**8장에서 발전. input이 여러 개일때 상태관리에 관한 챕터**

//InputSample.js

import React, {useState} from ‘react’;

function InputSample() {

const [inputs, setInputs] = useState( {

value:0, name:’’, nickName:’’

});

const {value, name, nickName} = inputs;

const onChange = (e) => {

const {value, name} = e.target;

setInputs({

…inputs, [name] : value

});

};

const onReset = () => {

setInputs({

value:0, name:’ ‘, nickName:’’

})

};

return (

<div>

<input name=”value” onChange={onChange} placeholder=”값” value={value} />

<input name=”name” onChange={onChange} placeholder=”이름” value={name} />

<input name=”nickName” onChange={onChange} placeholder=”닉네임” value={nickName} />

<div>

값 : {value} <br />

이름 :{name} <br/>

닉네임 : {nickName} <br/>

</div>

</div>

)

}

export default InputSample;

**설명**

-다음과 같이 useState를 사용하여 하나의 함수로 관리받는 다수의 변수를 선언할 수있음

const [inputs, setInputs] = useState({

value:0,

name:’’,

nickName:’’

});

-위처럼 다수의 변수를 inputs같이 하나의 변수에 넣어 사용할 때, 이후 코드들은 그 관계를 모르기에

다음처럼 따로 꺼내줘야함.

const {value, name, nickName} = inputs;

-input 마다 고유의 name을 지정해주어, 들어온 값의 구별을 할 수 있게 해준다.

<inpit name=”value” onChange={onChange} placeholder=”값” value={value} />

-name=”value” : 현재 input요소의 고유 이름. 사용할 변수 이름과 똑같이 한다.

-placeholder:워터마크로 표시되는 글자. 어떤 종류의 input인지 유저에게 알려주는 것

-value={value} : 위에서 따로 꺼내준 value에 유저가 입력한 value를 연계시키는 것.

-onChange함수는 다음처럼 한다.

-변화가 일어난 input요소의 name과 value를 추출함.

const {value, name} = e.target;

-기존의 요소를 그대로 복사하고, 그 다음에 위에서 추출한 name, value로 변화한 값을 수정함

setInputs({ …inputs, [name] : value});

**주의점**

-리액트에서 객체를 수정해야할 때는, inputs[name] = value 처럼 직접 수정하면 안된다.

-대신 setInputs({…inputs, [name]:value}); 처럼 **새로운 객체를 만들어서 새로운 객체에 변화를 주고,**

**이를 상태로 사용해주어야 한다.**

-이러한 작업을 “불변성을 지킨다”라고 부름. 이러한 불변성을 지켜주어야만 리액트 컴포넌트에서

상태가 업데이트 됐음을 감지할 수 있고, 이에따른 리렌더링이 진행됨.

-inputs[name] = value처럼 값을 직접바꾸면, 값은 바뀌지만 리렌더링이 일어나지 않음.

-또 이렇게 불변성을 지켜줘야만 컴포넌트 업데이트 성능 최적화를 제대로 할 수 있음.