사이드 에펙트 처리 (6) useReducer

1. useState 의 한계

복잡한 상태를 다루려면 여러 개의 snapshoot을 관리해야 한다, 그 과정에서 상태 변화가 꼬일 수도 있다.

```
const Login = (props) => {
    const[enteredEmail, setEnteredEmail] = useState(initialState: '');
    const[emailIsValid, setEmailIsValid] = useState();
    const[enteredPassword, setEnteredPassword] = useState(initialState: '');
    const[passwordIsValid, setPasswordIsValid] = useState();
    const[formIsValid, setFormIsValid] = useState(initialState: false);
}
```

대부분 정상적으로 동작하나,

```
const validdateEmailHandler = () => {
    setEmailIsValid(enteredEmail, includes('@'));
};
```

`validdateEmailHandler` 가 실행 될 때 아직 `enteredEmail` 이 값이 업데이트 안 되었을 수도 있다.

2. useReducer

> `useReducer` is a React Hook that lets you add a reducer to your component.

```
const [state, dispatch] = useReducer(reducer, initialArg, init?)
```

`dispatch` 는 단순하게 다른 로직을 추가해주거나 필요한 부분만 바꿔주거나 state를 관리할 수 있다. state가 조금 복잡해지면 `dispatch` 를 통해서 값을 입력받을 때, 값을 초기화 할때 등 여러 타입 별로 state를 어떻게 바꿔줄지 구분해줄 수 있다. 그런 의미로 `dispatch` 를 유용하게 사용할 수 있다.

`reducer` 함수는 state를 어떻게 바꿔줄건지 그 로직이 들어있는 공간이라고 이해하면 된다.
`initialArg` 는 초기값을 의미한다. (= 처음의 state 값) `init` 은 옵셔널한 값이다.

- **dispatch function**

: The dispatch function returned by `useReducer` lets you update the state to a different value and trigger a re-render. You need to pass the action(`({ type : 'incremmnted_age'});`) as the onlty argument to the `dispatch` function:

```
const [state, dispatch] = useReducer(reducer, { age: 42 });

function handleClick() {
    dispatch({ type : 'incremmnted_age'});
    //...
}
```

- **practice 4-6 (로그인 컴포넌트 위주)**

: state 로 되어있는 부분을 useReducer로 바꿔주기

```
const emailReducer = (state, action) => {
    //값이 바뀌는 경우. setEnteredEmail
    if(action.type === 'USER_INPUT'){
        //emailState
        return{
            value: action.val,
            isValid: action.val.includes('@')
        }
    }
    //유효성 검사를 해야 하는 경우, setEmailIsValid
    if(action.type === 'INPUT_BLUR'){
        return {
            value: state.value,
            isValid: state.value.includes("@"),
        };
    }
    return {
        value: '',
        isValid: null
    }
};
const passwordReducer = (state, action) => {
}

const Login = (props) => {
    const[formIsValid, setFormIsValid] = useState(false);
    const[emailState, dispatchEmail] = useReducer(emailReducer, {
        //초기값 선언
        value: '',
        isValid: null
    });
    const[passwordState, dispatchEmail] = useReducer(passwordReducer, {
        value: '',
        isValid: null
    });
}

//...

const emailChandeHandler = (event) => {
    dispatchEmail({type: 'USER_INPUT', val: event.target.value})

}
```

- 복잡한 상태 다룰 때 `useState` 로 하게 되면 side effect가 발생할 수 있다.

- useReducer는 하나의 복잡한 상태를 여러 타입으로 dispatch 하기에 적합한 훅이다.