9. 여러개의 input 상태 관리하기

지난 튜토리얼에서 우리는 input 상태를 관리하는 방법에 대하여 알아보았는데요, 이번에는 input 이 여러개일때는 어떻게 관리해야 하는지 알아보겠습니다.

우선 지난번에 만든 InputSample 에서 새로운 input 을 보여주세요.

이번에는 input 이 비어져있을 때 인풋에 대한 설명을 보여주는 `placeholder` 값도 설정해보겠습니다.
기존에 만들었던 상태는 지워주시고, `onChange` 와 `onReset` 함수는 비워주세요.

InputSample.js

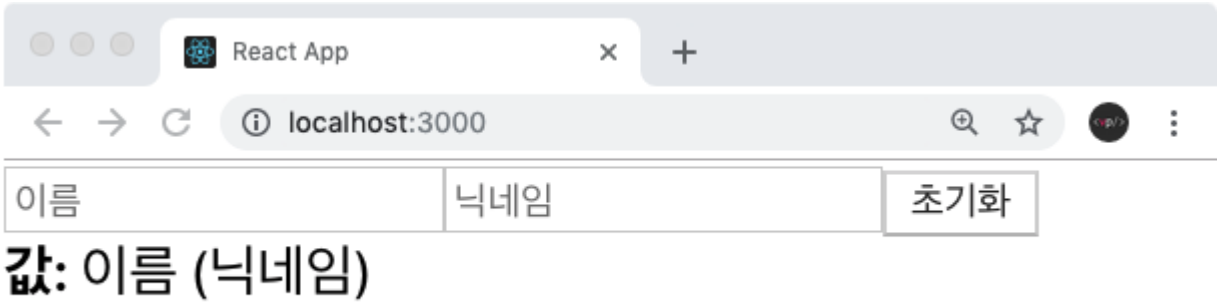
```
import React, { useState } from 'react';

function InputSample() {
  const onChange = (e) => {
  };

  const onReset = () => {
  };


  return (
    <div>
      <input placeholder="이름" />
      <input placeholder="닉네임" />
      <button onClick={onReset}>초기화</button>
      <div>
        <b>값: </b>
        이름 (닉네임)
      </div>
    </div>
  );
}

export default InputSample;
```



input 의 개수가 여러개가 됐을때는, 단순히 `useState` 를 여러번 사용하고 `onChange` 도 여러개 만들어서 구현 할 수 있습니다. 하지만 그 방법은 가장 좋은 방법은 아닙니다. 더 좋은 방법은, input 에 `name` 을 설정하고 이벤트가 발생했을 때 이 값을 참조하는 것입니다. 그리고, `useState` 에서는 문자열이 아니라 객체 형태의 상태를 관리해주어야 합니다.
한번 InputSample 컴포넌트를 다음과 같이 수정해보세요.

InputSample.js

```
import React, { useState } from 'react';

function InputSample() {
  const [inputs, setInputs] = useState({
    name: '',
    nickname: ''
  });

  const { name, nickname } = inputs; // 비구조화 할당을 통해 값 추출

  const onChange = (e) => {
    const { value, name } = e.target; // 우선 e.target 에서 name 과 value 를 추출
    setInputs({
      ...inputs, // 기존의 input 객체를 복사한 뒤
      [name]: value // name 키를 가진 값을 value 로 설정
    });
  };

  const onReset = () => {
    setInputs({
      name: '',
```

```
      nickname: '',
    })
  };


  return (
    <div>
      <input name="name" placeholder="이름" onChange={onChange} value={name} />
      <input name="nickname" placeholder="닉네임" onChange={onChange}
value={nickname}/>
      <button onClick={onReset}>초기화</button>
      <div>
        <b>값: </b>
        {name} ({nickname})
      </div>
    </div>
  );
}

export default InputSample;
```

리액트 상태에서 객체를 수정해야 할 때에는,

```
inputs[name] = value;
```

이런식으로 직접 수정하면 안됩니다.

그 대신에, 새로운 객체를 만들어서 새로운 객체에 변화를 주고, 이를 상태로 사용해주어야 합니다.

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

여기서 사용한 `...` 문법은 spread 문법입니다. 객체의 내용을 모두 "펼쳐서" 기존 객체를 복사해주는데요, 이 문법을 잘 모르신다면 이 링크 를 참고하세요. 이러한 작업을,"불변성을 지킨다" 라고 부릅니다. 불변성을 지켜주어야만 리액트 컴포넌트에서 상태가 업데이트가 됐음을 감지 할 수 있고 이에 따라 필요한 리렌더링이 진행됩니다. 만약에 `inputs[name] = value` 이런식으로 기존 상태를 직접 수정하게 되면, 값을 바꿔도 리렌더링이 되지 않습니다.
추가적으로, 리액트에서는 불변성을 지켜주어야만 컴포넌트 업데이트 성능 최적화를 제대로 할 수 있습니다. 컴포넌트 최적화에 대해서는 나중에 더 자세히 알아보도록 하겠습니다.

지금은 이것만 기억하세요. 리액트에서 객체를 업데이트하게 될 때에는 기존 객체를 직접 수정하면 안되고, 새로운 객체를 만들어서, 새 객체에 변화를 주어야 됩니다.