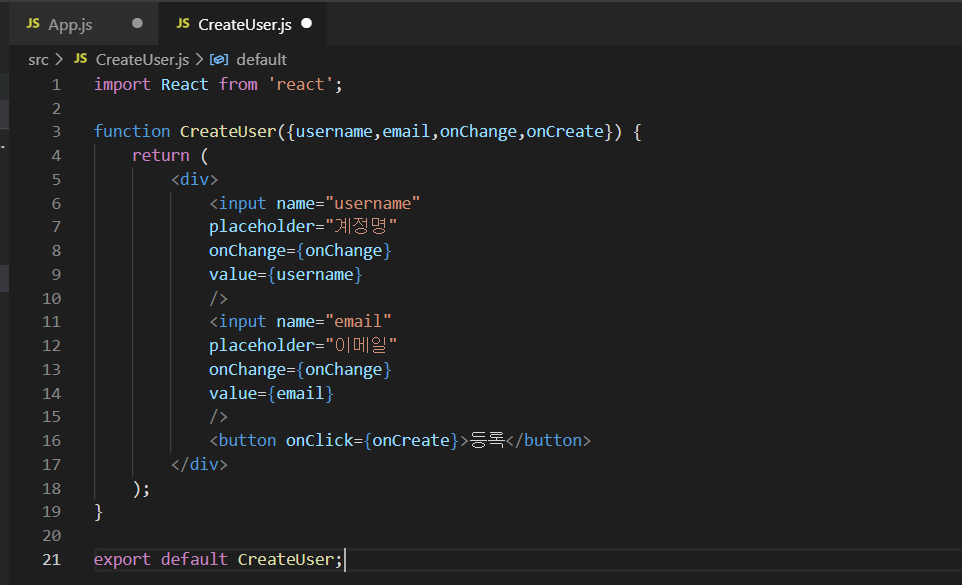
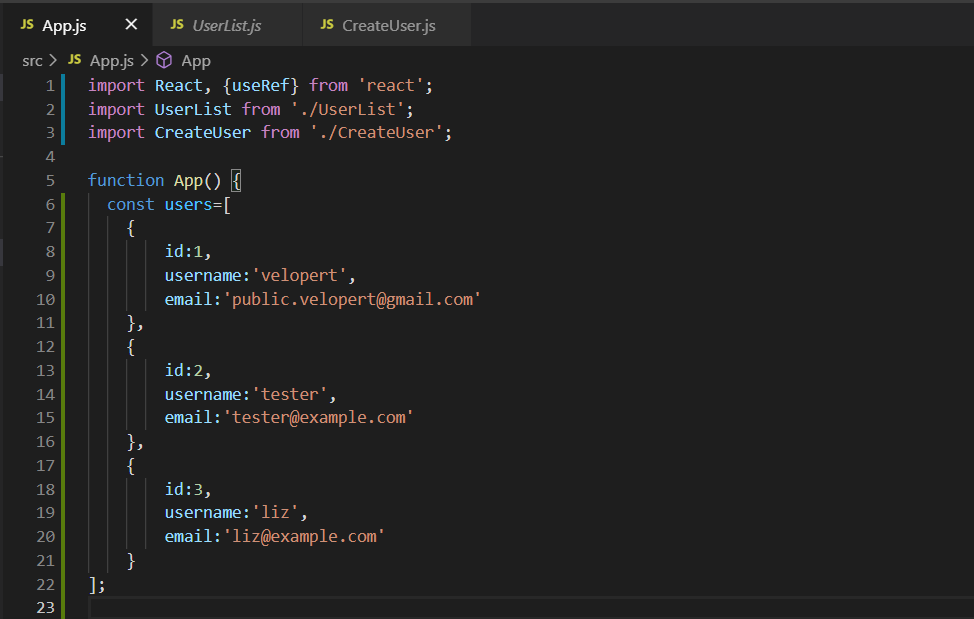
배열에 항목 추가하기

input 두 개와 button 하나로 이루어진 CreateUser.js라는 컴포넌트를 src 디렉토리에 만들어보자

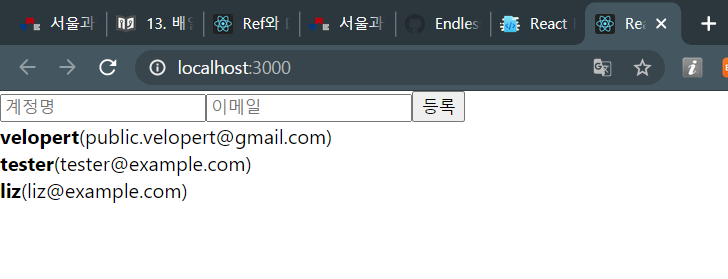


이번 컴포넌트에서는 상태 관리를 App에서 하고, input의 값 및 이벤트로 등록할 함수들을 props로 넘겨받아서 사용해보자. 이 컴포넌트를 App에서 UserList위에 렌더링해보자,





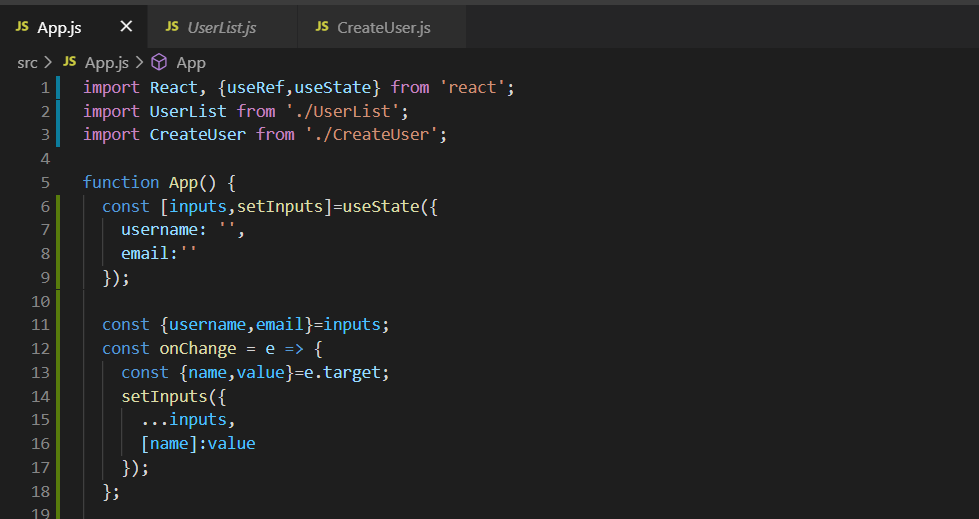
다음과 같이 작성하면 App.js를 작성하면 아래와 같은 결과를 얻을 수 있다. 아직 등록 버튼은 작동되지 않는 상태이다.



배열의 불변성을 지키면서 배열에 새 항목을 추가하는 방법은 두가지가 있다.

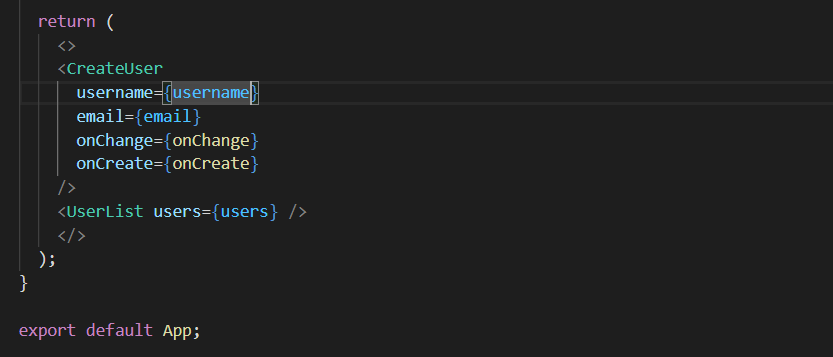
1. **spread 연산자(…)를 사용하는 것**

spread라는 단어가 가지고 있는 의미는 펼치다. 퍼뜨리다 이다. 이 문법을 사용하면, 객체 혹은 배열을 펼칠 수 있다.

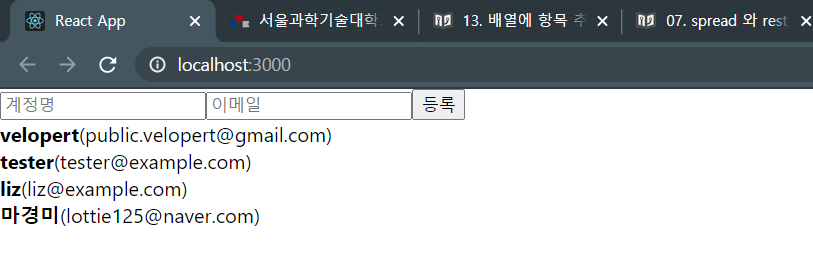




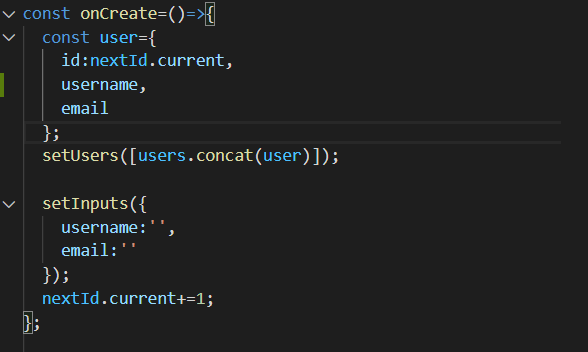




다음과 같이 작성하면 등록 버튼을 눌렀을 때 배열에 새로운 항목이 삽입되었음을 알 수 있다.



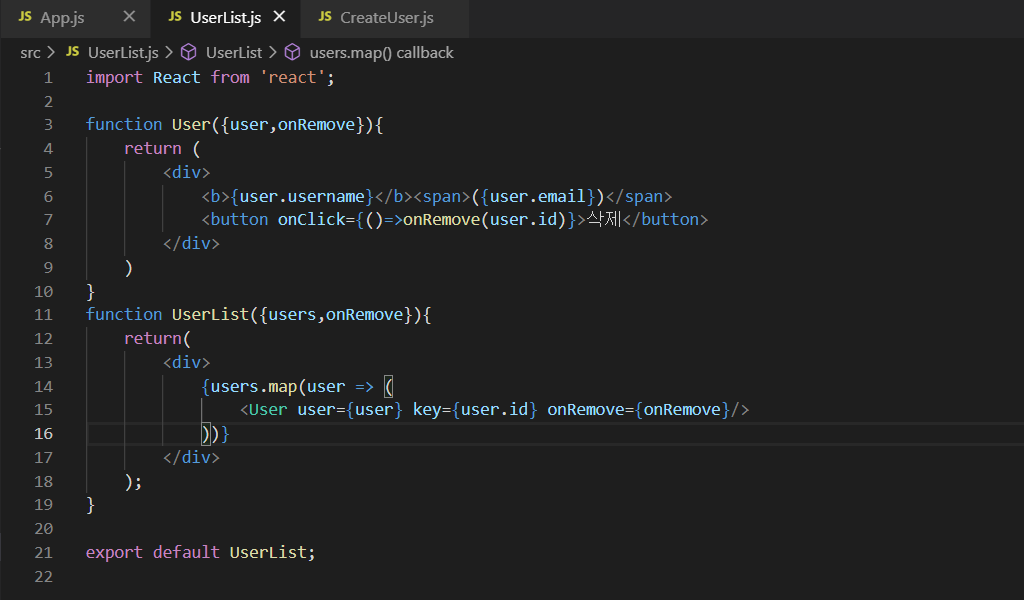
1. **concat 함수를 사용하여** 기존의 배열을 수정하지 않고, 새로운 원소가 추가된 새로운 배열을 만들 수 있다.



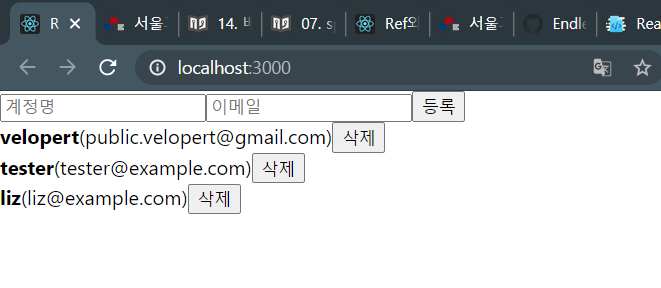
setUsers()부분을 다음과 같이 concat을 사용하여 정의한다

배열에 항목 제거하기

이번에는 배열에 항목을 제거하는 방법을 알아보자 UserList를 다음과 같이 수정한다



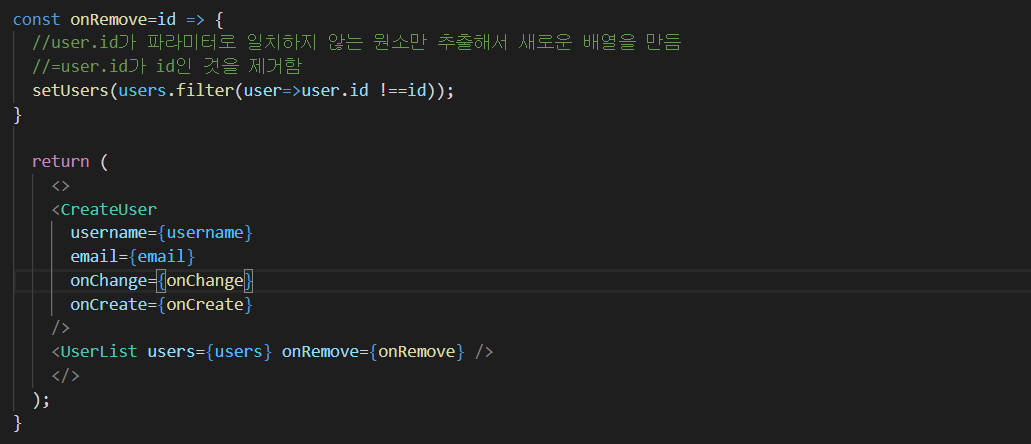
아래와 같이 삭제 버튼을 만들고 아직 삭제는 행동되지 않는다.



User컴포넌트의 삭제 버튼이 클릭 될 때에 user.id 값을 앞으로 props로 받아올 onRemove 함수의 파라미터로 넣어서 호출해줘야 한다. 이 때 onRemove의 역할은 id가 \_인 객체를 삭제하라 이다. 이 onRemove 함수는 UserList에서도 전달 받을 것이며, 이를 그대로 User 컴포넌트에게 전달해 줄 것이다.

이제 onRemove 함수를 불변성을 지키면서 구현해보자. **불변성을 지키면서 특정 원소를 배열에서 제거하기 위해선는 filter 배열 내장 함수를 사용하는 것이 가장 편하다.** 이 함수는 배열에서 특정 조건이 만족하는 원소들만 추출하여 새로운 배열을 만들어준다.

다음과 같이 App 컴포넌트에 onRemove 함수를 구현하자.



이와 같이 구현하면 삭제 버튼이 잘 작동됨을 알 수 있다.

배열 항목 수정하기

User 컴포넌트에 계정명을 클릭했을 때 색상이 초록색으로 바뀌고, 다시 누르면 검정색으로 바뀌도록 구현해보자.

App컴포넌트의 users 배열 안의 객체 안에 **active라는 속성을 추가한다.**

active 값에 따라 글자의 색상을 바꿀 수 있게 구현하고, 추가적으로 cursor 필드를 설정하여 마우스를 올렸을 때 커서가 손가락 모양으로 변하게 구현해 보자

이 때 App.js에서 onToggle 함수를 구현한다. 배열의 불변성을 유지하면서 배열을 업데이트 할 때에도 map 함수를 사용할 수 있다. id 값을 비교해서 id가 다르다면 그대로 두고, 같다면 active 값을 반전시키도록 구현을 한다. onToggle 함수를 만들어서 UserList 컴포넌트에게 전달해주자

UserList컴포넌트에서 onToggle을 받아와서 User에게 전달해주고, onRemove를 구현했던 것처럼 onToggle에 id를 넣어서 호출한다.

App.js

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

import UserList from './UserList';

import CreateUser from './CreateUser';

function App() {

  const [inputs,setInputs]=useState({

    username: '',

    email:''

  });

  const {username,email}=inputs;

  const onChange = e => {

    const {name,value}=e.target;

    setInputs({

      ...inputs,

      [name]:value

    });

  };

  const [users,setUsers]=useState([

    {

        id:1,

        username:'velopert',

        email:'public.velopert@gmail.com',

        active:true

    },

    {

        id:2,

        username:'tester',

        email:'tester@example.com',

        active:false

    },

    {

        id:3,

        username:'liz',

        email:'liz@example.com',

        active:false

    }

]);

const nextId=useRef(4);

const onCreate=()=>{

  const user={

    id:nextId.current,

    username,

    email

  };

  setUsers([...users,user]);

  setInputs({

    username:'',

    email:''

  });

  nextId.current+=1;

};

const onRemove = id => {

  //user.id가 파라미터로 일치하지 않는 원소만 추출해서 새로운 배열을 만듬

  //=user.id가 id인 것을 제거함

  setUsers(users.filter(user=>user.id !==id));

}

const onToggle=id=>{

  setUsers(

    users.map(user=>

      user.id===id ? {...user,active: !user.active} :user

      )

  );

};

  return (

    <>

    <CreateUser

      username={username}

      email={email}

      onChange={onChange}

      onCreate={onCreate}/>

    <UserList users={users} onRemove={onRemove} onToggle={onToggle} />

    </>

  );

}

export default App;

UserList.js

import React from 'react';

function User({user,onRemove,onToggle}){

    return (

        <div>

            <b style={{

                cursor:'pointer',

                color:user.active ?'green':'black'

            }}

            onClick={()=>onToggle(user.id)}

            >

                {user.username}

                </b>&nbsp;

                <span>({user.email})</span>

            <button onClick={()=>onRemove(user.id)}>삭제</button>

        </div>

    );

}

function UserList({users,onRemove,onToggle}){

    return(

        <div>

            {users.map(user => (

                <User user={user}

                key={user.id}

                onRemove={onRemove}

                onToggle={onToggle}/>

            ))}

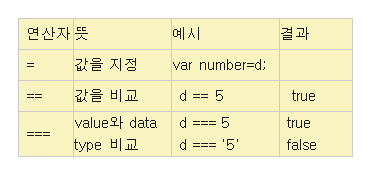
        </div>

    );

}

export default UserList;

다음과 같이 작성하면 아래와 같이 계정명을 누르면 초록색을 변하고 다시 눌렀을 때 검정색으로 변함을 볼 수 있다. 또한 캡쳐에는 나와있지 않지만 cursor 모양도 변한다.

이때 ===연산자는  value와 data type까지 비교하는 연산자이다.

