10-news-viewer

#01. 뉴스 Open API 키 발급받기

> https://newsapi.org/

#02. 프로젝트 생성

```
yarn create react-app 10-news-viewer
```

1) 추가 패키지 설치

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

지금까지 살펴본 기본 패키지

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

#03. 프로젝트 초기화

1) 불필요한 파일 삭제

1. **src폴더** 하위에서 App.css와 index.css, logo.svg 삭제
2. **App.js** 파일에서 App.css와 logo.svg에 대한 참조(import) 구문 제거
3. **index.js** 파일에서 index.css에 대한 참조(import) 구문 제거

2) `/src/store.js`

폴더와 파일을 직접 생성해야 함.

```
import { configureStore, getDefaultMiddleware } from '@reduxjs/toolkit';
import { createLogger } from 'redux-logger';

// Slice 오브젝트 참조 구문 명시 위치

const logger = createLogger();

const store = configureStore({
    reducer: {
```

```
        // 개발자가 직접 작성한  Slice 오브젝트들이 명시되어야 한다.
    },
    middleware: [
        ...getDefaultMiddleware({ serializableCheck: false }),
        createLogger()
    ],
    devTools: true
});

export default store;
```

3) /src/index.js

패키지 참조

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

랜더링 처리

```
ReactDOM.render(
    <React.StrictMode>
        <Provider store={store}>
            <BrowserRouter>
                <App />
            </BrowserRouter>
        </Provider>
    </React.StrictMode>,
    document.getElementById('root')
);
```

4) /src/app.js

패키지 참조

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

혹은

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

Route 처리

```
<Routes>
    <Route path='url정의' element={<컴포넌트명/>} />
</Routes>
```

링크걸기

```
<NavLink|NavLink to="url정의">링크텍스트</NavLink|Nav>
```

5) Slice 모듈 작성

a) 동기처리를 수행하는 경우

```
import { createSlice } from '@reduxjs/toolkit'

const slice이름 = createSlice({
    name: 'slice별칭',
    initialState: {
        // 이 모듈이 관리하고자하는 상태값들을 명시
    },

    reducers: {
        액션함수1: (state, action) => {...state},
        액션함수2: (state, action) => {...state}
    },
});

export const { 액션함수1, 액션함수2 } = slice이름.actions;
export default slice이름.reducer;
```

b) 비동기처리를 수행하는 경우

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

/** 비동기 처리 함수 구현 */
export const 액션함수 = createAsyncThunk("액션함수별칭", async (payload, {
rejectWithValue }) => {
    let result = null;

    try {
        result = await axios.get(URL및 파라미터);
    } catch (err) {
```

```
            result = rejectWithValue(err.response);
        }

        return result;
    });

    /** Slice 정의 (Action함수 + Reducer의 개념) */
    const slice이름 = createSlice({
        // slice 별칭
        name: 'slice별칭',
        // 상태값 구조 정의
        initialState: {
            rt: null,           // HTTP 상태 코드(200,404,500 등)
            rtmsg: null,        // 에러메시지
            item: [],           // Ajax 처리를 통해 수신된 데이터
            loading: false      // 로딩 여부
        },
        // 내부 action 및 동기 action
        reducers: {},
        // 외부 action 및 비동기 action
        extraReducers: {
            [액션함수.pending]: (state, { payload }) => {
                return { ...state, loading: true }
            },
            [액션함수.fulfilled]: (state, { payload }) => {
                return {
                    ...state,
                    rt: payload.status,
                    rtmsg: payload.statusText,
                    item: payload.data,
                    loading: false
                }
            },
            [액션함수.rejected]: (state, { payload }) => {
                return {
                    ...state,
                    rt: payload?.status ? payload.status : '500',
                    rtmsg: payload?.statusText ? payload.statusText : 'Server
    Error',
                    item: payload?.data,
                    loading: false,
                }
            }
        },
    });

    export default slice이름.reducer;
```

`/src/store.js` 파일에 정의한 Slice 모듈 명시

```
import { configureStore, getDefaultMiddleware } from '@reduxjs/toolkit';
import { createLogger } from 'redux-logger';

import slice이름 from './slices/MySlice';

const store = configureStore({
    reducer: {
        slice별칭: slice이름
    },
    middleware: [
        ...getDefaultMiddleware({ serializableCheck: false }),
        createLogger()
    ],
    devTools: true
});

export default store;
```

6) 컴포넌트에서 사용하기

a) 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';
```

b) 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();
```

c) 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));
```