**10. useRef로 컴포넌트 안의 변수 만들기**

* **지난 시간 정리**
* 컴포넌트에서 특정 DOM을 선택할 때, ref를 사용해야한다.
* 함수형 컴포넌트에서 이를 설정할 때 useRef를 사용하여 설정한다.
* **useRef의 새로운 용도**
* useRef Hook은 DOM을 선택하는 용도 외에도, 컴포넌트 안에서 조회 및 수정 할 수 있는 변수를 관리할 수 있는 용도가 있다.
* useRef로 관리하는 변수는 값이 바뀐다고 컴포넌트가 리렌더링 되지 않음.
* 리액트 컴포넌트에서의 상태는 상태를 바꾸는 함수를 호출하고 나서 그 다음 렌더링 이후로 업데이트 된 상태를 조회 할 수 있는 반면
* useRef로 관리하고 있는 변수는 설정 후 바로 조회할 수 있다.
* 이 변수로 다음과 같은 값 관리 가능

1. setTimeout, setInterval 을 통해 만들어진 id
2. 외부 라이브러리를 사용하여 생성된 인스턴스
3. scroll의 위치

<App 컴포넌트에서 useRef를 사용하여 변수를 관리해 보기, 용도 : 우리가 앞으로 배열에 새 항목을 추가할 때 사용할 고유 id를 관리하는 용도>

**해야할 일**

1. **배열을 App에서 선언하고 UserList에게 props로 전달**

* **Apps.js**

import React from 'react';

import UserList from './UserList';

function App() {

  const users = [

    {

      id: 1,

      username: 'velopert',

      email: 'public.velopert@gmail.com'

    },

    {

      id: 2,

      username: 'tester',

      email: 'tester@example.com'

    },

    {

      id: 3,

      username: 'liz',

      email: 'liz@example.com'

    }

  ];

  return <UserList users = {users} />

}

export default App;

* **UserList.js**

import React from 'react';

function User({user}){

    return(

        <div>

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

        </div>

    );

}

function UserList() {

      return(

          <div>

              {users.map(user => (

                  <User user = {user} key = {user.id} />

              ))}

          </div>

      )

}

export default UserList;

1. **App에서 useRef를 사용하여 nextId라는 변수 만들기**

**-App.js**

const nextId = useRef(4);

  const onCreate = () => {

    //나중에 구현 할 배열에 항목을 추가하는 로직

    //...

    nextId.current += 1;

}

* 이거 추가하기
* useRef()를 사용할 때 파라미터를 넣어주면, 이 값이 .current 값의 기본값이 된다.
* 이 값을 수정할 때에는 .current 값을 수정하면 되고 조회할 때는 .current 를 조회하면 된다.