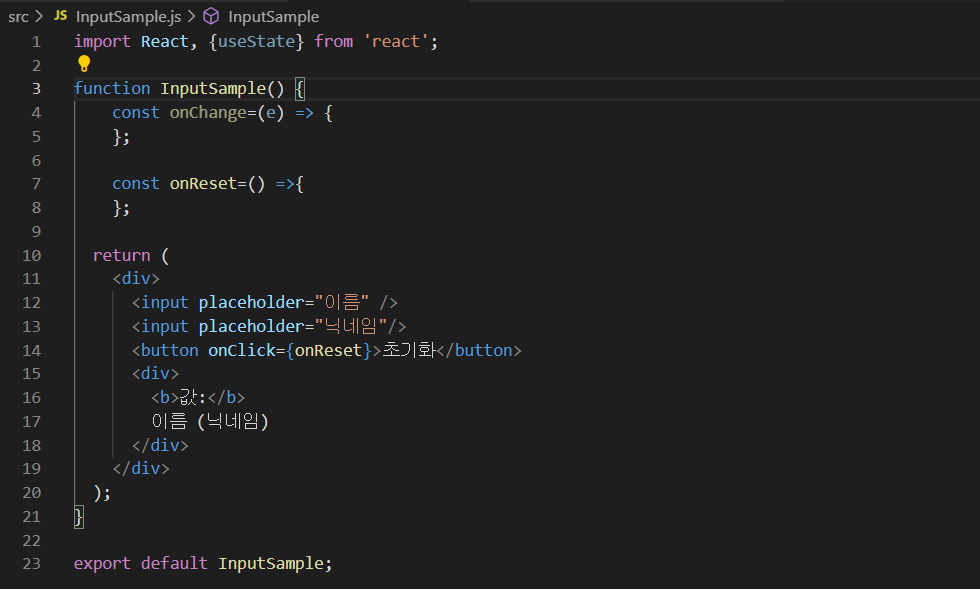
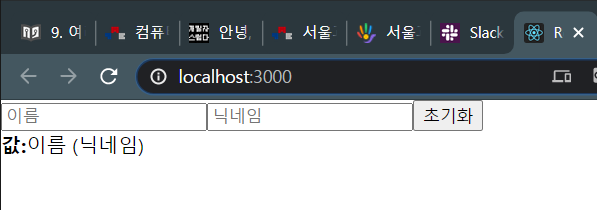
여러 개의 input 상태 관리하기

input이 비어져있을 때 인풋에 대한 설명을 보여주는 placeholder 값을 설정해보자.

onChange와 onReset 이벤트 함수의 내용은 비운다.



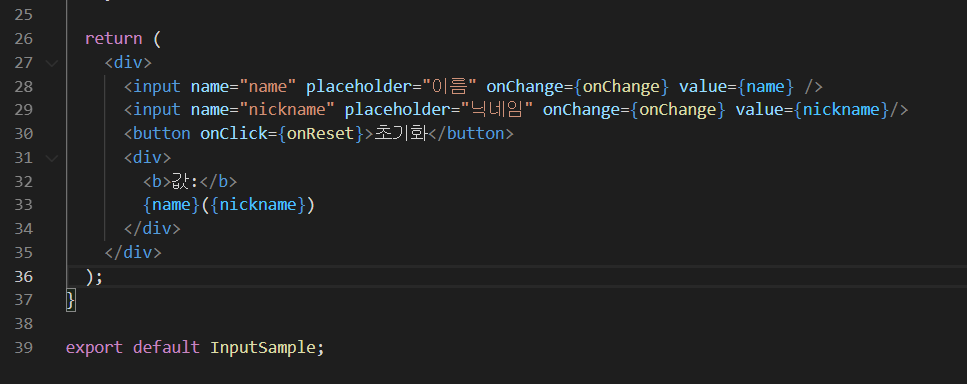


위와 같은 코드는 이름과 닉네임을 입력하여도 “값 : “ 의 형태는 바뀌지 않는다

input의 개수가 여러 개가 되었을 때는, 단순히 useState를 여러 번 사용하고 onChange를 여러 개 만들어서 구현 할 수 있지만, 그 방법이 가장 좋은건 아니다. 더 좋은 방법은, input에 name을 설정하고 이벤트가 발생했을 때 이 값을 참조하는 것이다. 그리고 useState에서는 문자열이 아니라 객체 형태의 상태를 관리해주어야 한다.

다음과 같이 수정해보자.

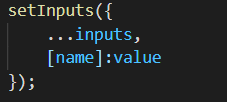




이 때 중요한 점은 리액트 상태에서 객체를 수정해야 할 때에는,

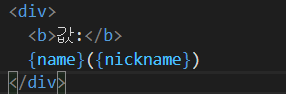
inputs[name]=value; 와 같은 방식으로 직접적으로 수정하면 안된다.

대신에, 새로운 객체를 만들어서 새로운 객체에 변화를 주고, 이를 상태로 사용해야 한다.

또한  여기서 사용된 … 문법은 spread 문법이다. 객체의 내용을 모두 “펼쳐서” 기존 객체를 복사해준다.

이러한 작업을, “불변성을 지킨다”라고 부른다. 불변성을 지켜주어야만 리액트 컴포넌트에서 상태가 업데이트가 되었음을 감지할 수 있고, 이에 따라 필요한 리랜더링이 진행된다.

또한, 리액트에서는 불변성을 지켜주어야만 컴포넌트 업데이트 성능 최적화를 제대로 할 수 있다.

 여기서 nickname에 ( ) 를 써주는 것은 다른 의미가 있는게 아니라 name과 ui적으로 구별하기 위해서 사용된 것이다. 다른 의미가 있는 줄 알았다.

리액트에서 객체를 업데이트하게 될 때에는 기존 객체를 직접 수정하면 안되고, 새로운 객체를 만들어서, 새 객체에 변화를 주여야 한다.

UseRef로 특정 DOM 선택하기

JavaScript를 사용할 때에는, 우리가 특정 DOM을 선택해야 하는 상황에 getElementById, querySelector 같은 DOM Selector 함수를 사용해서 DOM을 선택한다.

이 때 리액트에서는 ref라는 것을 사용한다.

**Ref를 사용해야 할 때**

바람직한 사용 사례는 다음과 같다.

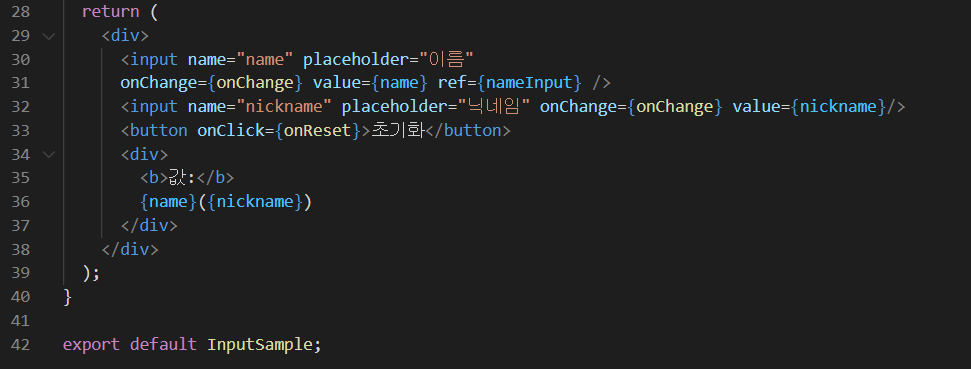
* 포커스, 텍스트 선택 영역, 혹은 미디어의 재생을 관리할 때
* 애니메이션을 직접적으로 실행시킬 때
* 서드 파티 DOM 라이브러리를 React와 같이 사용할 때

ref는 애필리케이션에 “어떤 일이 일어나게” 할 때 사용될 수도 있지만 만약 선언적으로 해결될 수 있는 문제라면 ref 사용을 지양하는 것이 좋다

함수형 컴포넌트에서 ref를 사용할 때에는 useRef라는 Hook 함수를 사용한다.

기존에 만들었던 InputSample의 초기화 버튼을 클릭햇을 때 이름 input에 포커스가 잡히도록 useRef를 사용하여 기능을 구현해보자.





useRef()를 사용하여 Ref객체를 만들고, 이 객체를 우리가 선택하고 싶은 DOM에 ref 값으로 설정해주어야 한다. 그러면, Ref 객체의 .current값은 우리가 원하는 DOM을 가리키게 된다.

위 코드는 onReset 함수에서 input에 포커스를 하는 focus() DOM API를 호출한다.

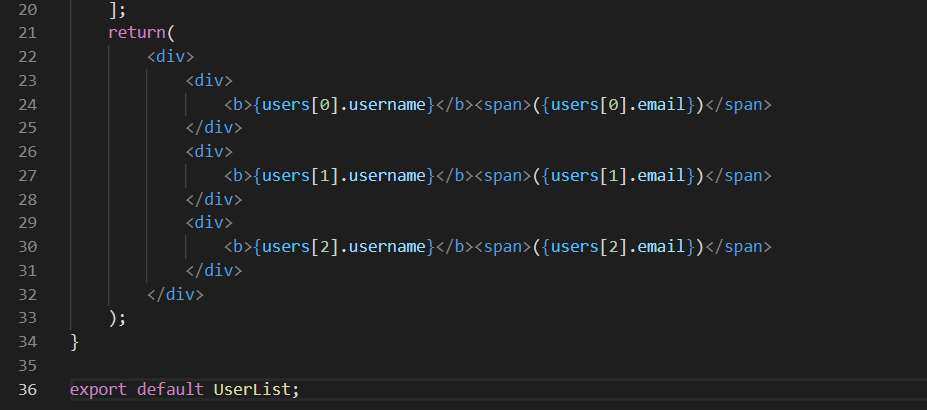
기존 코드와 다르게 초기화 버튼을 눌렀을 때 이름 input에 커서가 자동으로 focus되게 된다.

배열 렌더링하기

**UserList.js**

1. **가장 비효율적인 가장 기본적인, 그대로 코드 작성**

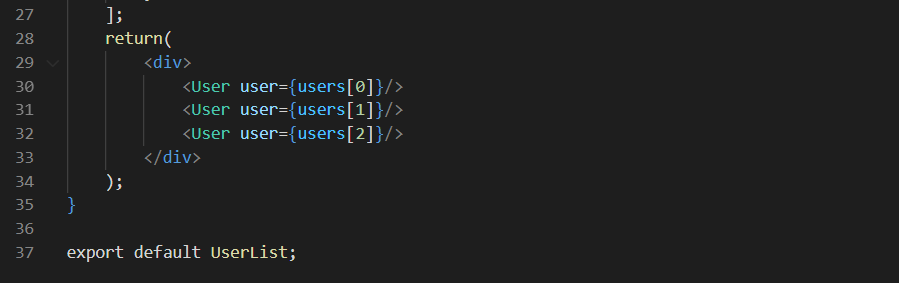




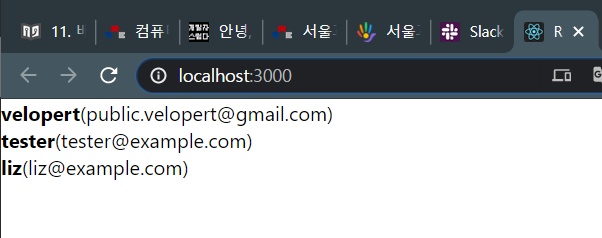
재사용될 수 있는 코드를 일일히 넣는건 별로 좋지 않음! 따라서 2번과 같이 재사용 할 수 있도록 만들어 주자. ( 한 파일에 여러 개의 컴포넌트를 선언해도 괜찮음)

1. **재사용 할 수 있는 코드**





이와 같이 작성하면

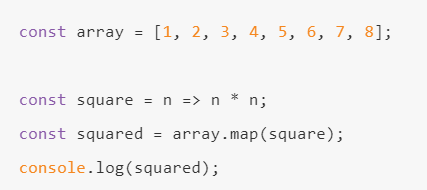


다음과 같은 결과를 얻을 수 있다.

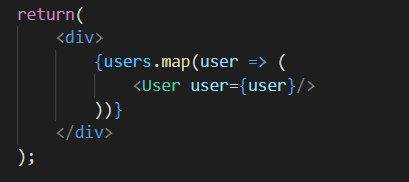
1. **map() 함수 이용하기.**

배열의 인덱스를 하나하나 조회해가면서 렌더링 하는 방법은 동적인 배열을 렌더링 하지 못한다. 동적인 배열을 렌더링 할 때에는 map() 함수를 사용해야 한다.

map()은 배열 안의 각 원소를 변환 할 때 사용 되며, 이 과정에서 새로운 배열을 만든다. 예를 들어 const array=[1,2,3,4,5,6,7,8]; 이라는 배열이 있다고 가정하고, 배열 안의 모든 숫자를 제곱해서 새로운 배열을 만들고 싶다면



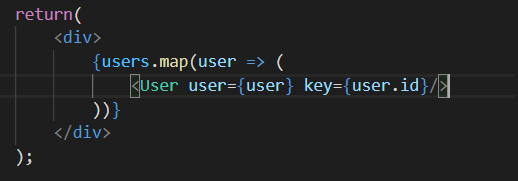
다음과 같이 간단하게 작성할 수 있다. 이제 앞의 코드를 수정해보자.



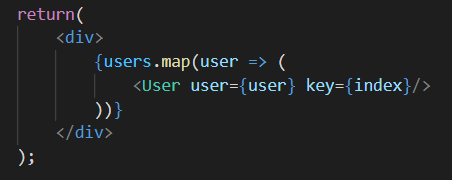
기존 코드에서 return() 부분만 map 함수를 이용해서 바꿔보았다. 반복되는 형태를 간단하게 표현할 수 있지만, 브라우저 콘솔을 열어보면 에러가 뜸을 확인 할 수 있다. 리액트에서 배열을 렌더링할 때에는 key라는 props를 설정해야 한다. key 값은 각 원소들마다 가지고 있는 고유값으로 설정을 해야 한다.

1. **key 값을 이용하기**

return() 내부에 다음과 같이 key 값을 추가해준다. 여기서는 id가 고유값이 될 수 있다.



만약 배열 안의 원소의 고유값이 없다면 index를 key 값으로 사용할 수 있다.



배열을 렌더링 할 때 key 값을 설정하지 않는다면 index가 기본적으로 사용되지만 경고 메시지는 여전히 뜬다. 경고 메시지가 뜨는 이유는 각 고유 원소에 key가 있어야먄 배열 이 업데이트 될 때 효율적으로 렌더링 될 수 있기 때문이다.

**key의 존재유무에 따른 업데이트 방식**

다음과 같은 배열이 있다고 가정하자.

const array = [‘a’,’b’,’c’,’d’];

위 배열을 다음과 같이 렌더링한다고 가정하자.

array.map(item => <div>{item}</div>);

key를 사용하지 않는 경우에는 -> 위 배열의 b와 c 사이에 z를 삽입하게 된다면, 리렌더링을 하게 될 대 <div>b</div>와 <div>c</div> 사이에 새 div 태그를 삽입하게 되는게 아니라, 기존의 c가 z로 바뀌고, d는 c로 바뀌고 맨 마지막에 d가 새로 삽입된다.

만약 a를 제거하게 된다면 기존의 a가 b로 바뀌고, b는 z로 바뀌고, 이와 같은 상황을 반복하여 c가 d로 바뀌고 맨 마지막에 있는 d가 제거된다.

이런 상황을 key를 이용하면 ->

다음과 같이 렌더링이 된다면

array.map(item=> <div key={item.id}>{item.text}</div>);

배열이 업데이트 될 때 key가 없을 때처럼 비효율적으로 업데이트하는 것이 아니라, 수정되지 않는 기존의 값은 그대로 두고 원하는 곳에 내용을 삽입하거나 삭제합니다. 즉 교체하는 과정 없이 바로 삽입/삭제가 가능하다.

useRef로 컴포넌트 안의 변수 만들기

컴포넌트에서 특정 DOM을 선택해야 할 때, ref를 사용해야 한다고 배웠다. useRef Hook은 DOM을 선택하는 용도 외에도, 컴포넌트 안에서 조회 및 수정 할 수 있는 변수를 관리하는 용도로 사용된다. useRef로 관리하는 변수는 값이 바뀐다고 해서 컴포넌트가 리렌더링 되지 않는다. 리액트 컴포넌트에서의 상태는 상태를 바꾸는 함수를 호출하고 나서 그 다음 렌더링 이후로 업데이트 된 상태를 조회할수 있는 반면, useRef로 관리하고 있는 변수는 설정 후 바로 조회 할 수 있다.

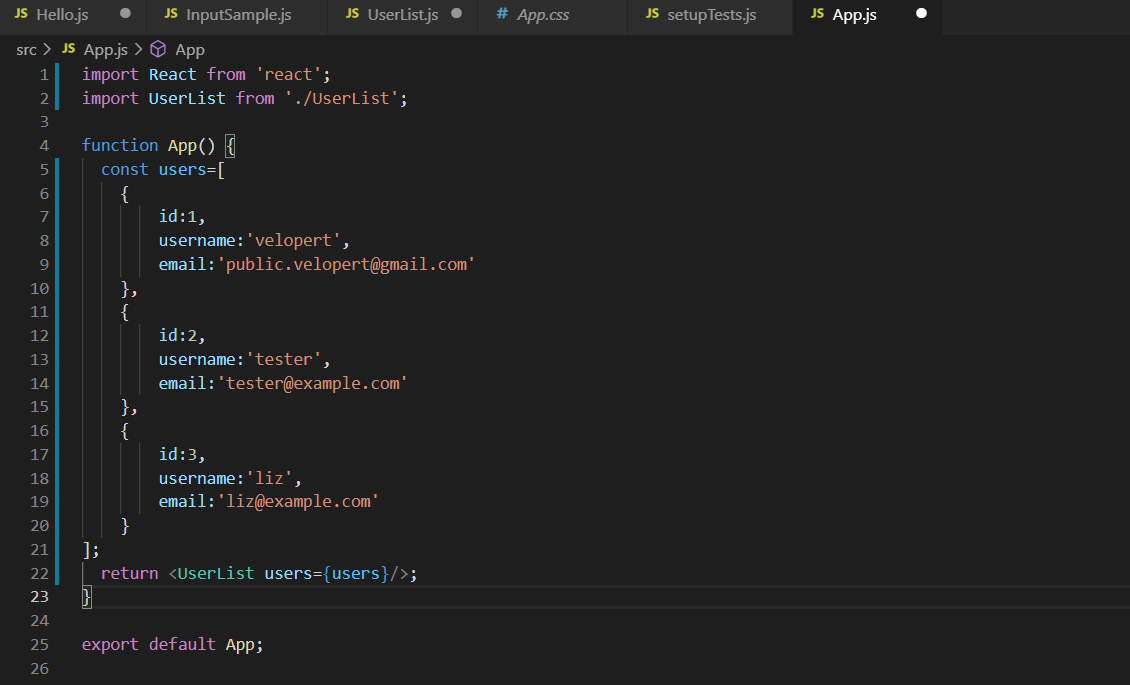
예를 들어,

* setTimeout, setInterval 을 통해서 만들어진 id
* 외부 라이브러리를 사용하여 생성된 인스턴스
* scroll 위치

등을 관리 할 수 있다.

우리는 배열에 새 항목을 추가하는데, 새 항목에서 사용할 고유 id를 관리하는 용도로 useRef를 사용하자.

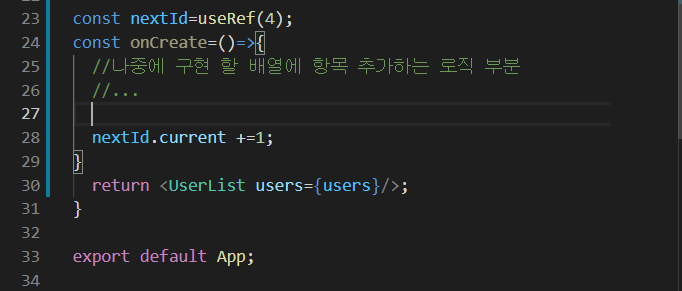
다음과 같이 배열을 App에서 선언하고 UserList에게 props로 전달해보자.



다음과 같이 UserList.js를 작성하자



여기서 App.js에서 **useRef()**를 사용하여 nextId 라는 변수를 만들어보자.



useRef()를 사용할 때 파라미터를 넣어주면, 이 값이 .current 값의 기본값이 된다.

이 값을 수정할 때에는 .current값을 수정하면 되고 조회할 때에도 마찬가지로 .current를 조회하면 된다.

+ useRef 함수는 current 속성을 가지고 있는 객체를 반환하는데, 인자로 넘어온 초기값을 current 속성에 할당한다. 이 current 속성은 값을 변경해도 상태를 변경할 때처럼 React컴포넌트가 다시 랜더링 되지 않는다. 또한 React 컴포넌트가 다시 랜더링 될 때도 마찬기지로 이 current 속성의 값이 유실되지 않는다. +