# 15. 배열 항목 수정하기

이번에는, 배열 항목을 수정하는 방법을 알아보겠습니다.

User 컴포넌트에 계정명을 클릭했을때 색상이 초록새으로 바뀌고, 다시 누르면 검정색으로 바뀌도록 구현을 해봅시다.

우선, App 컴포넌트의 users 배열 안의 객체 안에 `active` 라는 속성을 추가해주세요.

## 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.concat(user));

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

  const onRemove = id => {
    // user.id 가 파라미터로 일치하지 않는 원소만 추출해서 새로운 배열을 만듬
    // = user.id 가 id 인 것을 제거함
    setUsers(users.filter(user => user.id !== id));
  };

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

export default App;
```

그 다음에는 User 컴포넌트에서 방금 넣어준 `active` 값에 따라 폰트의 색상을 바꿔주도록 구현을 해보세요. 추가적으로, `cursor` 필드를 설정하여 마우스를 올렸을때 커서가 손가락 모양으로 변하도록 하겠습니다.

## UserList.js

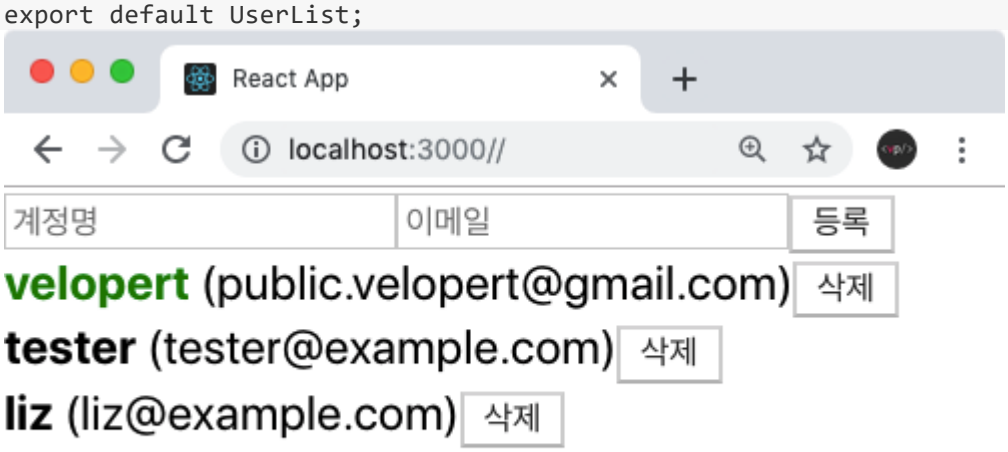
```
import React from 'react';

function User({ user, onRemove }) {
  return (
    <div>
      <b
        style={{
          cursor: 'pointer',
          color: user.active ? 'green' : 'black'
        }}
      >
        {user.username}
      </b>

      <span>({user.email})</span>
      <button onClick={() => onRemove(user.id)}>삭제</button>
    </div>
  );
}
```

```
function UserList({ users, onRemove }) {
  return (
    <div>
      {users.map(user => (
        <User user={user} key={user.id} onRemove={onRemove} />
      ))}
    </div>
  );
}

export default UserList;
```




이제 App.js 에서 `onToggle` 이라는 함수를 구현해보겠습니다. 배열의 불변성을 유지하면서 배열을 업데이트 할 때에도 `map` 함수를 사용 할 수 있습니다.
`id` 값을 비교해서 `id` 가 다르다면 그대로 두고, 같다면 `active` 값을 반전시키도록 구현을 하시면 됩니다.
`onToggle` 함수를 만들어서 UserList 컴포넌트에게 전달해주세요.

## App.js

```jsx
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.concat(user));

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

  const onRemove = id => {
    // user.id 가 파라미터로 일치하지 않는 원소만 추출해서 새로운 배열을 만듬
    // = user.id 가 id 인 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;
```

그 다음에는 UserList 컴포넌트에서 `onToggle`를 받아와서 User 에게 전달해주고, `onRemove` 를 구현했었던것처럼 `onToggle` 에 `id` 를 넣어서 호출해주세요.

## 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>
       
      <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;
```