Ask Company

리액트와 함께 장고 시작하기 / 리액트

상탯값 (state)

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

상탯값 (state)

UI (엘리먼트)로의 반영을 위해, 유지해야할 값들의 묶음

상탯값은 어디에 저장/관리되나요?

- 각 컴포넌트 내에서만 사용되는 값들은 컴포넌트 안에서 생성/갱신 → 리액트 기본 기본으로 동작
- 여러 컴포넌트에서 사용되는 값들은 별도 공간에서 생성/갱신 → 이때 Redux, Context API, MobX 등을 활용하면 편리

컴포넌트에서 상탯값에 대한 getter/setter 함수를 제공해줍니다.

- 상탯값을 직접 변경하진 말아요. → 성능 하락. 강제 UI 업데이트를 할 순 있지만 ... ☹

클래스형 컴포넌트에서의 상탯값

```
import React from 'react';


class Counter extends React.Component {
  state = {
    counter: 0,
  };

  onClick = () => {
    this.state.counter += 1;

    const counter = this.state.counter + 1;
    this.setState({ counter });
  }

  render() {
    const { counter } = this.state;
    return (
      <p onClick={this.onClick}>
        Counter : {counter}
      </p>
    );
  }
}

export default Counter;
```

절대 이렇게 상탯값을 변경하지 않아야 합니다.
모든 상탯값/속성값은 불변값으로 처리하셔야 합니다.

클래스형 컴포넌트에서는 항상 ...

1. this.state 객체를 통해 상탯값에 접근
2. this.setState 함수를 통해 상탯값을 변경

setState (1/3)

ReactComponent.setState(클래스형 컴포넌트에서 제공되는 유일한 상탯값 setter
    객체 또는 함수,
    처리가_끝났을_때_호출되는_콜백_함수
)

비동기로 동작

변경할 특정 state값들이 담긴 object를 지정하거나,

함수를 지정 가능 → 매개변수로 호출되기 직전 상태값을 받는다. → 선호

setter에 지정된 함수에서 상탯값을 직접 참조하고 있지 않아도, "직전 상탯값"을 인자로 전달받기에 유용.

immer 라이브러리와 엮어 쓰기에도 좋습니다.

setState를 2회 호출하지만, 비동기 동작이기에 count값은 1만 증가

```
onClick1 = () => {
  this.setState({ count: this.state.count + 1 });
  this.setState({ count: this.state.count + 1 });
}
```

setState가 비동기로 동작하지만, 함수로 지정했기에 직전 상태값을 받습니다. → 2 증가

```
onClick2 = () => {
  this.setState(prevState => ({count: prevState.count + 1}));
  this.setState(prevState => ({count: prevState.count + 1}));
}
```

setState (2/3)

상탯값 로직을 컴포넌트에서 분리하는 패턴도 가능합니다.

아. 이렇게도 할 수 있구나. 정도로만 보세요.

```
// 상탯값을 유지하지는 않고, 루틴만 제공합니다.

const actions = {

  // 초기 상탯값을 제공만 합니다.
  init() {
    return {count: 0};
  },

  // 직전 상탯값을 인자로 받아, count 값을 1증가시킨 객체를 반환
  increment(prevState) {
    return {count: prevState.count + 1};
  },

  // 직전 상탯값을 인자로 받아, count 값을 1감소시킨 객체를 반환
  decrement(prevState) {
    return {count: prevState.count - 1};
  },

};
```

```
class App extends React.Component {
  state = actions.init();

  render() {
    return (
      <div>
        Count: {this.state.visit}
        <button onClick={() => this.setState(actions.increment)}>
          증가
        </button>
        <button onClick={() => this.setState(actions.decrement)}>
          감소
        </button>
      </div>
    );
  }
};
```

상탯값을 변경할 함수를 인자로 넘깁니다.

setState (3/3)

setState를 통하지 않고 state 값을 직접 변경하면, UI에 자동 반영 X

UI에 강제반영하기 위해 this.forceUpdate(); 를 사용

클래스형 컴포넌트 생성자에서는 setState 호출은 ...

무시됩니다.

setState 호출은 컴포넌트가 마운트된 이후에만 유효하기 때문

데이터를 가져오기 위해 API를 호출하고 그 응답을 state에 반영코자 할 경우에

- componentDidMount 메서드를 활용
- 함수형 컴포넌트에서는 useEffect(() => {}, []) 훅을 활용

상탯값을 어디에 저장/관리할 것인가?

1) 컴포넌트 내부

각 컴포넌트 객체 단위로 상탯값을 유지하고, 하위 컴포넌트의 속성값으로 전달

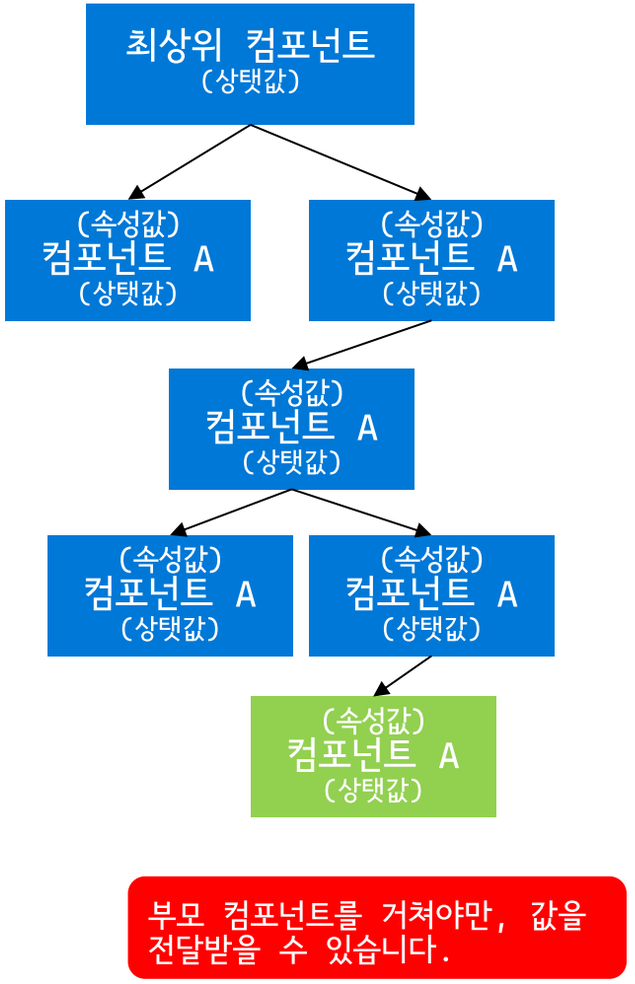
상탯값 setter 함수를 통해 상탯값을 직접 변경 (물론 Reducer 개념을 적용할 수도)

Context API와 함께 할 때 강력해집니다.

this.setState 함수, useState 훅을 활용

문제점 : 컴포넌트 계층이 복잡할 때, 상탯값/속성값 전파의 번거로움.

하지만, 리액트 16.3에 소개된 Context API로 전파가 쉬워졌습니다.



2) 컴포넌트 외부에서 "전역 상태 관리"

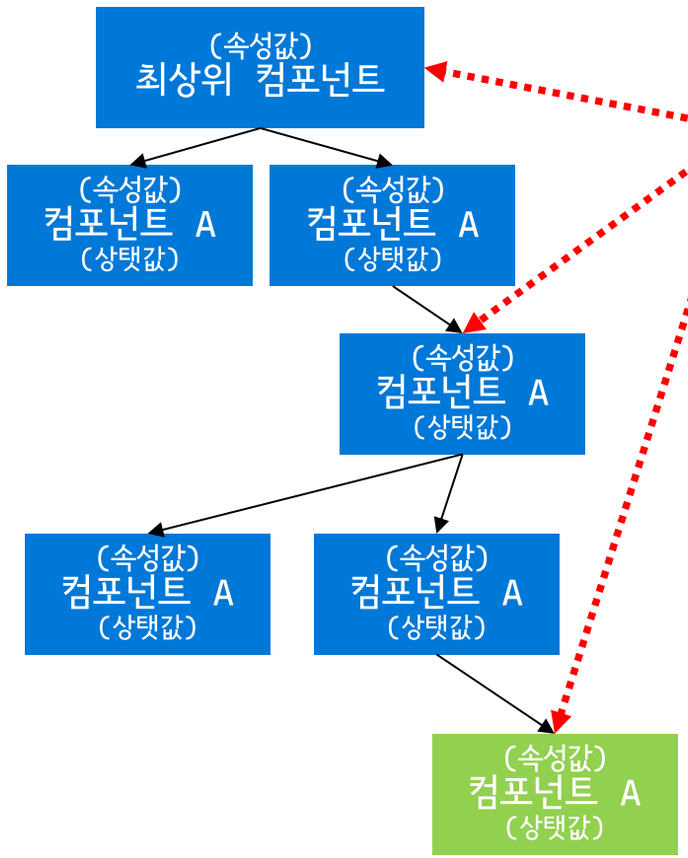
컴포넌트 외부에 별도의 상탯값 저장소를 둡니다.

여러 컴포넌트들에 의해 공유될 상탯값 (로그인 정보 등)들을 관리합니다.

모든 상탯값들을

컴포넌트에게 상탯값 setter 함수를 제공하기보다, dispatch 함수를 제공합니다.

State Reducer 패턴



상탯값 저장소

Redux를 사용할 때의 패턴

1. 상탯값에 대한 직접적인 변화를 가하는 setter 제공 X
2. 대신 하나의 함수에서 임의의 객체를 전달받아 (이벤트 개념), 이를 상탯값에 반영
   - 흔히 dispatch 함수, action 객체라 칭합니다.
3. 현 애플리케이션에서 **공유할** 상탯값들을 한 곳에서 관리

```
state = {
  value: 0,
};
```

setter함수를 사용할 경우

```
setState(prevState => ({
  value: prevState.value + 1,
}));

setState(prevState => ({
  value: prevState.value + 3,
}));
```

dispatch함수가 제공되는 경우

```
function dispatch(action, state) {
  const { type, payload } = action;
  if ( type === 'INCREMENT' ) {
    const { value } = payload;
    return {
      ...state,
      value: state.value + value,
    }
  }
  else {
    return state;
  }
}

const action = {type: 'INCREMENT', payload: {value: 1}};
dispatch(action);

dispatch({
  type: 'INCREMENT',
  payload: { value: 3 },
});
```

reduce : 상탯값을 누적/적용해서 현재의 상탯값을 이뤄낸다는 의미

```
const numbers = [1, 2, 3, 4, 5];
const sum = numbers.reduce((acc, n) => acc + n, 0);
console.log(sum);  // 15
```

Context API를 활용할 경우

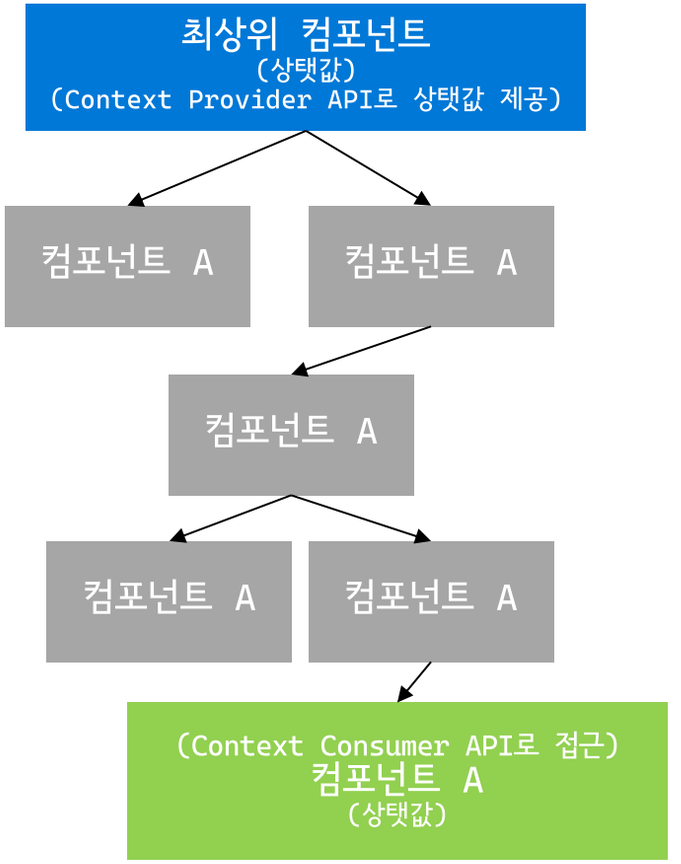
리액트 16.3부터 많은 개선

Context API를 사용하기 어렵던 시절에는 Redux만이 유일한 "전역 상태 관리" 방법

값을 공유하는 것만을 도와줄 뿐, 그 외의 기능은 없습니다.

useReducer 훅과 함께 State Reducer 패턴을 적용하기 좋습니다.

Redux 등의 라이브러리에서도 내부적으로 사용



리액트 개발의 핵심

어떻게 하면 상탯값을 효율적으로 잘 관리할 수 있을까?

상탯값에 따라 화면이 불필요하게 업데이트되지 않도록 하자.

Life is short.
You need Python and Django.

I will be your pacemaker.