**배운 내용 요약**

function App() {

    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);

    // 변수

    let [누른제목, 누른제목변경] = useState(0);

    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>

            {

                글제목.map(function(글, i){

                    return(

                        <div className = 'list' >

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

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

                        <hr/>

                    </div>

                    )

                })

            }

            {/\* 버튼을 누르면 각각 다른 모달창이 뜨게 할거임 \*/}

            <button onClick={()=>{ 누른제목변경(0) }}>버튼1</button>

            <button onClick={()=>{ 누른제목변경(1) }}>버튼2</button>

            <button onClick={()=>{ 누른제목변경(2) }}>버튼3</button>

            <button onClick={()=>{modal변경(!modal)}}>열고닫기</button>

            {

                modal===true

                // <Modal 작명 = {전송할state} ></Modal>

                ? <Modal 글제목 = {글제목} 누른제목={누른제목}></Modal>

                : null

            }

        </div>

    );

}

// 제목을 누를 때 각각 다른 모달창이 뜨게

// 모달창 만드는 법

// 모달창 보이는/안보이는 상태정보를 state에 저장하고

    // state가 0일때는 0번째 제목 출력, 1일때는 1번째 제목 출력

// ui 만드는 법

// ui와 관련된 중요 정보들을 state로 저장해놓고

// state에 따라서 ui가 수정되게 만들면 됨

function Modal(props){

    return(

        <div className="modal">

            <h2>{props.글제목[props.누른제목]}</h2>

            <p>날씨</p>

            <p>상세내용 </p>

        </div>

    )

}

props 문법을 배워봤는데, 이걸 이용해서 상세페이지스러운 UI 하나만 만들어보도록 합시다.

사이트 내에 만든 <h3> 글제목들을 누르면 모달창이 열리는데

0번째 <h3>를 누르면 0번째 글제목이 모달창 안에 떠야합니다.

1번째 <h3>를 누르면 1번째 글제목이 모달창 안에 떠야합니다.

그냥 모달창 안에 상세페이지 같은걸 만들어보자는 겁니다.

어떻게 만들면 될까요?

지금부터 설명할 **UI 제작 패턴**만 외워두시면 어떤 UI든 직접 다 만들어내실 수 있습니다.

**예전에 UI 만들 때 어떻게 만들었습니까**

모달창 어떻게 만들었었습니까.

1. 모달창이 현재 보이는/보이지않는 상태정보를 state로 저장해놓고

2. state가 true일 때 모달창 보여주라고 if문 썼었고

3. state가 false일 때 모달창 숨기라고 if문 썼었습니다.

그럼 지금 UI는 어떻게 만들어야할까요.

UI만드는 패턴은 다 비슷합니다.

1. 지금 0,1,2번째 중 몇번째 제목을 눌렀는지 상태정보를 state로 저장해놓고

2. state가 0일 때는 0번째 제목을 저기 출력해주고

3. state가 1일 때는 1번째 제목을 저기 출력해주고

이러면 되지 않을까요?

그냥 UI 관련 정보를 state로 저장해둔 뒤에 state가 달라지면 UI도 달라지게 만들면 됩니다.

패턴은 다 똑같으니 다른 모든 UI 만들때도 한번 적용해보시길 바랍니다.

**글말고 따로 버튼3개를 만들어서 한번 개발해봅시다.**

h3 글을 누르면 뭐 변해야하는데 거기다 개발하면 일단 복잡하니까

<button>을 따로 3개 만들어서 글을 대신해보도록 합시다.

각각 버튼을 누르면 각각 다른 제목의 모달제목이 떠야합니다.

일단 버튼 만들고 생각해봅시다.

function App (){

return (

<div>

...

<button onClick={()=>{}}>버튼1</button>

<button onClick={()=>{}}>버튼2</button>

<button onClick={()=>{}}>버튼3</button>

모달 어쩌구 있는곳 <Modal></Modal>

</div>

)

}

function Modal(props){

return (

<div className="modal">

<h2>제목 { props.글제목[방금누른글번호] }</h2>

<p>날짜</p>

<p>상세내용</p>

</div>

)

}

1. 일단 버튼을 3개 만들었습니다.

2. 그리고 각각 버튼을 누르면 글제목이 수정되어야하기 때문에.. Modal이라는 컴포넌트안에

제목부분을 **props.글제목[방금누른글번호]** 라고 수정했습니다.

말되죠?

그럼 **방금누른글번호**라는 변수가 0이면 0번째 제목이 뜰거아닙니까.

그럼 **방금누른글번호**라는 변수가 1이면 1번째 제목이 뜰거아닙니까.

방금누른글번호라는 변수는 어디다 만들까요?

중요하니까 App안에 만들어봅시다.

그리고 중요한 정보는 일반 변수가 아니라 state로 만들랬죠?

state를 만들어봅시다.

function App (){

let [누른제목, 누른제목변경] = useState(0);

return (

<div>

...

<button onClick={()=>{}}>버튼1</button>

<button onClick={()=>{}}>버튼2</button>

<button onClick={()=>{}}>버튼3</button>

모달 어쩌구 있는곳 <Modal></Modal>

</div>

)

}

function Modal(props){

return (

<div className="modal">

<h2>제목 { props.글제목[누른제목] }</h2>

<p>날짜</p>

<p>상세내용</p>

</div>

)

}

1. 그래서 App안에**누른제목**이라는 이름의 state를 만들고 기본값을 0으로 주었습니다.

이게 지금 **몇번째 글제목 눌렀는지**의 정보를 보관하는 곳이라고 보시면 되겠습니다.

2. 그리고 모달창 안에**props.글제목[누른제목]** 이라고 사용했습니다.

그럼 이제 모달창 제목이 의도대로 보일 것입니다.

근데 이렇게 쓰면 에러가나죠? 왜입니까.

부모가 가진 state를 쓰려면 props로 신고하고 써야하니까 그렇죠.

그래서 신고해보겠습니다. props 쓸 때는 2개의 step이 있다고 배웠었죠?

function App (){

let [누른제목, 누른제목변경] = useState(0);

return (

<div>

...

<button onClick={()=>{}}>버튼1</button>

<button onClick={()=>{}}>버튼2</button>

<button onClick={()=>{}}>버튼3</button>

모달 어쩌구 있는곳 <Modal 글제목={글제목} 누른제목={누른제목} ></Modal>

</div>

)

}

function Modal(props){

return (

<div className="modal">

<h2>제목 { props.글제목[props.누른제목] }</h2>

<p>날짜</p>

<p>상세내용</p>

</div>

)

}

1. <Modal>이라는 태그 안에서 원하는 이름의 props를 전송하고

2. 모달 안에서 props.이름 이런 식으로 써야한다고 했습니다. 그래서 해본겁니다.

이제 모달창은 **누른제목**이라는 state의 숫자에 따라서 제목이 변경됩니다.

state에 0,1,2 이런 식의 숫자를 부여하면 각각 다른 제목이 뿅 뜨겠군요.

이제 여러분이 할 일은 버튼을 눌렀을 때 state를 변경하는 것입니다.

그럼 기능이 완성되겠는데요.

**버튼을 눌렀을 때 state를 변경하려면?**

0번째 버튼을 누르면 **누른제목**이라는 state가 0이 되어야합니다.

1번째 버튼을 누르면 **누른제목**이라는 state가 1이 되어야합니다.

어떻게 코드를 짜면 되겠습니까.

function App (){

let [누른제목, 누른제목변경] = useState(0);

return (

<div>

<button onClick={()=>{ 누른제목 = 0 }}>버튼1</button>

<button onClick={()=>{ 누른제목 = 1 }}>버튼2</button>

<button onClick={()=>{}}>버튼3</button>

</div>

)

}

이런 식으로 짜면 되겠죠?

성공입니다. 기능개발 끝!

는 함정이었습니다.

알려드리는 코드를 그대로 아무 생각없이 복붙하시면 안됩니다. 언제나 의심해보십시오.

state를 변경할 땐 어떻게 하라고 했습니까. 등호쓰면 안됩니다.

state 변경함수를 쓰셔야 변경이 됩니다.

function App (){

let [누른제목, 누른제목변경] = useState(0);

return (

<div>

<button onClick={()=>{ 누른제목변경(0) }}>버튼1</button>

<button onClick={()=>{ 누른제목변경(1) }}>버튼2</button>

<button onClick={()=>{ 누른제목변경(2) }}>버튼3</button>

</div>

)

}

이렇게 써야합니다. 정말 끝입니다.

이제 버튼말고 직접 <h3> 글제목부분에 가셔서 누르면 state가 변경되게 만들어보십시오.

근데 반복문으로 되어있네요 이런

**<h3> 글제목부분에 가셔서 누르면 state가 변경되게 만들어봅시다**

h3 글제목들을 누르면 누른제목이라는 state를 변경해야합니다.

어떻게 코드짜면 되죠?

function App (){

return (

<div>

...

{ 글제목.map(function(a){

return (

<div className="list">

<h3 onClick={ ()=>{ 누른제목변경(0) } }>{ a } <span>?</span> </h3>

<p>2월 18일 발행</p>

<hr />

</div>

}) }

</div>

)

}

▲ 뭐 클릭했을 때 뭔가 동작하게 하려면 onClick말고 더 있습니까.

그리고 그 안에 state변경함수를 집어넣었습니다.

대충 0이라고 써넣었기 때문에 지금은 어떤 제목을 누르든간에 state가 0으로 변경됩니다.

그럼 0이 아니라 각각 제목들마다

**누른제목변경(0)**

**누른제목변경(1)**

**누른제목변경(2)**

의 코드를 실행하게 하려면 어떻게 해야할까요?

0을 하드코딩하지말고

**누른제목변경(반복문돌때마다 차례로 0,1,2가 되는 변수)**

이걸 집어넣으면 되겠죠뭐..!

그런 변수가 어딨냐고요?

모든 반복문을 보시면 항상 그런 변수가 있습니다.

function App (){

return (

<div>

...

{ 글제목.map(function(a, i){

return (

<div className="list">

<h3 onClick={ ()=>{ 누른제목변경(i) } }>{ a } <span>?</span> </h3>

<p>2월 18일 발행</p>

<hr />

</div>

}) }

</div>

)

}

▲ map 반복문을 쓸 때 i라는 파라미터를 뒤에 추가해주시면 됩니다.

두번째 파라미터는 바로 반복문이 돌면서 0,1,2,3 ... 이렇게 하나씩 증가하는 정수를 뜻합니다.

전문 개발용어로 enumerate한다 라고 하는데 그건 알필요가 없고

그냥 i 저렇게 써주시면 0,1,2,3,4... 이런 변수입니다.

그래서 그걸 이용해서 **누른제목변경(i)** 이렇게 입력한 것입니다.

기능개발 끝!

(결론) 그래서 UI만드는법은 다 똑같습니다.

1. state를 하나 만들고

2. state가 요런 상태면 UI를 이렇게 보여주세요~ 라고 코드 짜놓으면 끝입니다.

3. 그리고 버튼 누르거나할 땐 state를 이렇게 바꿔주세요~ 이런것도 필요하면 추가하고요.

**리액트 안쓰고 썡자바스크립트만으로 개발**할 땐

1. 그냥 버튼누르면 HTML을 이렇게 수정해주세요~ 이렇게 코드 짜는게 끝이었을겁니다.

그래서 옛날에 코딩하던 아조씨들일 수록 기존관습과 달라서 어려워합니다.

그럴땐 **"state는 UI의 현재상태를 보관하는 저장소역할을 하는구나"**만 기억하시면 쉽습니다.