**배운 내용 요약**

    let [글제목, 글제목변경] = useState(['남자 코트 추천', ' 강남 우동 맛집', '파이썬 독학']);

    let [좋아요, 좋아요변경] = useState([0,0,0]);

    let [좋아요1, 좋아요변경1] = useState(0);

    let [modal, modal변경] = useState(false); // 이거는 모달창을 켜고 닫는 스위치이다.

    // .map : array 내의 모든 데이터에 똑같은 작업을 시켜주고 싶을 때

    // var 어레이 = [2,3,4];

    // var 뉴어레이 = 어레이.map(function(a){

    //     return a\*2;

    //     // 이렇게 하면은 [4,6,8]이 됨

    // });

    function 좋아요함수(idx){

        var newArray = [...좋아요];

        newArray[idx] = 좋아요[idx]+1;

        좋아요변경(newArray)

    }

    return (

        <div className="App">

            <div className="black-nav">

                <div>개발 Blog

                </div>

            </div>

            <div className='list'>

                <h3> {글제목[0]} <span onClick={()=>{ 좋아요함수(0) }}> 👍 </span> {좋아요[0]} </h3>

                    <p>2월 17일 발행</p>

                    <hr/>

            </div>

            <div className = 'list'>

                <h3> {글제목[1]} <span onClick={ ()=>{ 좋아요함수(1)}}> 👍 </span> {좋아요[1]}   </h3>

                    <p>2월 17일 발행 </p>

                    <hr/>

            </div>

            <div className = 'list' >

                <h3> {글제목[2]}<span onClick={ ()=>{ 좋아요함수(2)}}> 👍 </span> {좋아요[2]}   </h3>

                <p>2 월 17일 발행</p>

                <hr/>

            </div>

            {/\* {// 반복문 쓰는 법 : for문말고 map

                [1,2,3].map(function(){

                    return <div>안녕</div>

                })

            } \*/}

            {/\* {

                // 반복될 때 마다 제목부분에 각각 다른 데이터가 들어가고 싶음

                글제목.map(function(글){ // 글제목만큼 본문을 반복시켜야함

                    return (

                    <div className = 'list' >

                    <h3> {글}<span onClick={()=>{좋아요변경(좋아요++)}}> 👍 </span> </h3>

                    <p>2 월 17 일 발행</p>

                    <hr/>

                    </div>

                    )

                })

            } \*/}

        </div>

    );

}

function Modal(){

    return(

        <div className="modal">

            <h2>제목</h2>

            <p>날씨</p>

            <p>상세내용 </p>

        </div>

    )

}

{

                // 반복될 때 마다 제목부분에 각각 다른 데이터가 들어가고 싶음

                글제목.map(function(글,i){ // 글제목만큼 본문을 반복시켜야함

                    return (

                    <div className = 'list' >

                    <h3> {글}<span onClick={()=>{ 좋아요함수(i)}}> 👍 </span> {좋아요[i]}   </h3>

                    <p>2 월 17 일 발행</p>

                    <hr/>

                    </div>

                    )

                })

            }

리액트환경에선 for 반복문을 이용해 HTML을 줄일 수도 있습니다.

그냥 반복문 적고 안에 HTML 담으시면 끝입니다.

하지만 JSX 코딩하던 중간중간에 for 반복문을 직접 쓸 수는 없고 특별한 방법이 좀 필요합니다.

함께 알아봅시다.

**자바스크립트 map 함수 쓰는 법**

JSX 중간중간 중괄호 열고 { 변수명 } 넣던거 기억나시죠?

그럼 반복문도 중괄호 안에서 { for (){} } 이렇게 집어넣을 수 있지않을까라고 생각할 수 있겠지만

{} 중괄호안에는 변수, 함수 이런 것만 입력가능합니다.

그래서 중괄호 안에서 쓸 수 있는 map 이라는 반복문을 이용합니다.

반복문은 아니고 array에 붙일 수 있는 일종의 내장함수인데 뭔지 자세히 알아봅시다.

array 자료형이 하나 있다고 칩시다.

array 안의 모든 자료에 똑같은 작업을 하나씩 시켜주고 싶을 때가 가끔 있습니다.

그럴 때 쓰는 문법이 바로 map 이라는 내장 함수입니다.

var 어레이 = [2,3,4];

어레이.map(function(){

});

위처럼 모든 array에 붙일 수 있으며 소괄호 안에 콜백함수 하나 넣는게 기본입니다.

그러면 map 안의 코드가 어레이 자료의 갯수만큼 실행됩니다.

3번 실행되겠네요.

어레이안의 자료에 전부 10을 곱해주고 싶으면 이렇게 작성합니다.

var 어레이 = [2,3,4];

어레이.map(function(a){

return a \* 10

});

콜백함수 소괄호 안에 파라미터를 아무이름이나 입력해주시면 (저는 a로 입력)

이 a라는 파라미터는 어레이 안에 있던 모든 자료를 하나씩 출력해주는 역할입니다.

그리고 a에 10을 곱하든 20을 곱하든 원하는 작업을 하나 시켜주시면 됩니다.

그럼 어레이에 있던 모든 자료가 10씩 곱해져셔 [20, 30, 40] 이 됩니다.

근데 참고로 map 함수는 원본 자료형을 변형시키지 않아서

보통 새로운 변수에 담아서 사용합니다.

var 어레이 = [2,3,4];

var newArray = 어레이.map(function(a){

return a \* 10

});

위처럼 작성하면 newArray에는 [20,30,40]이 담겨져있습니다.

원래 어레이에는 [2,3,4]가 담겨져있고요.

실은 기존 array 자료형을 변형시켜 새로운 array를 만들기 위해 나온 문법입니다.

암튼 여기까지가 map이라는 자바스크립트 문법의 사용법입니다. 끝

이제 이걸 리액트에서 반복문처럼 이용해보도록 합시다.

**JSX 안에서 map으로 반복문을 돌리고 싶으면**

<div>태그를 서너개 생성하고 싶은데, 좀 배운사람처럼 간지나게 반복문을 써보도록 합시다.

글제목이라는 state 갯수에 맞게 3개만 한번 생성해봅시다.

그냥 이렇게 써보시면 잘됩니다.

function App (){

return (

<div>

HTML 잔뜩있는 곳

...

{ 글제목.map(function(){

return (<div>안녕</div>)

}) }

</div>

)

}

▲ 이렇게 하시면 **<div>안녕</div>** 이라는게 **글제목**이라는 array내의 자료 갯수만큼 남습니다.

**글제목**이라는 array엔 3개 데이터가 들어있으니

실행해보시면 div도 3개가 남습니다.

그럼 우리가 예전에 만들었던 제목 UI를 반복문으로 생성해봅시다.

function App (){

return (

<div>

HTML 잔뜩있는 곳

...

{ 글제목.map(function(){

return (

<div className="list">

<h3>{ 글제목[1] }</h3>

<p>2월 18일 발행</p>

<hr />

</div>

}) }

</div>

)

}

▲ 이렇게 하시면 됩니다.

그냥 반복문 쓰실 땐 1. 원하는 자료에다가 map을 붙이시면 그 자료 갯수만큼 반복문을 돌릴 수 있으며

2. 반복을 원하는 HTML을 return 안에 적으면 긑입니다.

그러면 기존 코드보다 훨씬 적은 코드로 글목록 HTML을 생성가능합니다. 끝!

**근데 반복된 HTML에 각각 다른 제목을 부여하고 싶다면**

지금 위의 코드를 잘 보시면 **<h3>{ 글제목[1] }</h3>** 이라는 제목만 계속 3번 반복되고 있습니다.

그게 아니라 반복문이 돌 때마다 <h3></h3> 안에 각각

{ 글제목[0] }

{ 글제목[1] }

{ 글제목[2] }

이런 데이터가 들어가게 만들고 싶으면 어떻게 코드를 수정해야할까요?

그것은 아까 map 함수의 사용법을 잘 상기시켜본다면

콜백함수 안에 a라는 파라미터 하나 추가하면 그게 바로 어레이 안의 데이터들을 뜻한댔죠?

그걸 한번 써보시면 됩니다.

function App (){

return (

<div>

HTML 잔뜩있는 곳

...

{ 글제목.map(function(a){

return (

<div className="list">

<h3>{ a }</h3>

<p>2월 18일 발행</p>

<hr />

</div>

}) }

</div>

)

}

▲ 이렇게 하시면 이제 각각 다른 제목이 출력됩니다. 끝!

(참고로 a라는 파라미터는 맘에 안들면 아무렇게나 다른 이름으로 작명가능 ㅇㅇ)

a라는 파라미터는 map이 반복될 때마다 어레이 안에 있던 하나하나의 데이터들을 의미한다고 했습니다.

그니까 a라는건 map 안의 코드가 3번 반복될 때

차례로 **'남자 코트추천' '강남 우동 맛집'** 이런 데이터들이 된다는 소리입니다.

그걸 그냥 <h3> 안에 집어넣었을 뿐입니다.

[collapse]

반복된 HTML 안에 onClick={} 이런거 넣으셔도 잘 작동합니다.

function App (){

return (

<div>

HTML 잔뜩있는 곳

...

{ 글제목.map(function(a){

return (

<div className="list">

<h3 onClick={ ()=>{ 따봉변경(따봉+1) } }>{ a }</h3>

<p>2월 18일 발행</p>

<hr />

</div>

}) }

</div>

)

}

▲ 위의 예제코드는 그냥 대충 예시로 넣어봤는데,

아무튼 이렇게 해도 문제없이 잘 된다는 소리입니다.

하지만 이렇게 하드코딩하면 3개의 리스트 전부 똑같은 기능을 실행하게 되니

a라는 파라미터나 그런걸 이용해서 각각 리스트마다 다른 기능을 실행하게 만들 수 있습니다.

그건 나중에 해볼 예정이지만 심심하면 강의에서 설명한 따봉기능 직접 완성해보시길 바랍니다.

**일반 for 반복문을 쓰고싶다면**

그냥 따로 함수를 만들어서 쓰셔야합니다.

1. 따로 일반 함수를 만들고

2. 함수안에 HTML을 담을 array 자료를 하나 생성합니다.

3. 함수안에서 for 반복문을 이용해 array내에 HTML을 추가해줍니다.

4. 완성된 array를 return 해줍니다.

5. 그리고 함수를 원하는 곳에 { 함수명() } 데이터바인딩 해주시면 됩니다.

예를 들면 이런 식입니다.

function App (){

function 반복된UI(){

var 어레이 = [];

for (var i = 0; i < 3; i++) {

어레이.push(<div>안녕</div>)

}

return 어레이

}

return (

<div>

HTML 잔뜩있는 곳

{ 반복된UI() }

</div>

)

}

▲ 위의 예제코드는 for 반복문을 이용해서 div를 3개 만들어보는 예제입니다.

일반 for 반복문은 이런 식으로 함수를 만들고 안에서 쓰는 방법밖엔 없습니다.

for in / for of도 마찬가지입니다.

함수만들기 귀찮으니 코딩꼰대가 아니라면 그냥 map을 쓰도록 합시다.

**(참고) 왜 <div>들을 array에 담냐면**

그냥 여러가지 HTML들을 한곳에 보관하고 싶으면 array 자료형에 보관하시면 됩니다.

그니까 <div></div>를 세개 담고싶으면

**[ <div></div>, <div></div>, <div></div> ]**

이렇게 쭉 담아도 된다는 소리입니다. 이걸 원하는 곳에서 {데이터바인딩} 하시면 div 3개가 연달아서 잘 보입니다.

리액트에선 그냥 HTML들, 정확히 말하면 JSX들을 그냥 array에 담겨있어도 잘 렌더링해줘서 이렇게 쓸 수 있는 것입니다.