**1장. 리액트 입문 2/2**

**<<11. 배열 렌더링하기>>**

**\* 배열과 컴포넌트 재사용을 이용하여 렌더링**

- UserList.js

import React from 'react';

function User({ user }) { //컴포넌트 재사용

return (

<div>

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

</div>

);

}

function UserList() {

const users = [ //배열 선언

{

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 (

<div>

<User user={users[0]} /> //배열 사용

<User user={users[1]} />

<User user={users[2]} />

</div>

);

}

export default UserList;

텍스트이(가) 표시된 사진

자동 생성된 설명

- 배열이 고정적이라면 상관 없겠지만, 배열의 인덱스를 하나하나 조회해가면서 렌더링 하는 방법은 동적인 배열을 렌더링하지 못함

- 동적인 배열을 렌더링해야 할 때는 자바스크립트 배열의 내장함수 map()을 사용함

- **map()함수**는 배열 안에 있는 각 원소를 변환하여 새로운 배열을 만들어줌. 리액트에서 동적인 배열을 렌더링해야 할 때는 이 함수를 사용하여 일반 데이터 배열을 리액트 앨리먼트로 이루어진 배열로 변환해주면 됨

- (필수) 리액트에서 배열을 렌더링 할 때는 key라는 props를 설정해야 함. key값은 각 원소들마다 가지고 있는 고유값으로 설정해야 함. 지금의 경우는 id가 고유값임

- UserList.js

import React from 'react';

function User({ user }) {

return (

<div>

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

</div>

);

}

function UserList() {

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 (

<div>

{users.map(user => (

<User user={user} key={user.id} /> //map() 사용, key 설정

))}

</div>

);

}

export default UserList;

**\* key의 존재 유무에 따른 업데이트 방식**

const array = ['a', 'b', 'c', 'd'];

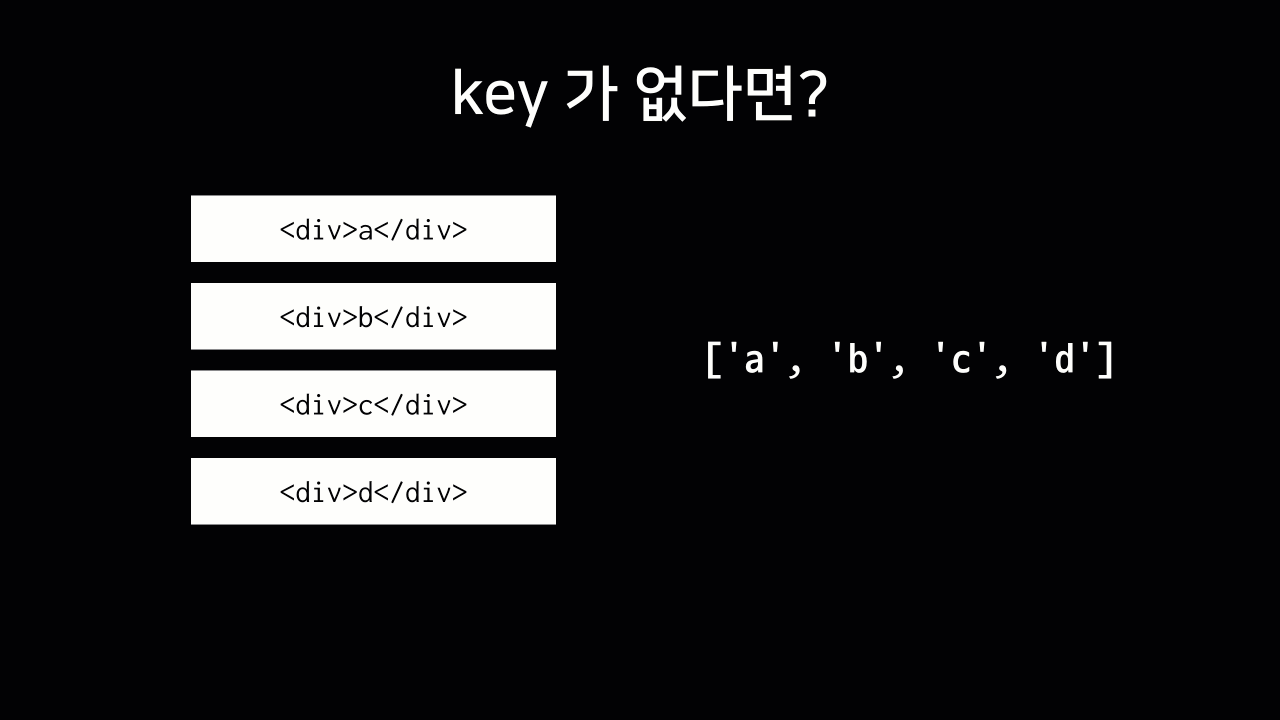
- 해당 배열이 존재한다고 가정

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

- 다음과 같이 렌더링

- 위 배열의 b와 c 사이에 z를 삽입하게 된다면, 리렌더링을 할 때 <div>b</div>와 <div>c</div> 사이에 새 div 태그를 삽입을 하게 되는 것이 아니라 기존의 c가 z로 바뀌고 d는 c로 바뀌고 맨 마지막에는 d가 새로 삽입됨.

- 그 다음에 a 를 제거하게 된다면, 기존의 a 가 b 로 바뀌고, b 는 z 로 바뀌고, z는 c로 바뀌고, c는 d 로바뀌고, 맨 마지막에 있는 d 가 제거됨



- 비효울적임. 하지만 key가 있다면 개선할 수 있음

[

{

id: 0,

text: 'a'

},

{

id: 1,

text: 'b'

},

{

id: 2,

text: 'c'

},

{

id: 3,

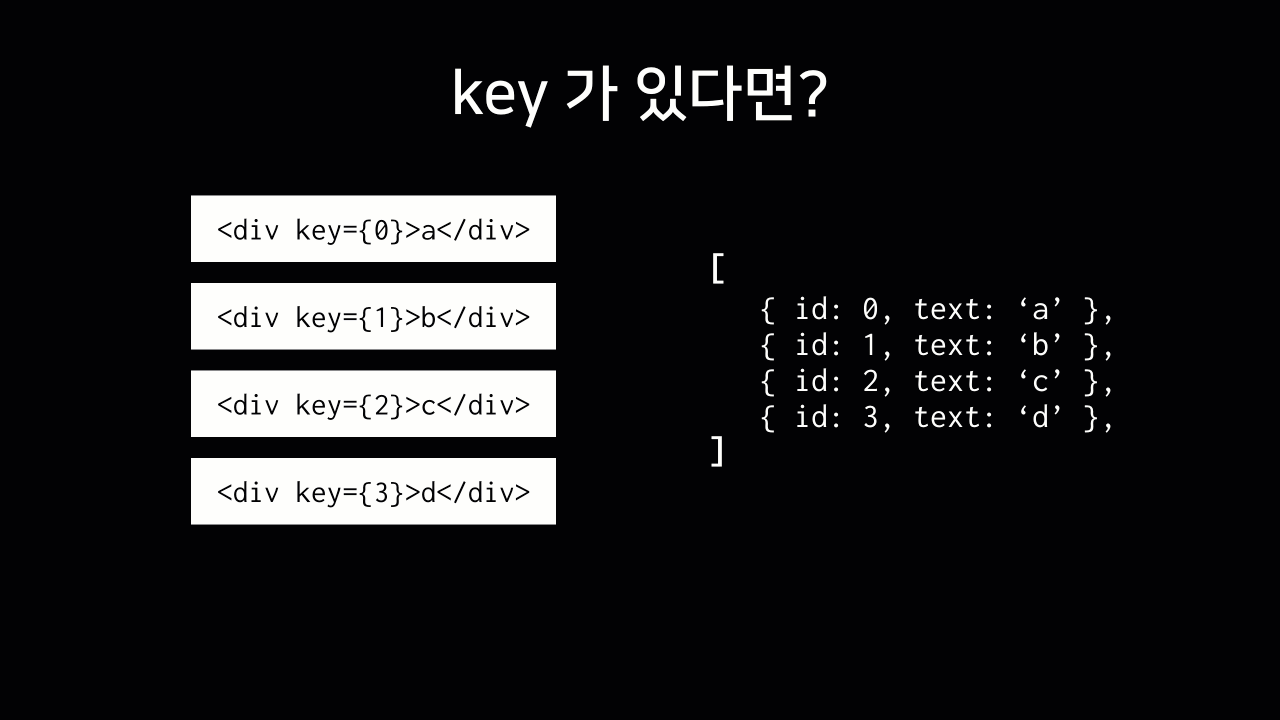
text: 'd'

}

];

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

- 배열이 업데이트 도리 때 key가 없을 때처럼 비효율적으로 업데이트 하는 것이 아니라, 수정되지 않는 기존의 값은 그대로 두고 원하는 곳에 내용을 삽입하거나 삭제함



- 때문에, 배열을 렌더링 할 때에는 고유한 key 값이 있는것이 중요하며, 만약에 배열안에 중복되는 key 가 있을 때에는 렌더링시에 오류메시지가 콘솔에 나타나게 되며, 업데이트가 제대로 이루어지지 않게 됨