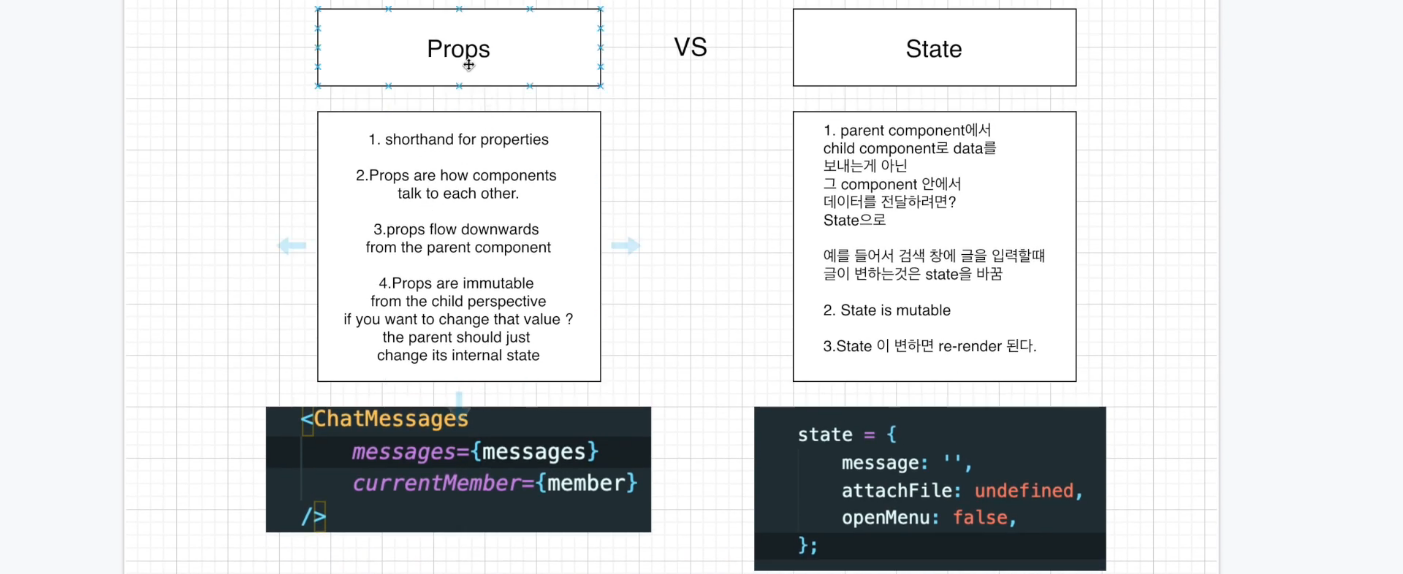
**1. 리덕스의 개념은 무엇이고 왜 쓰는가**

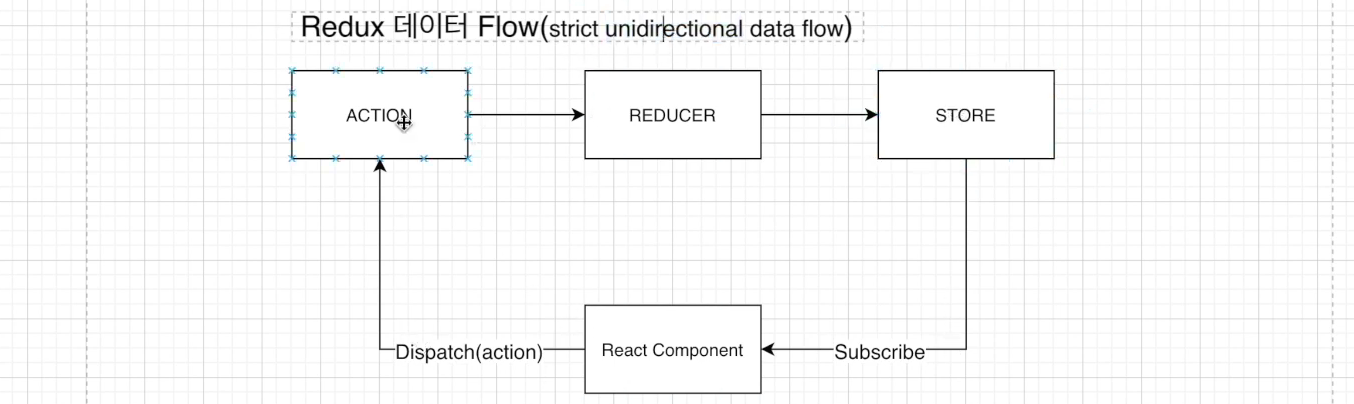
### Redux

- predictable state container for JS apps

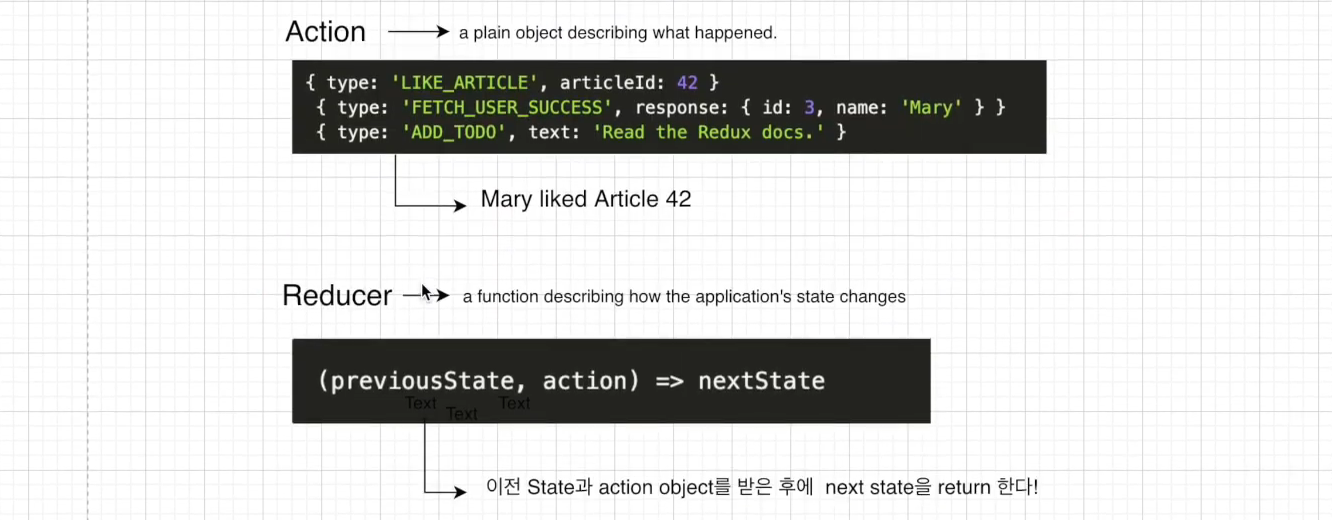
- state 상태 관리 library



|  |  |
| --- | --- |
| https://k.kakaocdn.net/dn/Ep14P/btqE2kZC0VD/TsLJIKCKVYgJyH6Hb27NP1/img.png출처: https://www.liberaldictionary.com/redux/ | https://k.kakaocdn.net/dn/bCJdnU/btqE3edzLmc/ykWMeewE9kQVnnDw2Iw9xK/img.png |
| - 예를 들어 댓글을 관리하는 component 중 하위 component에서 변경이 일어나면 그 변경된 값이 상위 component를 타고 올라가야 한다. 그러나 redux를 이용하면 외부의 store를 통해 변경된 지점에서 직접 정보를 주고받을 수 있다. | |
|  | |



- 한 방향으로 흐른다.



**Action** : 무엇이 일어났는지 설명하는(상태를 알려주는) 객체

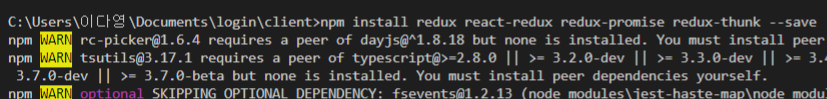
**Reducer** : 기존state값이 action으로 인해 다른 state 값으로 바뀌었다는 것을 설명해주는 곳

**Store**

- 전체적인 application의 state를 감싸주는 역할

- store 내의 여러 method를 통해 state을 관리할 수 있다.

**redux 설치**

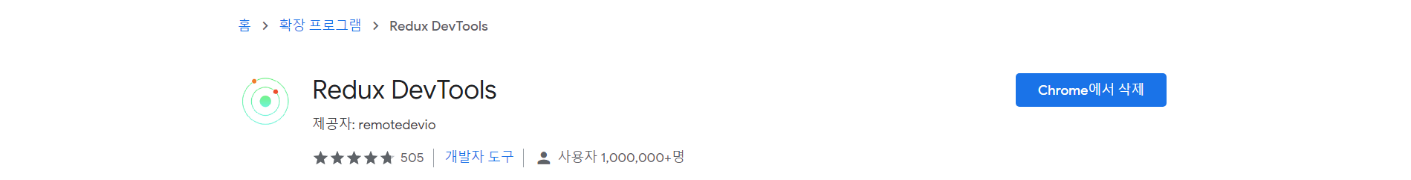


npm install redux react-redux redux-promise redux-thunk --save

- action이 object(객체)형식 이어야 store가 받을 수 있다. 그런데, store에서 언제나 object 형태의 action만 받는 것은 아니다. promise , function 형태로도 온다. 그럴 때 redux의 middleware인 **Redux-promise / Redux-thunk** 가 필요하다.

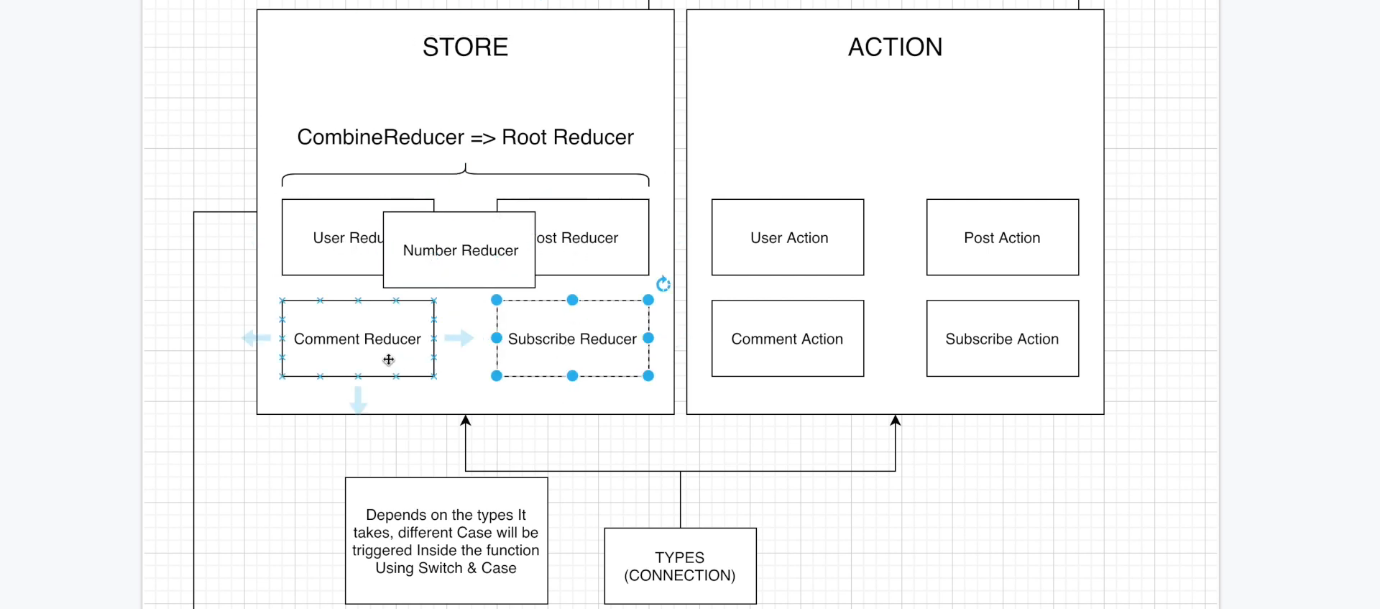
**Redux-promise**: dispatch에게 어떻게 promise를 받는지 알려준다.

**Redux-thunk :**dispatch에게 어떻게 function을 받는지 알려준다.



 - chrome webstore에서 확장 프로그램 다운로드

**CombineReducer**



login > client > src > index.js 코드

**import** React **from** 'react';

**import** ReactDOM **from** 'react-dom';

**import** './index.css';

**import** App **from** './App';

**import** \* **as** serviceWorker **from** './serviceWorker';

**import** { Provider } **from** 'react-redux';

*// redux에서 제공하는 Provider로 redux를 우리 App에 연결*

**import** 'antd/dist/antd.css';

**import** { applyMiddleware, createStore } **from** 'redux';

*// applyMiddleware, createStore method를 redux에서 가져옴.*

**import** promiseMiddleware **from** 'redux-promise';

**import** ReduxThunk **from** 'redux-thunk';

**import** Reducer **from** './\_reducers';

*// src/\_reducers/index.js에 정의한 reducer을 가져옴*

**const** createStoreWithMiddleware = applyMiddleware(promiseMiddleware, ReduxThunk)(createStore)

*// object뿐 아니라 promise, function을 모두 받기 위해 redux-promise, redux-thunk을 가져와 redux에서 store생성*

ReactDOM.render(

<Provider

store={createStoreWithMiddleware(Reducer,

window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_ &&

window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_()

)}

// 위에서 만든 store를 store에 넣었다.

// web에서 받은 redux 확장프로그램 삽입

>

<App />

</Provider>

*// App을 Provider로 감쌈.*

, document.getElementById('root')

);

*// If you want your app to work offline and load faster, you can change*

*// unregister() to register() below. Note this comes with some pitfalls.*

*// Learn more about service workers: https://bit.ly/CRA-PWA*

serviceWorker.unregister();

login > client > src > \_reducers > index.js 코드

**import** { combineReducers } **from** 'redux';

*// store 안에는 user reducer, comment reducer 등 수많은 reducer들이 있다.*

*// combine reducer는 그것들을 묶어 root reducer에서 하나로 합쳐준다.*

*// import user from './user\_reducer';*

**const** rootReducer = combineReducers({

*// user, comment*

})

**export** **default** rootReducer;

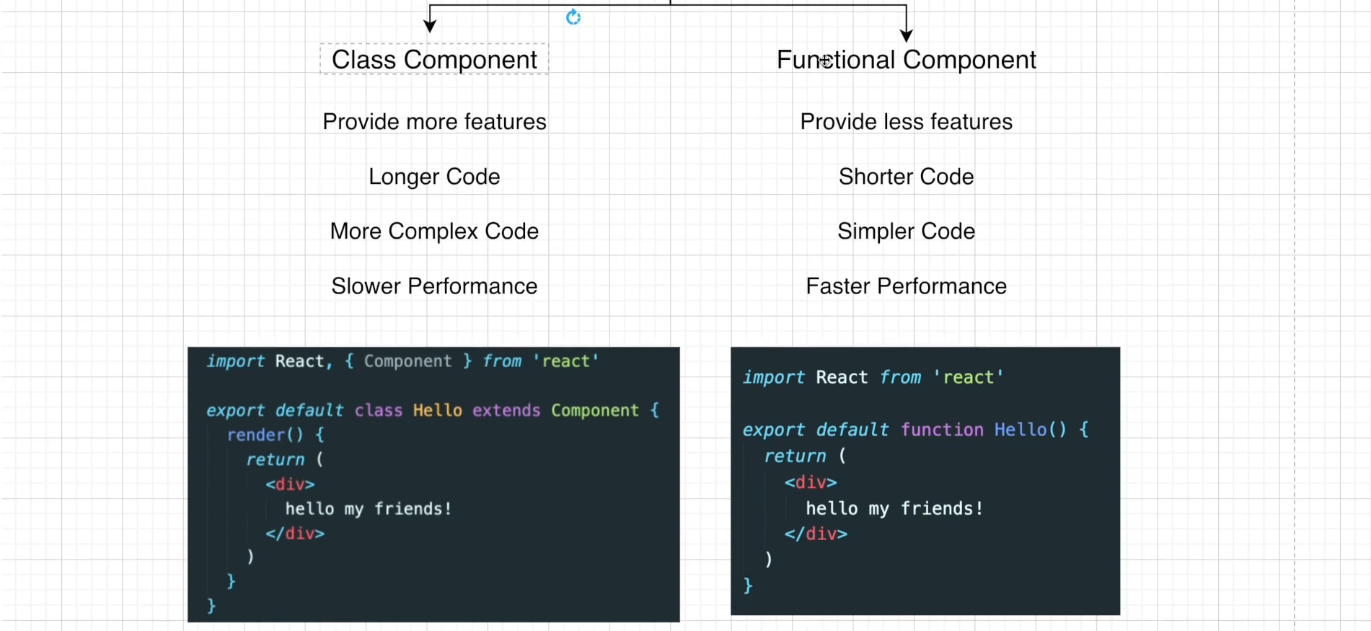
**참고**

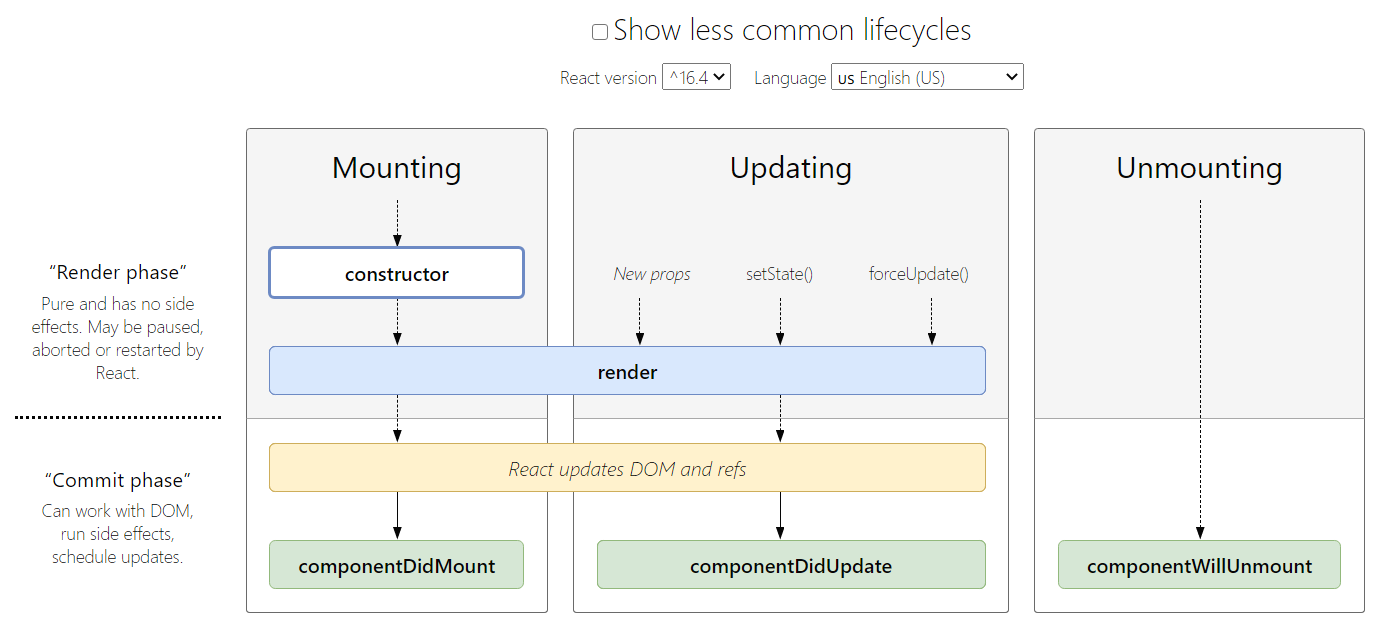
강의: [youtu.be/dJC\_uAR7d60](https://youtu.be/dJC_uAR7d60)

칼럼: <https://www.liberaldictionary.com/redux/>

**2. hook 과 mobx는 무엇이고 왜 쓰는가**

**Class / Functional Component**



출처: https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

**Hook**

**- React Hooks** are functions that let us **hook** into the **React** state and lifecycle features from function components.

- React Hooks are currently available in the **alpha** release of [react@16.7.0-alpha.0](mailto:react@16.7.0-alpha.0).

- function component에서는 원하는 기능을 함수로 만든 후 필요한 곳에 넣어 주기만 하면 되기 때문에 로직의 재사용이 유연해진다.

- class component가 가지고 있던 복잡성, 재사용성의 단점까지 해결된다.

**예제코드**

|  |  |  |
| --- | --- | --- |
| **Class Component** |  | **Functional Component** |
| https://k.kakaocdn.net/dn/IhSZl/btqE135bOiH/vXCzQHDl0kc4akgO3GHvWK/img.png | | |
| componentDidMount() | → | useEffect() |
| this.state | → | usestate |

[**MobX**](https://github.com/mobxjs/mobx)

- 프론트엔드를 위한 어플리케이션 상태 관리 라이브러리

- 주로 React와 함께 사용되며, 상태 관리 라이브러리라는 특성 때문에 종종 Redux와 비교된다. Redux가 함수형 아이디어를 적용했다면 MobX는 반응형 아이디어를 적용했다. MobX에서는 반응형 프로그래밍 패러다임에서 필수적으로 언급되는 옵저버(Observer) 패턴을 적극적으로 사용한다.

- 다만, MobX에서는 오직 상태(데이터)만 관찰 가능하며, 상태가 변경되었을 때 반응한다. 또한 상태를 변경시키는 것도 직접해야 한다. 한마디로 말해, MobX는 상태를 Observable하게 관리할 수 있도록 돕는 라이브러리다.

**참고**

강의 : [youtu.be/L\_mN8n6FHDg](https://youtu.be/L_mN8n6FHDg)

칼럼 : <https://hackernoon.com/react-hooks-usestate-using-the-state-hook-89ec55b84f8c>

[React Hooks: useState(using the state hook) | Hacker Noon](https://hackernoon.com/react-hooks-usestate-using-the-state-hook-89ec55b84f8c" \t "_blank)

[In this article, we will attempt to understand what is React Hooks and rules we have to follow when using them. Plus we will create a counter application using useState React Hook. By this, we mean that hooks allow us to easily manipulate the state of our](https://hackernoon.com/react-hooks-usestate-using-the-state-hook-89ec55b84f8c" \t "_blank)

[hackernoon.com](https://hackernoon.com/react-hooks-usestate-using-the-state-hook-89ec55b84f8c" \t "_blank)

<https://overreacted.io/ko/how-are-function-components-different-from-classes/>

[함수형 컴포넌트와 클래스, 어떤 차이가 존재할까?](https://overreacted.io/ko/how-are-function-components-different-from-classes/" \t "_blank)

[전혀 다른 '포켓몬'이라고 할 수 있다.](https://overreacted.io/ko/how-are-function-components-different-from-classes/" \t "_blank)

[overreacted.io](https://overreacted.io/ko/how-are-function-components-different-from-classes/" \t "_blank)

<https://bravenamme.github.io/2020/04/08/react-hook/>

**3. webpack과 babel은 무엇이고 왜 쓰는가**

**Babel**

- 최신 JavaScript 문법을 지원하지 않는 브라우저들을 위해 최신 JavaScript 문법을 구형 브라우저에서도 돌 수 있게 (ES5 문법으로) 변환

**Webpack**

- 많은 모듈들을 합하여 간단하게 만들어 준다. 번들시킨다.

- src 폴더 안을 관리 / public 폴더는 관리하지 않는다.

( ex) image 파일을 넣을 때 src 폴더 안에 넣어야 webpack이 관리할 수 있다.)

