Redux

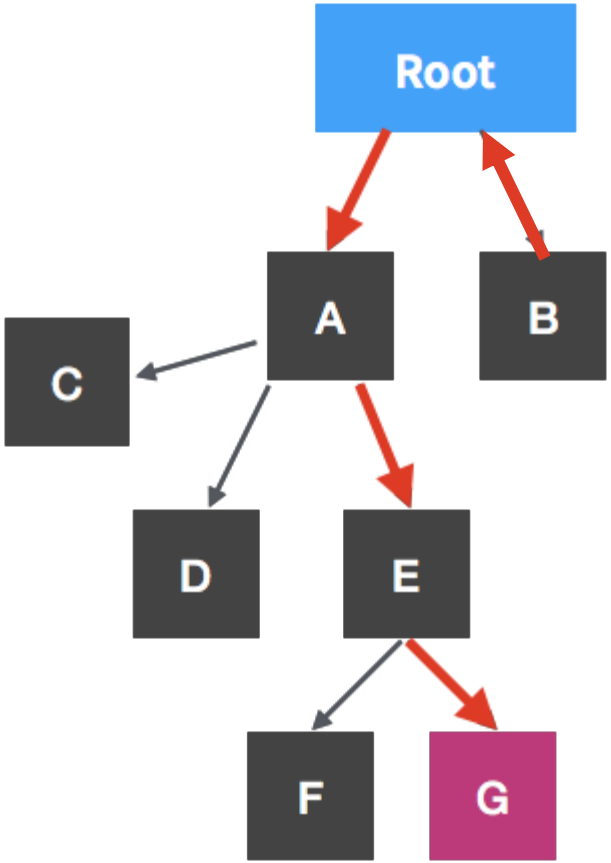
소프트웨어 개발의 원리와 실습

2019. 03. 21

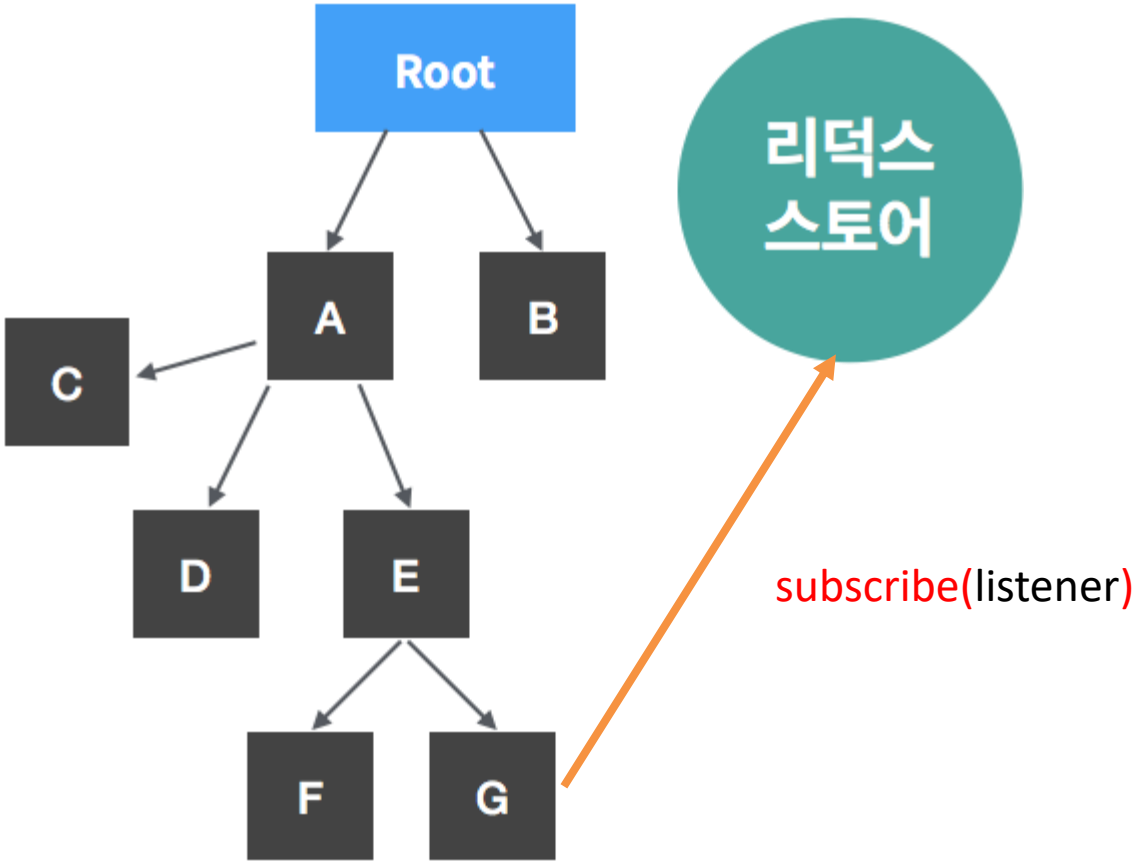
Redux!

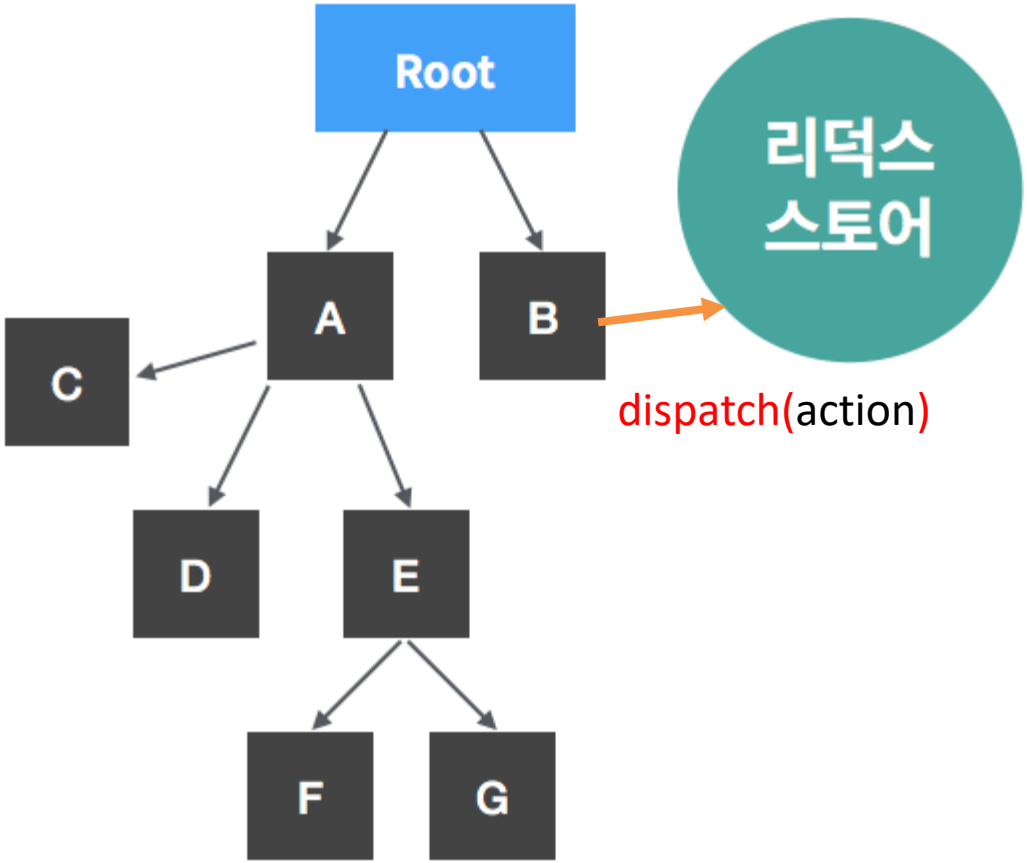
- 자바스크립트로 만들어진 상태 관리 라이브러리
- 로직을 따로 분리시켜 관리를 용이하게!

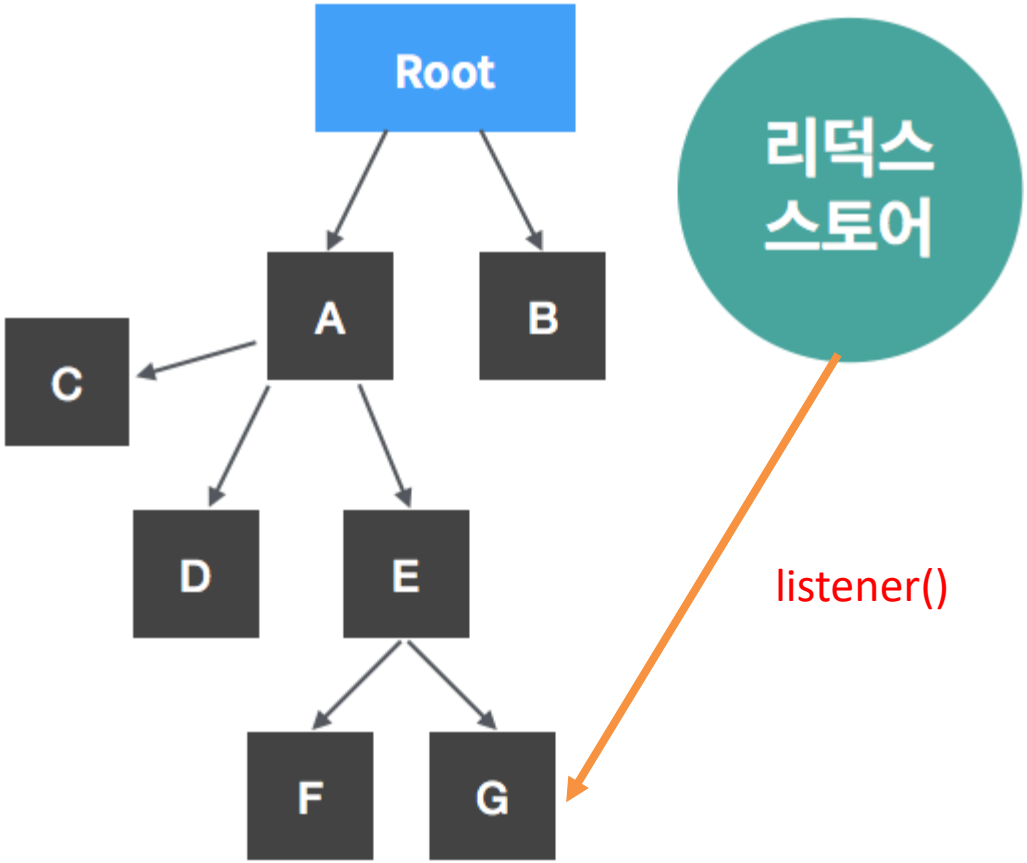
- https://redux.js.org/
- https://deminoth.github.io/redux/ (한글 번역)



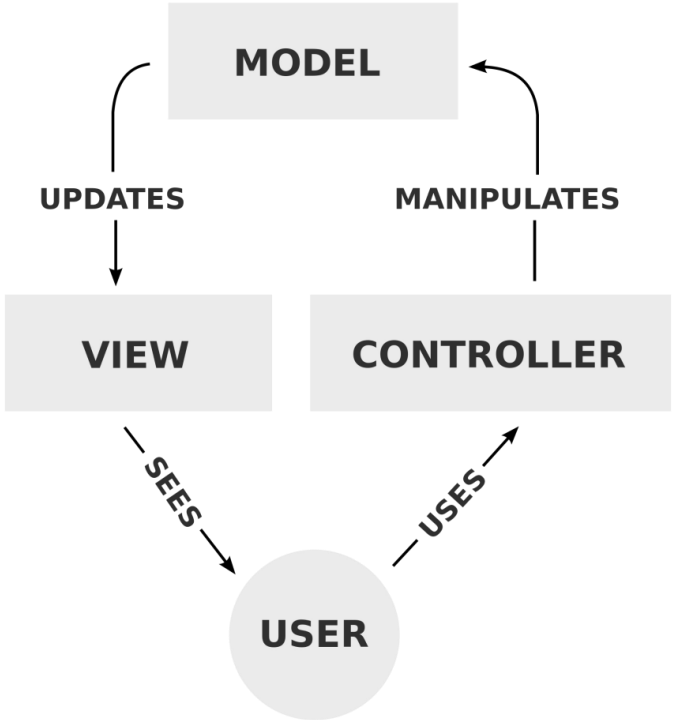

```
// B 에서 callback 호출
this.props.callbackFromParent(myCallback)

// App.js 에서 A 렌더링
<A value={5}>

// A.js 에서 E 렌더링
<E value={this.props.value} />

// B.js 에서 G 렌더링
<G value={this.props.value} />
```

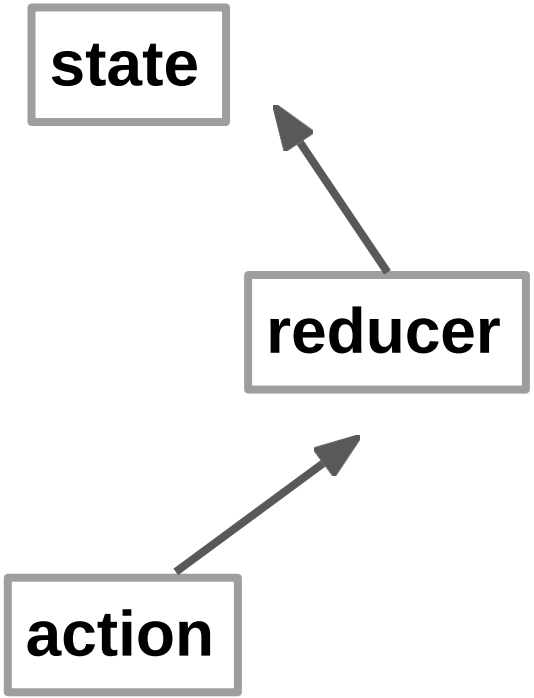

Root
A
B
C
D
E
F
G
리덕스
스토어
subscribe(listener)


Root
A
B
C
D
E
F
G
리덕스
스토어
dispatch(action)


Root
A
B
C
D
E
F
G
리덕스
스토어
listener()

Redux!


MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER


state
reducer
action

Action

- 어플리케이션에서 스토어로 보내는 데이터 묶음
- type과 그 외에 필요한 정보 포함
- store.dispatch(action) 사용하여 정보 전달

```
{
  type: ADD_TODO,
  text: 'Build my first Redux app'
}
```

Reducer

- (prevstate, action) => nextstate
- dispatch로 넘겨온 action이 state를 어떻게 변화시키는지 서술

```
function todoApp(state = initialState, action) {
  switch (action.type) {
  case ADD_TODO:
    return Object.assign({}, state, {
      todos: [...state.todos, {
        text: action.text,
        completed: false
      }]
    });
  default:
    return state;
  }
}
```

Store

- Action과 Reducer를 함께 가진 객체
  - state 저장 및 접근
  - dispatch(action)를 통해 state 수정
  - subcribe(listener)로 listener 등록

Redux의 세 가지 원칙

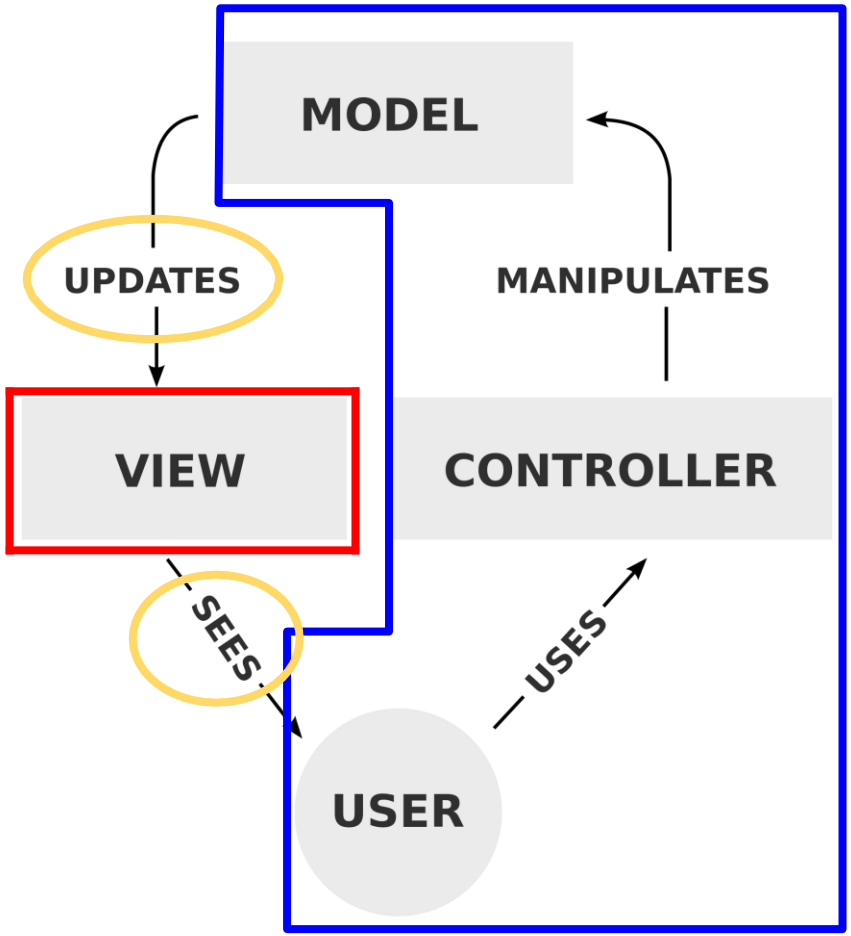
1. 모든 state를 하나의 store 안에
2. state는 읽기만, 바꾸려면 action을 통해
3. reducer는 pure function이다(side effect가 없다!)
    - state × action -> next state

React-Redux

React

Redux

React-Redux



React-Redux

	Presentational 컴포넌트	Container 컴포넌트
목적	어떻게 보여질 지 (마크업, 스타일)	어떻게 동작할 지 (데이터 불러오기, 상태 변경하기)
Redux와 연관됨	아니오	예
데이터를 읽기 위해	props에서 데이터를 읽음	Redux 상태를 구독
데이터를 바꾸기 위해	props에서 콜백을 호출	Redux 액션을 보냄

https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0

Redux middleware

- action을 reducer에서 처리하기 전에 작업
- 비동기 작업, side effect 처리 => redux-saga
- https://redux-saga.js.org/