

```
import { configureStore } from '@reduxjs/toolkit';

const store = configureStore({
    reducer: {

    },
    middleware: (getDefaultMiddleware) =>
getDefaultMiddleware({serializableCheck: false}),
    devTools: true
});

export default store;
```

slice 템플릿

```
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit'
import axios from 'axios';

export const ? = createAsyncThunk("?", async (payload, { rejectWithValue
}) => {
    let result = null;

    try {
        result = await axios.get('?');
    } catch (err) {
        result = rejectWithValue(err.response);
    }

    return result;
});

const ? = createSlice({
    name: '?',
    initialState: {
        data: null,
        loading: false,
        error: null
    },
    reducers: {},
    extraReducers: {
        [?.pending]: (state, { payload }) => {
            return { ...state, loading: true }
        },
        [?.fulfilled]: (state, { payload }) => {
            return {
                data: payload?.data,
                loading: false,
                error: null
            }
        },
```

```
        [?.rejected]: (state, { payload }) => {
            return {
                data: payload?.data,
                loading: false,
                error: {
                    code: payload?.status ? payload.status : 500,
                    message: payload?.statusText ? payload.statusText :
'Server Error'
                }
            }
        }
    },
});

export default .reducer;
```

7) 컴포넌트에서 redux 사용하기

필요한 기능 참조하기

```
// 상태값을 로드하기 위한 hook과 action함수를 dispatch할 hook 참조
import { useSelector, useDispatch } from 'react-redux'
// Slice에 정의된 액션함수들 참조
import { 함수1, 함수2 } from '../slices/MySlice';
```

컴포넌트 내부에서 hook을 통해 필요한 Object 생성

```
// hook을 통해 slice가 관리하는 상태값 가져오기
const {변수1, 변수2} = useSelector((state) => state.slice별칭);

// dispatch 함수 생성
const dispatch = useDispatch();
```

필요한 이벤트 핸들러 안에서 액션함수 디스패치하기

Slice에서 정의한 액션함수의 `action.payload` 파라미터로 전달된다.

다수의 파라미터가 필요한 경우 JSON객체로 묶어서 전달한다.

```
dispatch(액션함수1(파라미터));
dispatch(액션함수2(파라미터));
```

8) 프로젝트 실행

프로젝트를 VSCode로 열고, `Ctrl` + `~`를 눌러 터미널 실행

```
yarn start
```