13. 배열에 항목 추가하기

이 섹션에서 사용된 코드는 다음 페이지에서 확인 할 수 있습니다.

이번에는 배열에 새로운 항목을 추가하는 방법을 알아보겠습니다.

우선, input 두개와 button 하나로 이루어진 CreateUser.js 라는 컴포넌트를 src 디렉터리에 만들어봅시다.

CreateUser.js

```
import React from 'react';

function CreateUser({ username, email, onChange, onCreate }) {
  return (
    <div>
      <input
        name="username"
        placeholder="계정명"
        onChange={onChange}
        value={username}
      />
      <input
        name="email"
        placeholder="이메일"
        onChange={onChange}
        value={email}
      />
      <button onClick={onCreate}>등록</button>
    </div>
  );
}

export default CreateUser;
```

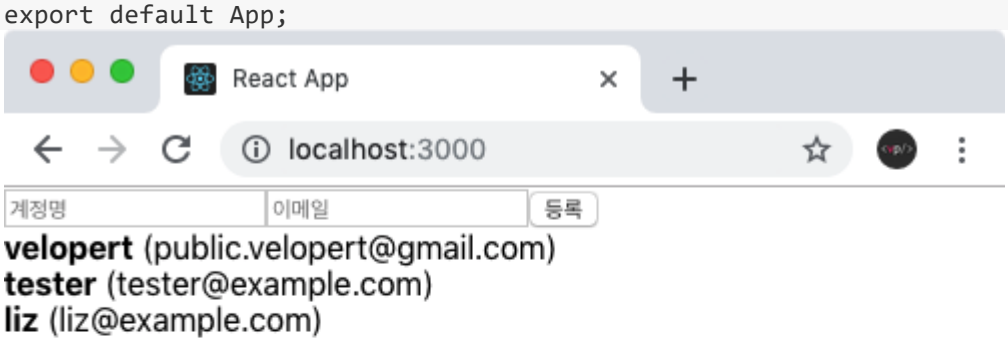
이번 컴포넌트에서는 상태관리를 CreateUser 에서 하지 않고 부모 컴포넌트인 App 에서 하게 하고, input 의 값 및 이벤트로 등록할 함수들을 props 로 넘겨받아서 사용해주겠습니다.

이 컴포넌트를 App 에서 UserList 위에 렌더링해보세요.

App.js

```
import React, { useRef } from 'react';
import UserList from './UserList';
import CreateUser from './CreateUser';

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'
    }
  ];

  const nextId = useRef(4);
  const onCreate = () => {
    // 나중에 구현 할 배열에 항목 추가하는 로직
    // ...

    nextId.current += 1;
  };
  return (
    <>
      <CreateUser />
      <UserList users={users} />
    </>
  );
}

export default App;
```



CreateUser 컴포넌트에게 필요한 props 를 App 에서 준비해주겠습니다.

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 = [
    {
      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'
    }
  ];

  const nextId = useRef(4);
  const onCreate = () => {
    // 나중에 구현 할 배열에 항목 추가하는 로직
    // ...

    setInputs({
      username: '',
      email: ''
    });
    nextId.current += 1;
  };
  return (
    <>
      <CreateUser
        username={username}
        email={email}
        onChange={onChange}
        onCreate={onCreate}
      />
      <UserList users={users} />
    </>
```

```
  );
}

export default App;
```

코드를 작성 후, input 에 값을 입력하고, 등록 버튼을 눌렀을 때 input 값들이 잘 초기화되는지 확인해보세요.

그 다음에는, users 도 `useState` 를 사용하여 컴포넌트의 상태로서 관리해주세요.

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'
    },
    {
      id: 2,
      username: 'tester',
      email: 'tester@example.com'
    },
    {
      id: 3,
      username: 'liz',
      email: 'liz@example.com'
    }
  ]);

  const nextId = useRef(4);
  const onCreate = () => {
    // 나중에 구현 할 배열에 항목 추가하는 로직
    // ...

    setInputs({
      username: '',
      email: ''
    });
    nextId.current += 1;
```

```
  };
  return (
    <>
      <CreateUser
        username={username}
        email={email}
        onChange={onChange}
        onCreate={onCreate}
      />
      <UserList users={users} />
    </>
  );
}

export default App;
```

이제 배열에 변화를 줄 차례입니다. 배열에 변화를 줄 때에는 객체와 마찬가지로, 불변성을 지켜주어야 합니다. 그렇기 때문에, 배열의 `push`, `splice`, `sort` 등의 함수를 사용하면 안됩니다. 만약에 사용해야 한다면, 기존의 배열을 한번 복사하고 나서 사용해야합니다.
불변성을 지키면서 배열에 새 항목을 추가하는 방법은 두가지가 있습니다.

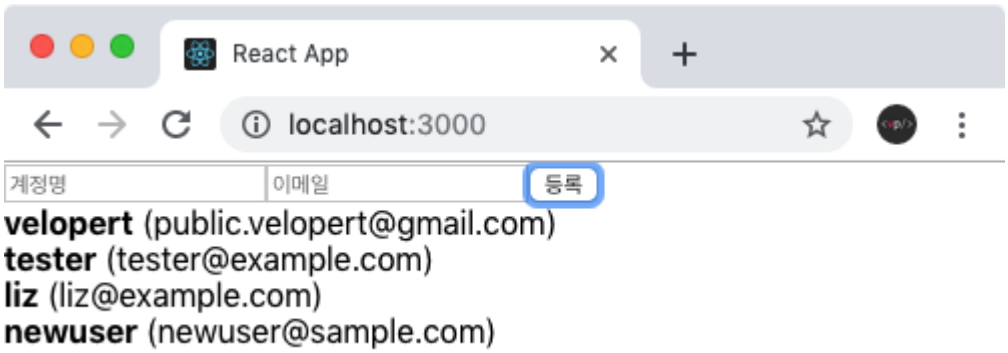
첫번째는 spread 연산자를 사용하는 것 입니다.

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'
    },
    {
      id: 2,
      username: 'tester',
      email: 'tester@example.com'
    },
    {
      id: 3,
```

```
      username: 'liz',
      email: 'liz@example.com'
    }
  ]);

  const nextId = useRef(4);
  const onCreate = () => {
    const user = {
      id: nextId.current,
      username,
      email
    };
    setUsers([...users, user]);

    setInputs({
      username: '',
      email: ''
    });
    nextId.current += 1;
  };
  return (
    <>
      <CreateUser
        username={username}
        email={email}
        onChange={onChange}
        onCreate={onCreate}
      />
      <UserList users={users} />
    </>
  );
}

export default App;
```

자, 새 항목이 잘 등록되는지 확인해보세요.



또 다른 방법은 `concat` 함수를 사용하는 것 입니다. `concat` 함수는 기존의 배열을 수정하지 않고, 새로운 원소가 추가된 새로운 배열을 만들어줍니다.

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'
    },
    {
      id: 2,
      username: 'tester',
      email: 'tester@example.com'
```

```
    },
    {
      id: 3,
      username: 'liz',
      email: 'liz@example.com'
    }
  ]);

  const nextId = useRef(4);
  const onCreate = () => {
    const user = {
      id: nextId.current,
      username,
      email
    };
    setUsers(users.concat(user));

    setInputs({
      username: '',
      email: ''
    });
    nextId.current += 1;
  };
  return (
    <>
      <CreateUser
        username={username}
        email={email}
        onChange={onChange}
        onCreate={onCreate}
      />
      <UserList users={users} />
    </>
  );
}

export default App;
```

배열에 새 항목을 추가 할 때에는 이렇게 spread 연산자를 사용하거나, `concat` 을 사용하시면 됩니다.