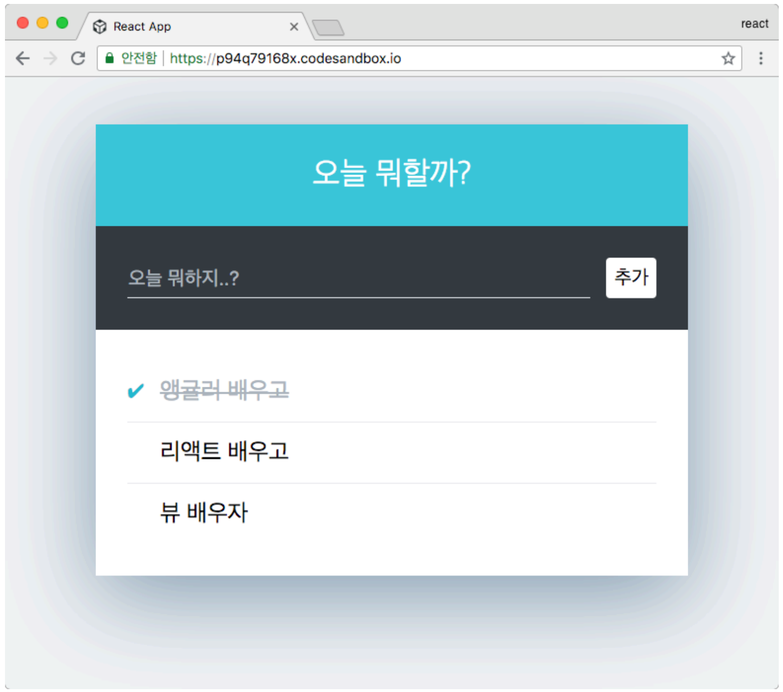
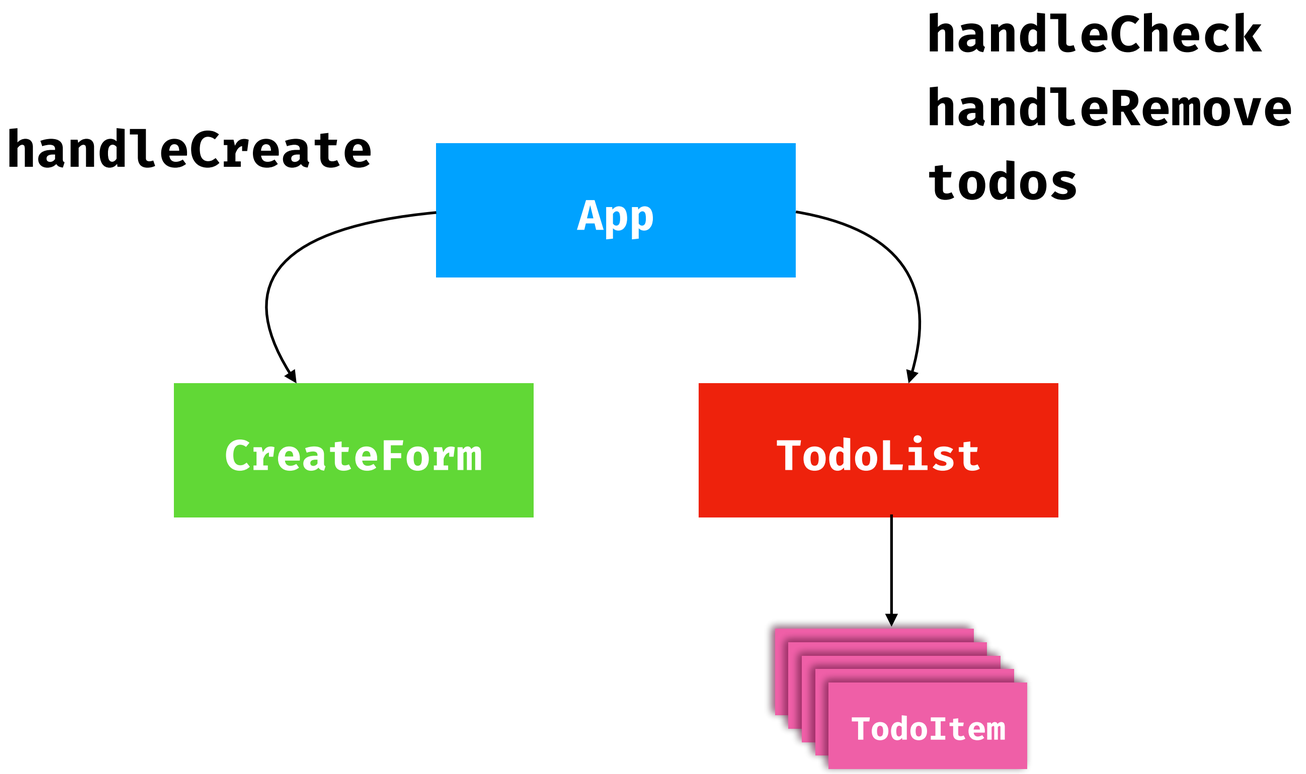
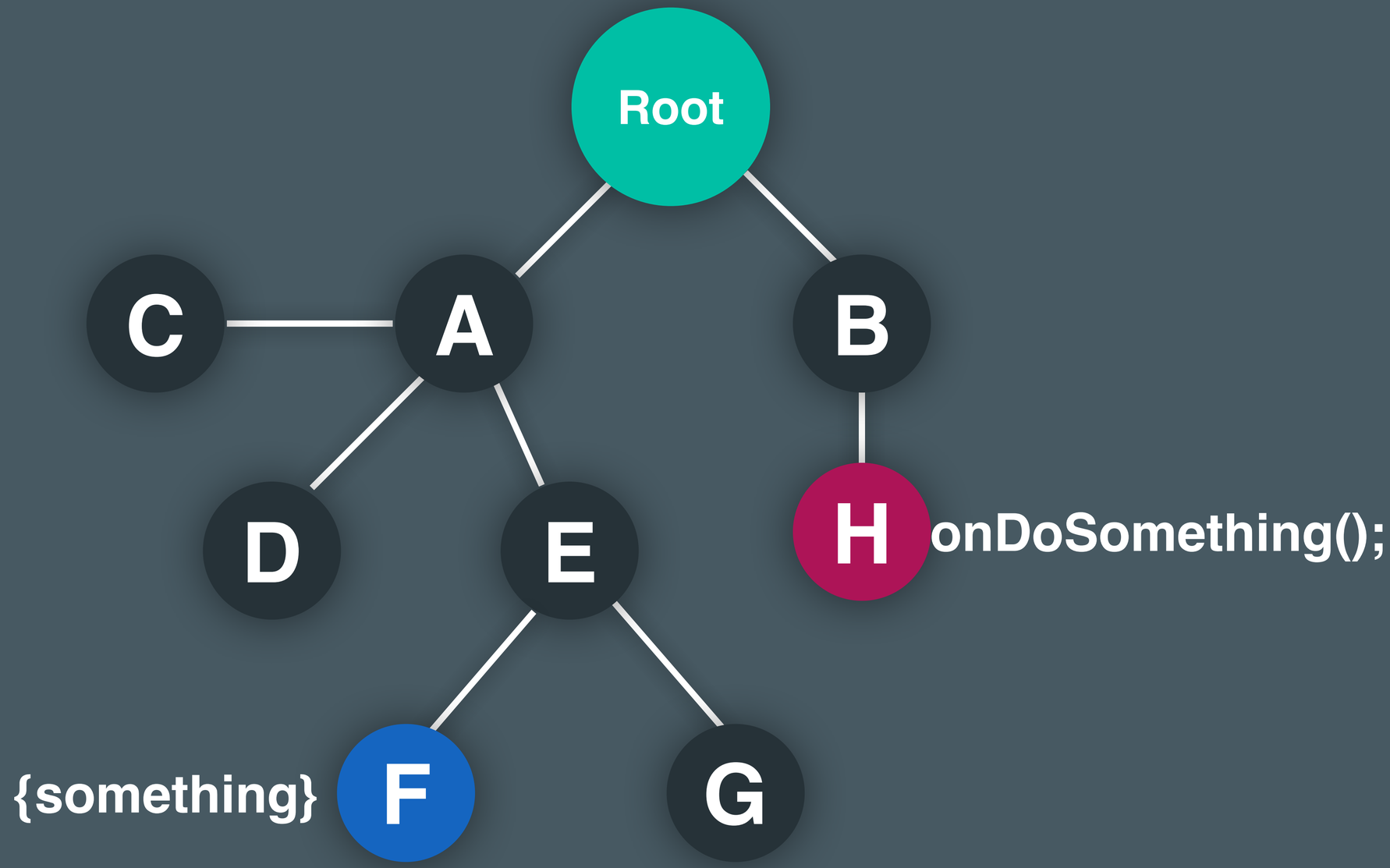
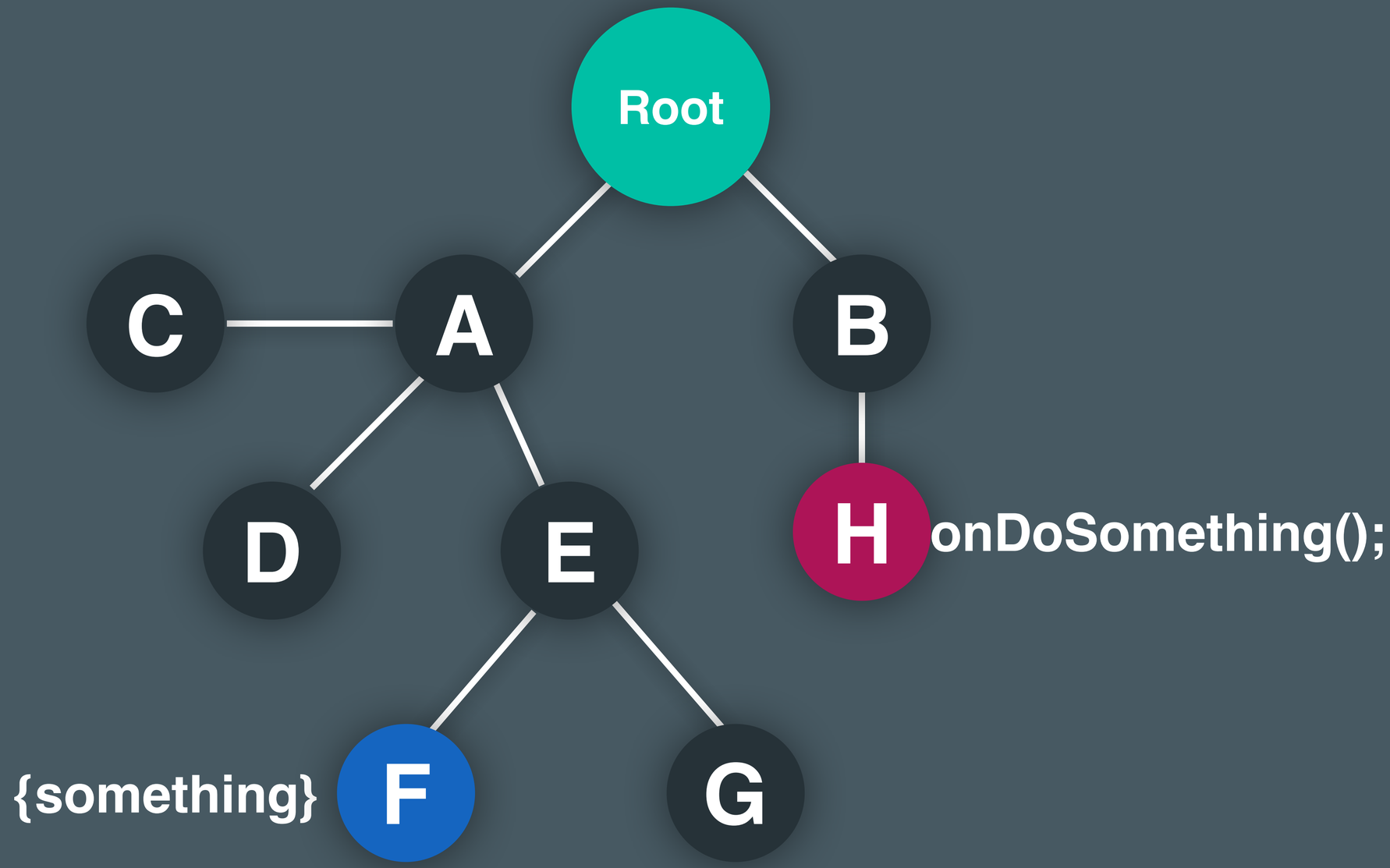
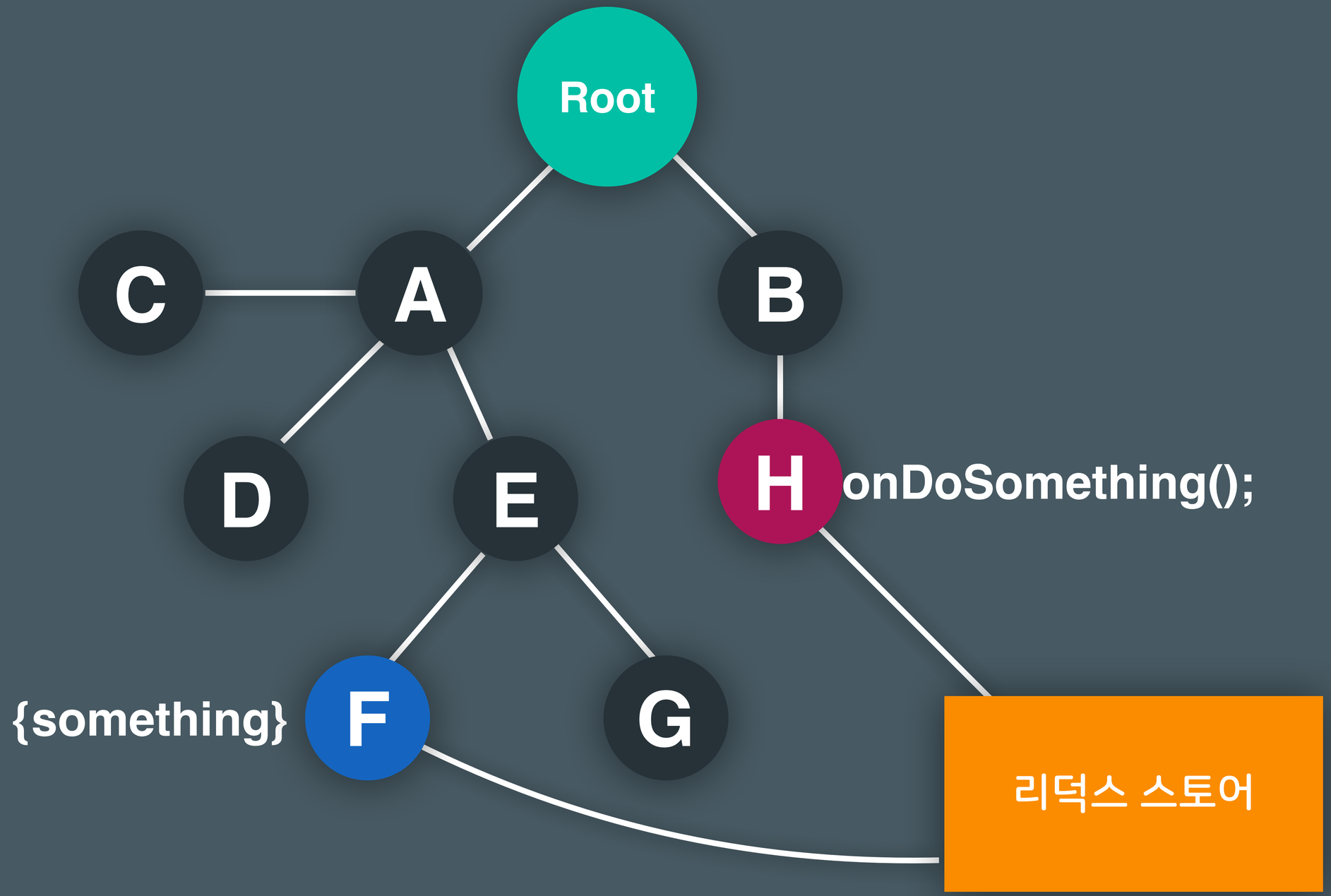
리액트 상태 관리 라이브러리
Redux / MobX


React App
react
안전함 | https://p94q79168x.codesandbox.io
오늘 뭐할까?
오늘 뭐하지..?
추가
앵귤러 배우고
리액트 배우고
뷰 배우자


handleCreate
handleCheck
handleRemove
todos
App
CreateForm
TodoList
TodoItem


Root
C
A
B
D
E
H
onDoSomething();
{something}
F
G


Root
C
A
B
D
E
H
onDoSomething();
{something}
F
G


Root
C
A
B
D
E
H
onDoSomething();
{something}
F
G
리덕스 스토어

상태관리 라이브러리를 사용하는 이유

1. 손 쉬운 글로벌 상태 관리
2. 컴포넌트에서 업데이트 관련 로직을 분리
3. 용이한 테스팅

라이브러리 따로 안써도, 리액트 16.3^ 버전에선
새로워진 Context API 로 글로벌 상태 조금 더 쉽게 관리 가능

언제 Redux 대신 Context 를 사용해야 할 까?

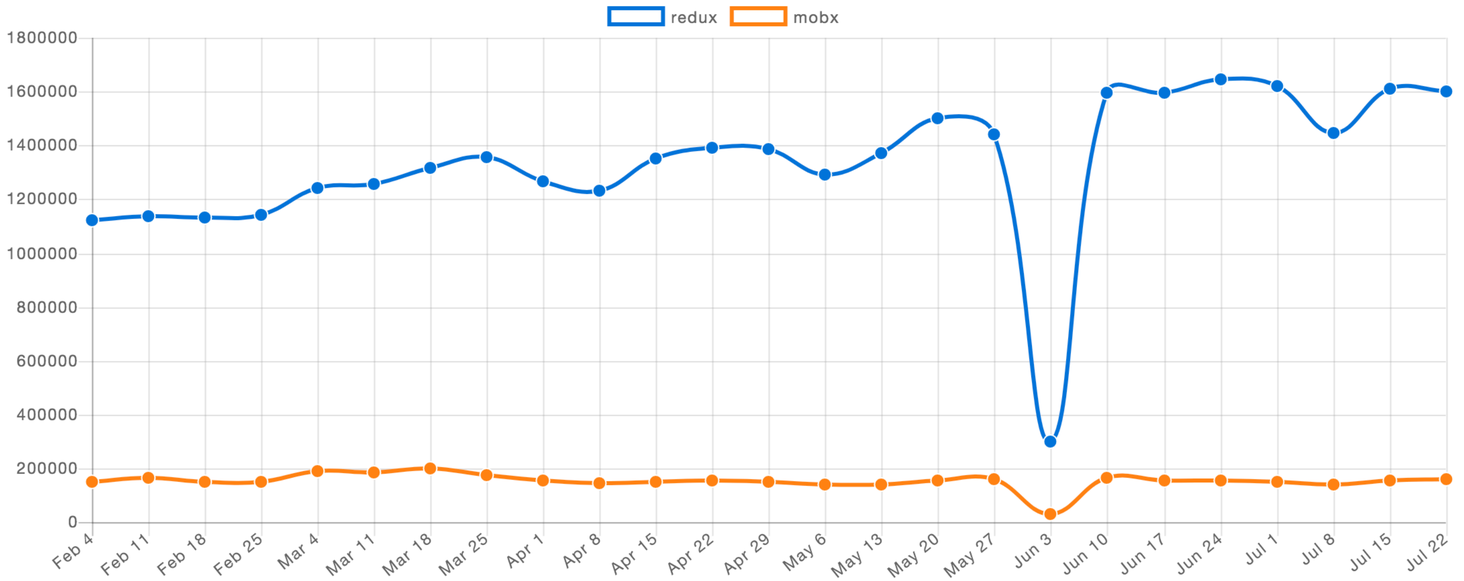
다음과 같은 이유로 쓰는 경우:

- 단순히 글로벌 상태를 사용하기 위하여
- 단순히 깊숙한곳에 있는 컴포넌트에 props 를 전달하기 위하여

Redux 나 MobX 에는 훨씬 많은 기능이 있다.

Redux & MobX

Downloads in past 6 Months



Github Stats

	stars 🌟	forks 🍽	issues ⚠	updated 🛠	created 🐣
redux	42893	10548	34	Jul 24, 2018	May 30, 2015
mobx	16086	933	53	Jul 26, 2018	Mar 14, 2015

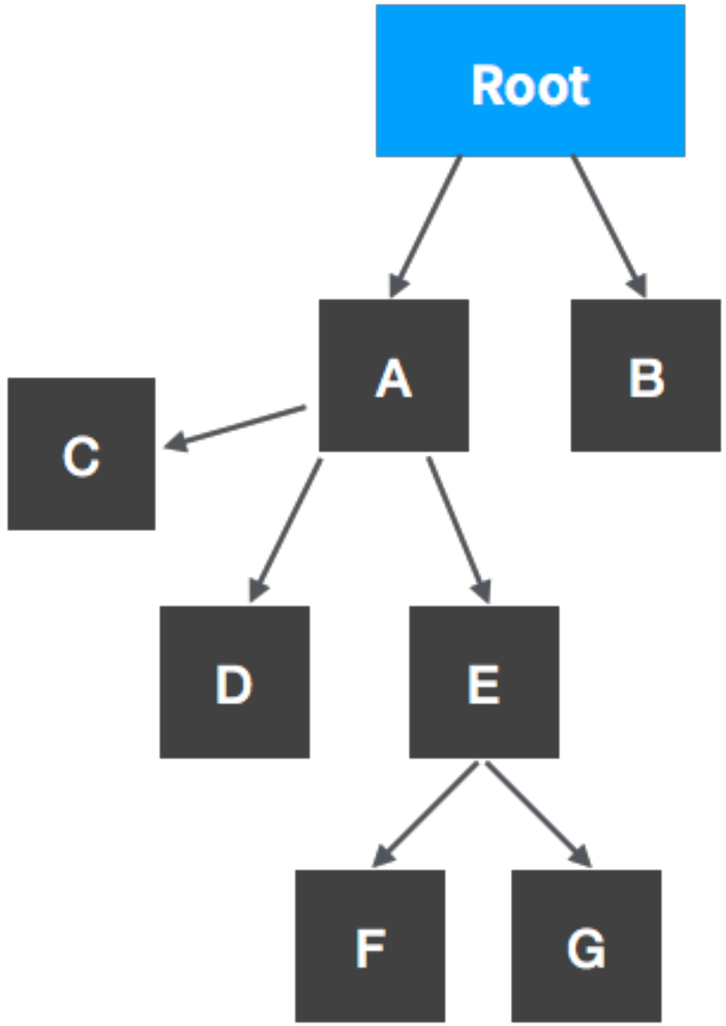
Redux

- 사용률 훨씬 높음
- 좋은 관습 만들어져있음
- 리액트스러움
- 함수형 프로그래밍
- 불변성 유지함
- 상태 관리의 정석?
- 진입장벽 있음 (이 강의에선 없음)

MobX

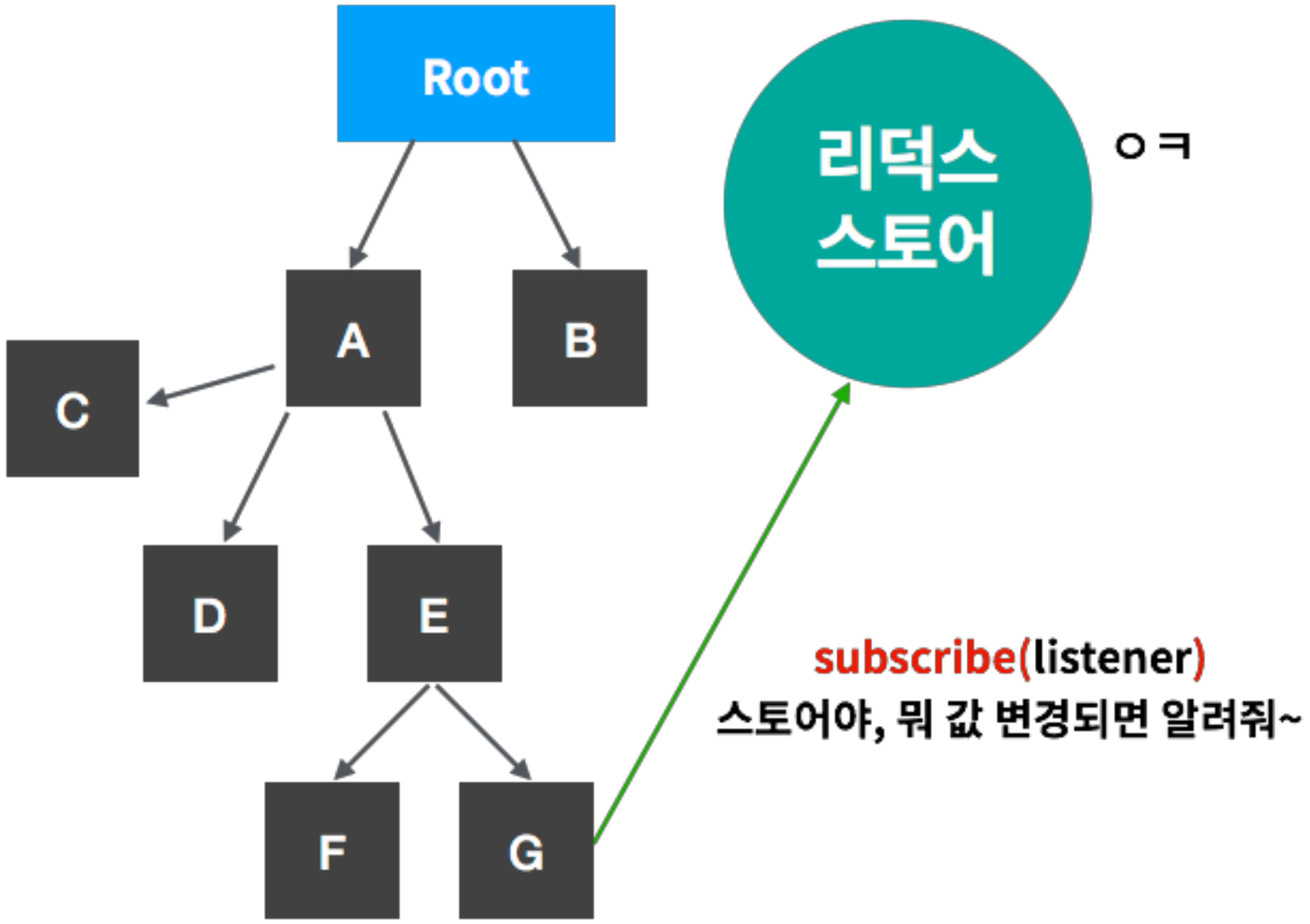
- 리액트에 흑마법 끼얹은 느낌
- 다른 프레임워크 쓰는 느낌
- 불변성 필요없음
- 몇가지 규칙 따라주면 자동 최적화
- 예제 프로젝트 많진 않음
- 배우기엔 엄청나게 쉬움
- 사용하기도 굉장히 쉬움
- 진입장벽 적음
- 신선함
- 늦게 배우면 조금 후회함

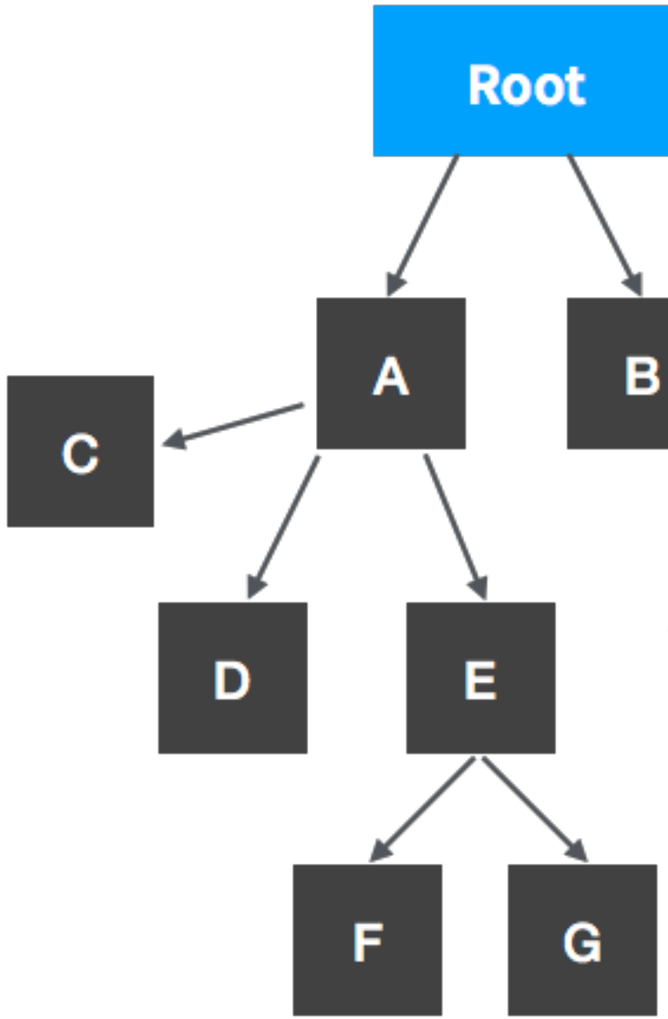
둘 다 사용해보자!



리덕스
스토어

B 에서 호출된 것이
G 에 영향을 끼치는 상황


Root
A
B
C
D
E
F
G
리덕스
스토어
ㅇㅋ
subscribe(listener)
스토어야, 뭐 값 변경되면 알려줘~



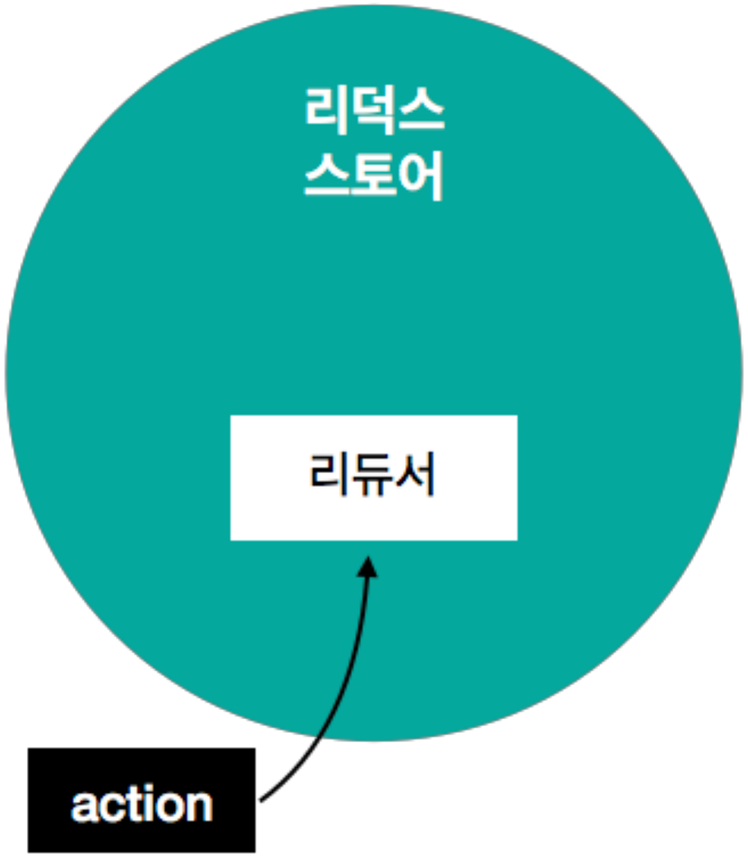
dispatch(action)

상태 업데이트 해줘.

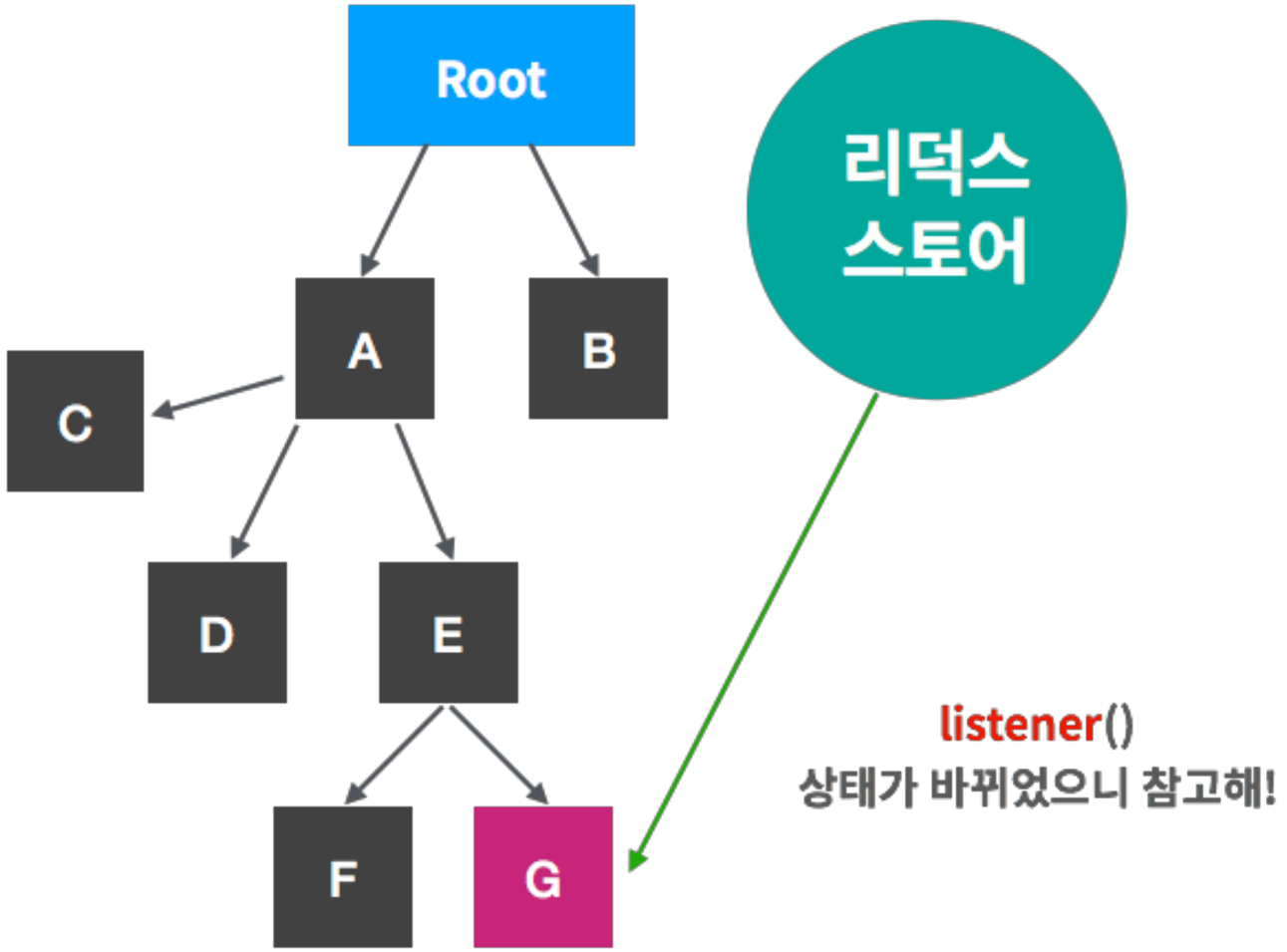
업데이트 할땐 내가 준 데이터 참고 하고~

{ type: 'INCREMENT' }

{ type: 'INCREMENT', diff: 2 }




```
function reducer(state, action) {
   return updatedState;
}
```



오우, 새로우 상태? 그러면 리렌더링 할게

Immutable.js

```
// 1. 객체는 Map
const obj = Map({
  foo: 1,
  inner: Map({
    bar: 10
  })
});
```

```
// 2. 배열은 List
const arr = List([
  Map({ foo: 1 }),
  Map({ bar: 2 }),
]);
```

```
// 3. 설정할땐 set
let nextObj = obj.set('foo', 5);
```

```
// 4. 값을 읽을땐 get
console.log(obj.get('foo'));
```

```
// 5. 읽은다음에 설정 할 때는 update
// 두번째 파라미터로는 updater 함수가 들어감
nextObj = nextObj.update('foo', value => value + 1);
```

```
// 6. 내부에 있는걸 ~ 할땐 In 을 붙인다
nextObj = obj.setIn(['inner', 'bar'], 20);
let nextArr = arr.setIn([0, 'foo'], 10);
```

```
// 7. List 내장함수는 배열이랑 비슷하다
nextArr = arr.push(Map({ qaz: 3 }));
nextArr = arr.filter(item => item.get('foo') === 1);
```

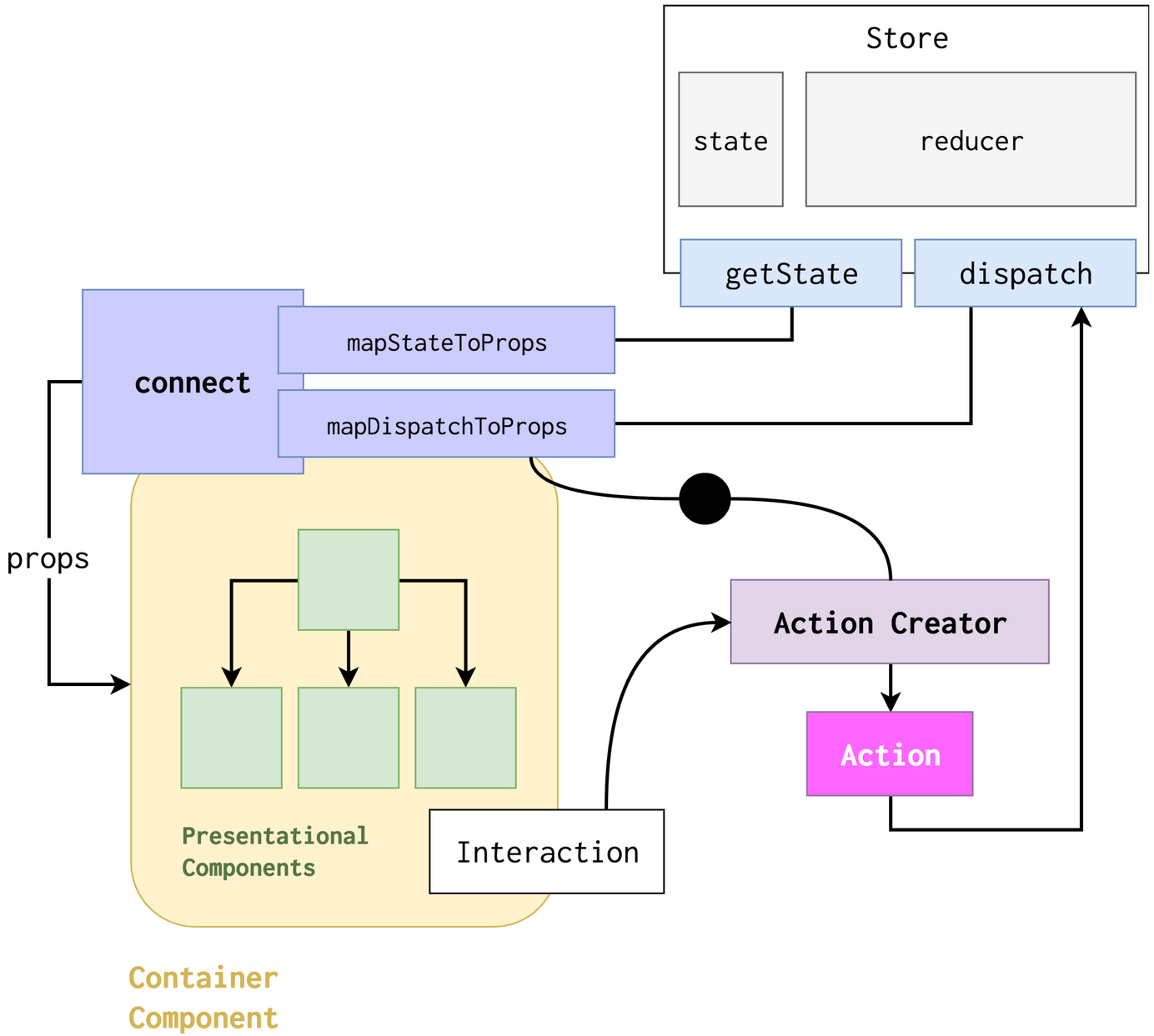
Immer.js

```
import produce from "immer"

const baseState = [
    {
        todo: "Learn typescript",
        done: true
    },
    {
        todo: "Try immer",
        done: false
    }
]

const nextState = produce(baseState, draftState => {
    draftState.push({ todo: "Tweet about it" })
    draftState[1].done = true
})
```

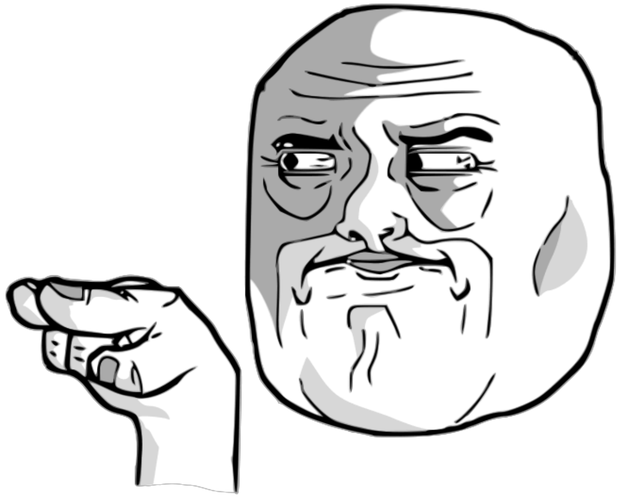
```
// 8. delete 로 key 를 지울 수 있음
nextObj = nextObj.delete('foo');
nextArr = nextArr.delete(0);
```


Store
state
reducer
getState
dispatch
connect
mapStateToProps
mapDispatchToProps
props
Action Creator
Action
Presentational
Components
Interaction
Container
Component

MobX 주요 개념

1. Observable State

State



http://worldartsme.com/i-am-watching-you-clipart.html#gal_post_74835_i-am-watching-you-clipart-1.jpg

2. Computed Value

연산된 값을 캐싱해서 사용하기

3. Reactions

상태가 바뀌면 특정 작업 하기!

4. Actions

상태에 변화를 일으키는 것
리덕스의 액션 객체랑 다름
그냥 액션 = 상태 바꾸는 코드

Middleware 는 왜 필요할까?