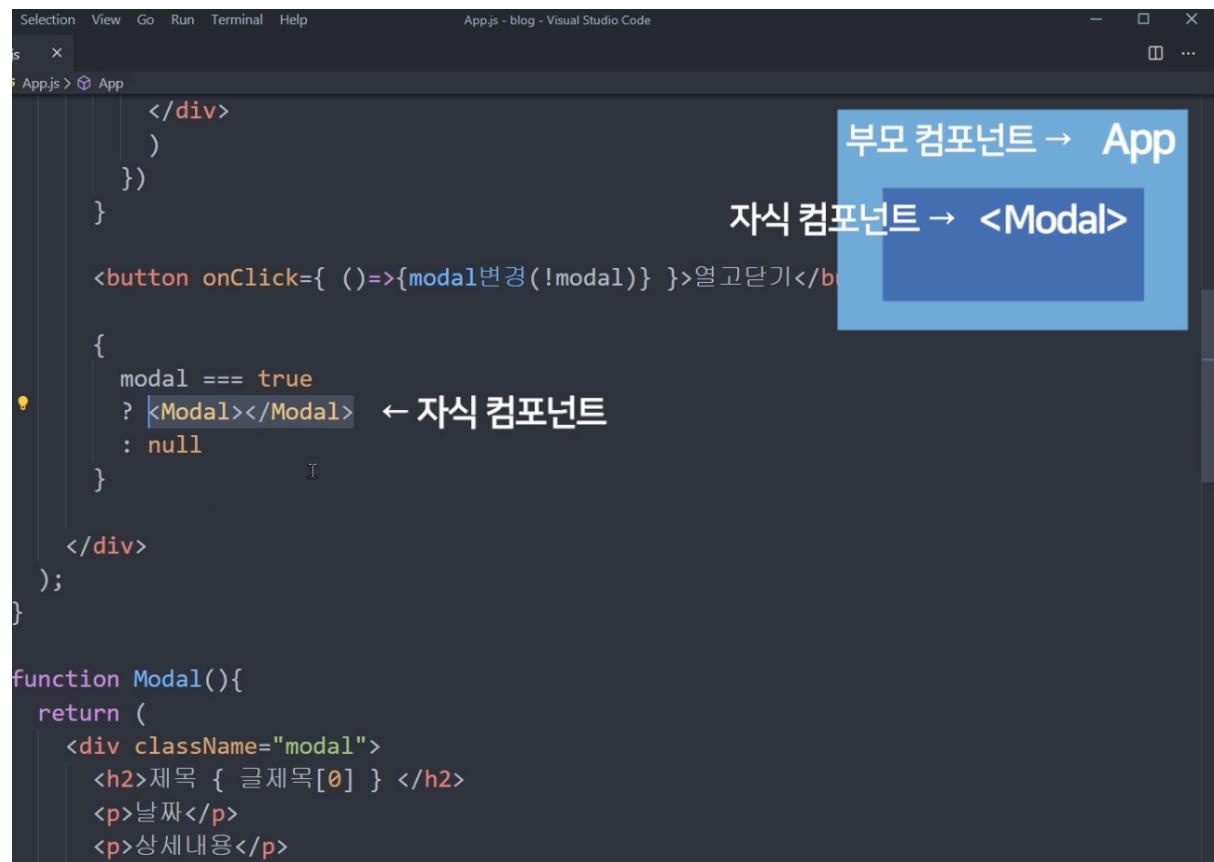


# 9월 4일 (토)

## 1. 코딩애플 - 리액트 강의

10) props : 자식이 부모의 state를 가져다 쓰고 싶을땐 말하고 쓰셔야 합니다.



```
Selection View Go Run Terminal Help          App.js - blog - Visual Studio Code
App.js > App
      </div>
    )
  })
}

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

{
  modal === true
  ? <Modal></Modal> ← 자식 컴포넌트
  : null
}

</div>
);
}

function Modal(){
  return (
    <div className="modal">
      <h2>제목 { 글제목[0] } </h2>
      <p>날짜</p>
      <p>상세내용</p>
    
```

부모 컴포넌트 → App

자식 컴포넌트 → <Modal>

자식컴포넌트에게 부모컴포넌트 state 를 props 키워드를 통해 전해줄 수 있다.

```
Selection View Go Run Terminal Help • App.js - blog - Visual Studio Code
App.js > App
    <p>작성자: 토니 클링턴</p>
    <hr/>
    </div>
)
})
}

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

{
  modal === true
  ? <Modal 작명=전송할state ></Modal>
  : null
}

</div>
);
}

function Modal(){
  return (
    <div className="modal">
      <h2>제목 { 글제목[0] } </h2>
      <p>내용</p>
    
```

props로 자식에게 state 전해주는 법  
1. <자식컴포넌트 작명={state명} />

원하는 컴포넌트를 전송하는 법 - step1. 하위 컴포넌트에 위 문법을 쓴다.

```
Selection View Go Run Terminal Help App.js - blog - Visual Studio Code
App.js > App
    }
}

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

{
  modal === true
  ? <Modal 글제목={글제목}></Modal>
  : null
}

</div>
);
}

function Modal(){
  return (
    <div className="modal">
      <h2>제목 { 글제목[0] } </h2>
      <p>날짜</p>
      <p>상세내용</p>
    </div>
  )
}

```

**props로 자식에게 state 전해주는 법**  
**1. <자식컴포넌트 작명={state명} />**

보통 작명은 전해주는 state와 똑같이 쓴다.

```

src > JS App.js > Modal
    <button onClick={()=>{modal변경(!modal)}}>열고닫기</button>

    {
        modal === true
        ? <Modal 글제목={글제목}></Modal>
        : null
    }

    </div>
);
}

▼ 부모에서 전달받은 props는 여기에 다 들어있음

function Modal(props){
    return (
        <div className="modal">
            <h2>제목 { 글제목[0] } </h2>
            <p>날짜</p>
            <p>상세내용</p>
        </div>
    )
}

```

props로 자식에게 state 전해주는 법

1. <자식컴포넌트 작명={state명} />
2. 자식컴포넌트에서 props 파라미터 입력 후 사용

step 2. 부모에서 전달받은 props 를 인자안에 적는다.

```
src > JS App.js > Modal
```

```
        }
      <button onClick={()=>{modal변경(!modal)}}>열고닫기</button>
    {
      modal === true
      ? <Modal 글제목={글제목}></Modal>
      : null
    }
  </div>
);
}

function Modal(props){
  return (
    <div className="modal">
      <h2> { props.글제목[1] } </h2>
      <p>날짜</p>
      <p>상세내용</p>
    </div>
  )
}
```

props로 자식에게 state 전해주는 법  
1. <자식컴포넌트 작명={state명} />  
2. 자식컴포넌트에서 props 파라미터 입력 후 사용

step 3. 사용할 state 앞에 props 를 붙이고 사용한다.

## 11) (UI 제작 패턴) props를 응용한 상세페이지 만들기

UI 만드는 법칙

The screenshot shows a code editor with a dark theme. The file is named `App.js`. The code defines a `Modal` component and a main `App` component. The `App` component contains a button to open the modal and a conditional rendering block. The `Modal` component displays a title, a note, and a detailed description.

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

{
  modal === true
  ? <Modal 글제목={글제목} ></Modal>
  : null
}

</div>
);

function Modal(props){
  return [
    <div className="modal">
      <h2> { props.글제목[1] } </h2>
      <p>날짜</p>
      <p>상세내용</p>
    </div>
  ]
}

```

각각 다른 모달창 제목 만드는 법  
- 몇번째 제목 눌렀는지 상태정보를 state에 저장하고  
- state가 0일 때는 0번째 제목 출력  
- state가 1일 때는 1번째 제목 출력

Q. 제목을 누를 때  
각각 다른 모달창이 뜨게?

The screenshot shows a code editor with a dark theme. The file is named `App.js`. The code defines a `Modal` component and a main `App` component. The `App` component contains a button to open the modal and a conditional rendering block. The `Modal` component displays a title, a note, and a detailed description.

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

{
  modal === true
  ? <Modal 글제목={글제목} ></Modal>
  : null
}

</div>
);

function Modal(props){
  return [
    <div className="modal">
      <h2> { props.글제목[1] } </h2>
      <p>날짜</p>
      <p>상세내용</p>
    </div>
  ]
}

```

UI 만드는 법 :  
1. UI와 관련된 중요 정보들을 state로 저장해놓고  
2. state에 따라서 UI가 수정되게 만들면 됩니다

Q. 제목을 누를 때  
각각 다른 모달창이 뜨게?

(Tip) 행 복사 : Alt + Shift + down(버튼)

The screenshot shows a code editor window with a dark theme. On the left is a sidebar with icons for file, search, and other navigation. The main area contains two files: 'App.js' at the top and 'Modal' below it. The 'Modal' file's code is as follows:

```
JS App.js
src > JS App.js > Modal
<button onClick={()=>{modal변경(!modal)}}>클릭하기</button>


{
    modal === true
    ? <Modal 글제목={글제목}></Modal>
    : null
  }


);
}

function Modal(props){
  return (
    <div className="modal">
      <h2> { props.글제목[2] } </h2>
      <p>날짜</p>
      <p>상세내용</p>
    </div>
  )
}
```

Annotations on the right side of the code:

- 0번째 버튼을 누르면 props.글제목[0]
- 1번째 버튼을 누르면 props.글제목[1]
- 2번째 버튼을 누르면 props.글제목[2]

Below the annotations is a question:

Q. 버튼을 누를 때  
각각 다른 모달창이 뜨게?

```
JS App.js
src > JS App.js > Modal
    <button onClick={()=>{modal=0}}>0번 버튼</button>
    <button onClick={()=>{modal=1}}>1번 버튼</button>
    <button onClick={()=>{modal=2}}>2번 버튼</button>
  </div>
)
}

function Modal(props){
  return (
    <div className="modal">
      <h2> { props.글제목[내가방금누른제목넘버] } </h2>
      <p>날짜</p>
      <p>상세내용</p>
    </div>
  )
}
```

0번째 버튼을 누르면 props.글제목[0]  
1번째 버튼을 누르면 props.글제목[1]  
2번째 버튼을 누르면 props.글제목[2]

Q. 버튼을 누를 때  
각각 다른 모달창이 뜨게?

변화가 있어야하는 곳에 변수를 하나 지정해준다.

The screenshot shows a code editor window with a dark theme. The file is named 'App.js' and is located in a 'src' directory. The code uses React's useState hook to manage state for a list of items, a counter, and a modal. It also includes a click event handler for one of the list items.

```
JS App.js
src > JS App.js > App
import React, { useState } from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  let [글제목, 글제목변경] = useState(['남자코트 추천', '강남 우동맛집',
  '파이썬독학']);
  let [따봉, 따봉변경] = useState(0);
  let [modal, modal변경] = useState(true);
  let [누른제목, 누른제목변경] = useState(0);

  return [
    <div className="App">
      <div className="black-nav">
        <div>개발 Blog</div>
      </div>

      {
        글제목.map(function(글){
          return (
            <div className="list">
              <h3> { 글 } <span onClick={()=>{ 따봉변경(따봉 + 1) }}>👍</span>
              {따봉} </h3>
            </div>
          );
        })
      }
    </div>
  ];
}

export default App;
```

Q. 버튼을 누를 때  
각각 다른 모달창이 뜨게?

누른제목이라는 변수를 만들어준다.

The screenshot shows a code editor window with a dark theme. The file is named 'App.js'. The code defines a 'Modal' component and uses it within another component. A tooltip or callout box is overlaid on the right side of the screen, containing the text 'Q. 버튼을 누를 때 각각 다른 모달창이 뜨게?' (Q. When you click the button, how can you make different modals appear?).

```
JS App.js x
src > JS App.js > Modal

    {
      modal === true
      ? <Modal 글제목={글제목} 누른제목={누른제목}></Modal>
      : null
    }

  </div>
}

function Modal(props){
  return (
    <div className="modal">
      <h2> { props.글제목[누른제목] } </h2>
      <p>날짜</p>
      <p>상세내용</p>
    </div>
  )
}

```

하지만 Modal 안에 해당 변수를 사용하지 못하므로 props 를 사용하기 위해 Modal 에 해당 변수를 지정 해줘야 한다.

The screenshot shows the same code editor window after making changes. The '누른제목' prop is now explicitly passed from the parent component to the Modal component. The tooltip 'Q. 버튼을 누를 때 각각 다른 모달창이 뜨게?' is no longer visible.

```
JS App.js x
src > JS App.js > App

    {
      modal === true
      ? <Modal 글제목={글제목} 누른제목={누른제목}></Modal>
      : null
    }

  </div>
}

function Modal(props){
  return (
    <div className="modal">
      <h2> { props.글제목[누른제목] } </h2>
      <p>날짜</p>
      <p>상세내용</p>
    </div>
  )
}

```

```

        }

function Modal(props){
  return (
    <div className="modal">
      <h2> { props.글제목[props.누른제목] } </h2>
      <p>날짜</p>
      <p>상세내용</p>
    </div>
  )
}

Q. 버튼을 누를 때  

각각 다른 모달창이 뜨게?

```

이제 변수를 Modal 함수내에서 사용할 수 있게 되었다.



```

JS App.js
src > JS App.js > App > 글제목.map() callback

{
  글제목.map(function(글, i){
    return [
      <div className="list">
        <h3 onClick={()=>{ 누른제목변경(0) }}> { 글 } <span onClick={()=>{ 따봉변경(따봉 + 1) }}>👍</span> {따봉} </h3>
        <p>2월 18일 발행</p>
        <hr/>
      </div>
    ]
  }
}

```

map 인자 안에 있는 콜백함수의 i 인자는 0..1..2..등 숫자를 돌린다. (이유는 잘 모르겠음)

## 12) input 다루기 1 : 사용자가 입력한 글을 변수에 저장하는 법

html에서 input 태그는 단일태그이지만, 리액트에서는 꼭 엔드태그를 가지고 있어야 한다.

- 인풋에 입력된 내용 저장하는 방법

일단 저장공간이 필요하므로 state를 하나 만들자.

```
js App.js x
src > js App.js > App
import './App.css';

function App() {

  let [글제목, 글제목변경] = useState(['남자코트 추천', '강남 우동맛집',
  '파이썬독학']);
  let [따봉, 따봉변경] = useState(0);
  let [modal, modal변경] = useState(true);
  let [누른제목, 누른제목변경] = useState(0);

  let [입력값, 입력값변경] = useState(''); (저장공간)

  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) }} > { 글 } <span onClick={()=>
              { 따봉변경(따봉 + 1) }}>👍</span> {따봉} </h3>

```

사용자가 input에 입력한 값을 state로 저장해보자

```
js App.js x
src > js App.js > App
import './App.css';

function App() {

  let [글제목, 글제목변경] = useState(['남자코트 추천', '강남 우동맛집',
  '파이썬독학']);
  let [따봉, 따봉변경] = useState(0);
  let [modal, modal변경] = useState(true);
  let [누른제목, 누른제목변경] = useState(0);

  let [입력값, 입력값변경] = useState(''); ▲ 초기값

  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) }} > { 글 } <span onClick={()=>
              { 따봉변경(따봉 + 1) }}>👍</span> {따봉} </h3>

```

사용자가 input에 입력한 값을 state로 저장해보자

```
JS App.js
src > JS App.js > App
  ...
  { 땠봉변경(땀봉 + 1) } }>👍</span> {땀봉} </h3>
  <p>2월 18일 발행</p>
  <hr/>
  </div>
)
}
}

▼ 뭔가 입력이 될 때 안의 함수가 실행됨
<input onChange={ ()=>{} } />

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

{
  modal === true
  ? <Modal 글제목={글제목} 누른제목={누른제목} ></Modal>
  : null
}
</div>
);
}
```

사용자가 input에 입력한 값을  
입력값 state로 저장해보자

html에서는 onInput 과 onChange 의 기능을 나누지만 리액트에서는 같은 기능으로 치기에 onChange 라고 하면 된다.

```
App.js
src > App.js > App
    { 땠봉변경(땀봉 + 1) } }>👍 </span> {땀봉} </h3>
    <p>2월 18일 발행</p>
    <hr/>
    </div>
)
}
}

<input onChange={()=>{ console.log('안녕') }} />

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

{
    modal === true
    ? <Modal 글제목={글제목} 누른제목={누른제목} ></Modal>
    : null
}
</div>
);
}
```

사용자가 input에 입력한 값을  
입력값 state로 저장해보자

인풋 안에 한 글자만 써도 안녕이라는 문자열이 콘솔창에 출력된다.

### 사용자가 입력한 값을 가져오는 법

```
이벤트 동작한 곳.value
<input onChange={()=>{ e.target.value }} />
```

e.target 은 이벤트가 동작한 곳을 뜻하며

```
▼ input에 입력된 값
<input onChange={()=>{ e.target.value }} />
```

value는 input에 입력된 값을 의미한다.

e라는 event 객체를 사용할 땐 파라미터에 e를 넣어줘야 한다.

```
<input onChange={(e)=>{ console.log( e.target.value ) }} />

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

{
  modal === true
  ? <Modal 글제목={글제목} 누른제목={누른제목}></Modal>
  : null
}

</div>
);
```

input에 입력된 값이 계속해서 콘솔창에 출력이 된다.

```
<input onChange={(e)=>{ 입력값변경(e.target.value) }} />

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

{
  modal === true
  ? <Modal 글제목={글제목} 누른제목={누른제목}></Modal>
  : null
}

</div>
);
```

입력값을 state에 저장하기 위한 방법이다.

map 반복문으로 구성한 div 태그에 key 입력하기

## 개발 Blog

남자코트 추천  0

2월 18일 발행

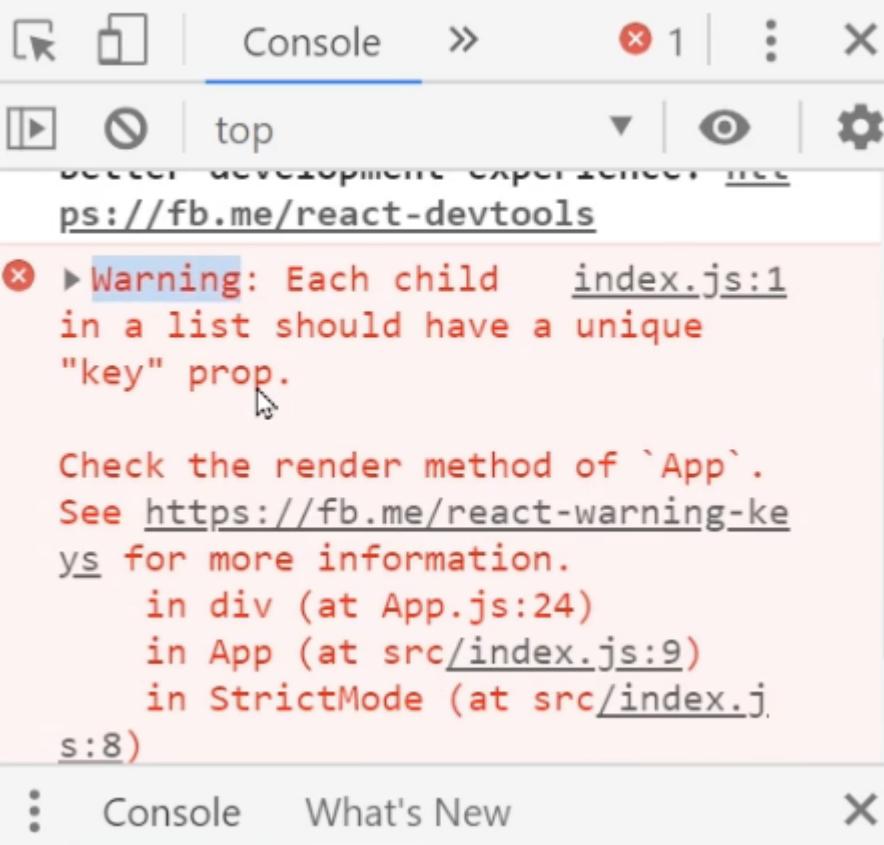
강남 우동맛집  0

2월 18일 발행

파이썬독학  0

2월 18일 발행

열고닫기



key를 입력하지 않아서 뜨게 되는 warning

The screenshot shows a browser window displaying a blog application titled "개발 Blog". The sidebar lists posts: "남자코트 추천" (0 likes), "2월 18일 발행"; "강남 우동맛집" (0 likes), "2월 18일 발행"; and "파이썬독학" (0 likes), "2월 18일 발행". A modal window titled "열고닫기" (Open/Close) is visible. Below the browser is a code editor with the following React component code:

```
src > App.js > App > 글제목.map() callback


<div>개발 Blog</div>



{
  글제목.map(function(글, i){
    return (
      <div className="list" key={i}>
        <h3 onClick={()=>{ 누른제목변경(i) }}> { 글 } <span onClick={()=>{ 따봉변경(따봉 + 1) }}>👍</span> { 따봉 } </h3>
        <p>2월 18일 발행</p>
        <hr/>
      </div>
    )
  })
}

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

{
  modal === true
}
```

The code editor highlights the line `<div className="list" key={i}>` with a red bracket, indicating a warning. To the right of the code, a text box contains the message: "map 반복문으로 돌린 HTML에는 key={}가 필요합니다" (key={} is required for HTML generated by map loops).

map 으로 구성한 div 태그들은 각자의 key 를 가지고 있어야 한다. key 를 입력하지 않으면 warning 이 뜨게 된다. (에러는 아님 )

## 13) input 다루기 2 : 블로그 글 발행 기능 만들기

```
        })
    }

<div className="publish">
    <input onChange={ (e)=>{ 입력값변경(e.target.value) } } />
    <button onClick={ ()=>{
        var arrayCopy = [...글제목];
        arrayCopy.unshift(입력값);
        글제목변경( arrayCopy );
    } }>저장</button>
</div>

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

{

```

글발행기능만들기

- 사용자가 입력한 글 state로 저장하기
- (실전) 서버로 먼저 보내서 영구저장하고..
- 버튼누르면 입력한 글 state를 글제목 state에 추가

Ln 8, Col 64 (34 selected) Spaces: 2 UTF-8 LF JavaScript ⌂ ⌂

## 14) class 를 이용한 옛날 옛적 React 문법

현업에서 이미 짜져있는 리액트 코드를 수정할 일도 있으니 배워둬야 한다.

- 컴포넌트를 지정하고 state 를 만드는 법

```
<p>상세내용</p>
      </div>
    )
}

class Profile extends React.Component {
  constructor(){
    super();
  }

  render(){
    return (
      <div>프로필입니다</div>
    )
  }
}

- component 만드는 기본 문법
예전 리액트 문법
```

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

      <Profile/>
    {
      modal === true
      ? <Modal 글제목={글제목} 누른제목={누른제목}></Modal>
      : null
    }
  </div>
};

function Modal(props){
  return (
    <div className="modal">
      <h2> { props.글제목[props.누른제목] } </h2>
      <p>날짜</p>
      <p>상세내용</p>
    </div>
  );
}

- component 만드는 기본 문법
예전 리액트 문법
```

사용법은 똑같다.

```
        }

class Profile extends React.Component {
  constructor(){
    super();
  }

  render(){
    return (
      <div>프로필입니다</div>
    )
  }
}

Ln 73, Col 1  Spaces: 2  UTF-8  LF  JavaScript  ⚙  ⌂
```

class : 변수/함수 보관하는 덩어리

```
        }

    > class Profile extends React.Component {
    >     constructor(){
    >         super();
    >         this.state = { name : 'Kim' }
    >     }

    >     render(){
    >         return (
    >             <div>프로필입니다</div>
    >         )
    >     }
    >
    >
    >
```

state는 constructor 안에 작성

state 지정하는 방법 this.state = {객체} 형식으로 지정해준다.

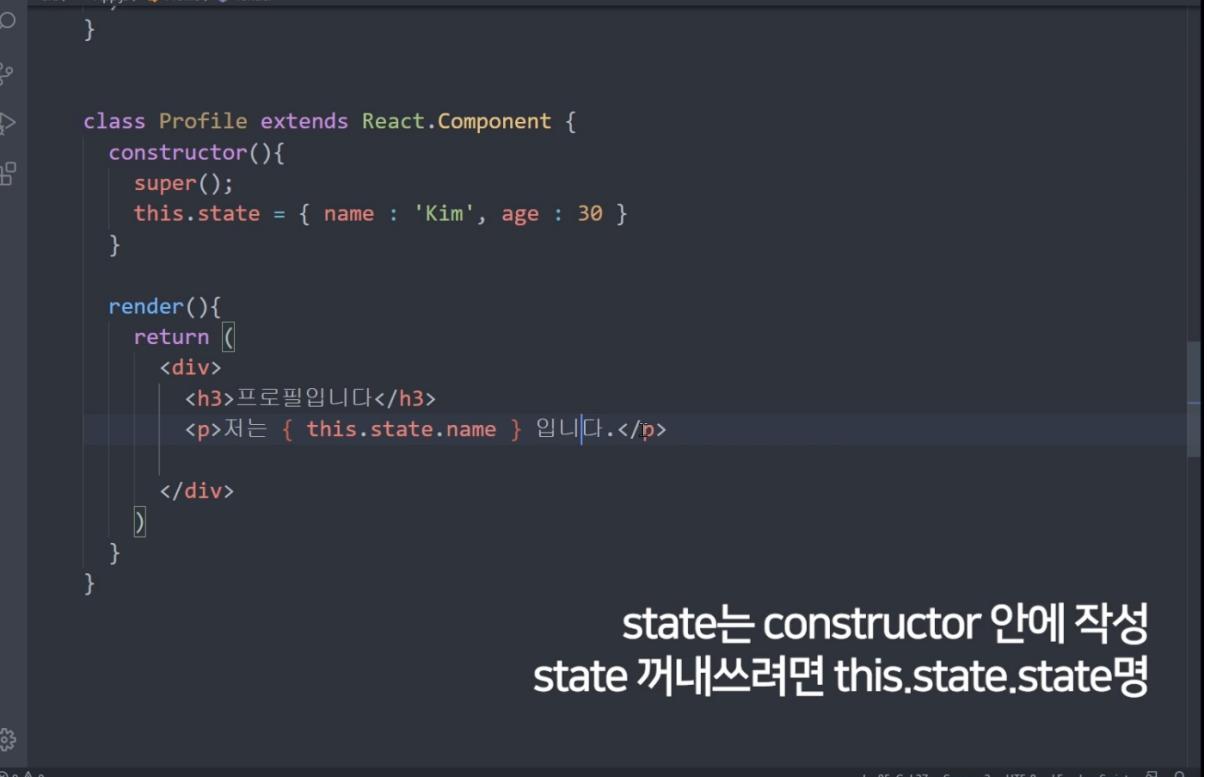
```
        }

    > class Profile extends React.Component {
    >     constructor(){
    >         super();
    >         this.state = { name : 'Kim' }
    >     }

    >     render(){
    >         return (
    >             <div>프로필입니다</div>
    >         )
    >     }
    >
    >
```

constructor : class의 변수/초기값 저장할 때 씁니다

- state 꺼내 쓰는 법



```
        }

    class Profile extends React.Component {
        constructor(){
            super();
            this.state = { name : 'Kim', age : 30 }
        }

        render(){
            return [
                <div>
                    <h3>프로필입니다</h3>
                    <p>저는 { this.state.name } 입니다.</p>
                </div>
            ]
        }
    }

```

state는 constructor 안에 작성  
state 꺼내쓰려면 this.state.state명

Ln 85, Col 37 Spaces: 2 UTF-8 LF JavaScript

```
        }

    > class Profile extends React.Component {
      constructor(){
        super();
        this.state = { name : 'Kim', age : 30 }
      }

      render(){
        return [
          <div>
            <h3>프로필입니다</h3>
            <p>저는 { this.state.name } 입니다.</p>
          </div>
        ]
      }
    }

Ln 86, Col 1  Spaces: 2  UTF-8  LF  JavaScript  ⚙  ⌂
```

오늘의 교훈) 요즘 문법이 편하군

- state 변경함수

```
        }

    > class Profile extends React.Component {
      constructor(){
        super();
        this.state = { name : 'Kim', age : 30 }
      }

      render(){
        return (
          <div>
            <h3>프로필입니다</h3>
            <p>저는 { this.state.name } 입니다.</p>
            <button onClick={ ()=>[ this.setState( {name: 'Park'} ) ] }>버튼</button>
          </div>
        )
      }
    }

  
```

- 이전 문법은 setState(변경할state)  
버튼을 누르면 state를 변경해보자

무조건 this 를 붙이고 setState 함수를 사용해야 한다.

```
class Profile extends React.Component {
  constructor(){
    super();
    this.state = { name : 'Kim', age : 30 }
  }

  render(){
    return (
      <div>
        <h3>프로필입니다</h3>
        <p>저는 { this.state.name } 입니다.</p>
        <button onClick={ ()=>{ this.setState({ name: 'Park' }) } }>버튼</button>
      </div>
    )
  }
}

- setState(변경할 state만 넣기)
버튼을 누르면 state를 변경해보자
```

신문법 useState 변경함수와 차이점: useState는 기존 변수를 완전히 바꿔버리는 반면, setState는 변경하고 싶은 값만 변경할 수 있기 때문에 변경에 있어서 훨씬 편하다.

- 함수 만들기

```
class Profile extends React.Component {
  constructor(){
    super();
    this.state = { name : 'Kim', age : 30 }
  }

  changeName(){
    this.setState( {name: 'Park'} )
  }

  render(){
    return (
      <div>
        <h3>프로필입니다</h3>
        <p>저는 { this.state.name } 입니다.</p>
        <button onClick={ }>버튼</button>
      </div>
    )
  }
}
```

함수만들기

생성자함수 constructor 와 render 사이에 함수를 만들어서 사용이 가능하다.

```
class Profile extends React.Component {
  constructor(){
    super();
    this.state = { name : 'Kim', age : 30 }
  }

  changeName(){
    this.setState( {name: 'Park'} )
  }

  render(){
    return (
      <div>
        <h3>프로필입니다</h3>
        <p>저는 { this.state.name } 입니다.</p>
        <button onClick={ this.changeName }>버튼</button>
      </div>
    )
  }
}
```

함수만들기

함수를 사용할 때도 this 를 꼭 붙여줘야한다!!

```
class Profile extends React.Component {
  constructor(){
    super();
    this.state = { name : 'Kim', age : 30 }
  }

  changeName(){
    this.setState( {name: 'Park'} )
  }

  render(){
    return (
      <div>
        <h3>프로필입니다</h3>
        <p>저는 { this.state.name } 입니다.</p>
        <button onClick={ this.changeName.bind(this) }>버튼</button>
      </div>
    )
  }
}
```

만약 에러가 뜨면 위와 같이 .bind(this) 를 적어주는데 이와 같은 방법이 귀찮다면 아래와 같이 해보자.

```
super();
this.state = { name : 'Kim', age : 30 }

changeName = () => [
  this.setState( {name: 'Park'} )
]

render(){
  return (
    <div>
      <h3>프로필입니다</h3>
    </div>
  )
}
```

이렇게 함수를 arrow 함수로 만들어주면 .bind(this) 를 안쓰고도 함수를 사용할 수 있다.

유지보수할 상황이 아니라면

신문법을 쓰자 !!!

공식문서에서도 컴포넌트를 만들 땐 신문법을 권장한다.

