

# 9월 3일 (금)

## 1. 코딩애플 - 리액트 강의 - Part 1 : 블로그 제작 & 기초 문법

### 3) JSX를 이용해서 HTML 페이지 제작하기

- 메인페이지 작동원리 : 왜 App.js 에 html을 입력할까?

⇒ app.js에서 입력한 App 함수를 index.js로 가져다가 다시 index.js에서 public 폴더에 있는 index.html의 root div에 띄운다.

- App 함수안의 return 안에 html 작성

- 리액트가 쌩 HTML코딩보다 편리한 이유

⇒ 데이터 바인딩이 쉽다. 데이터바인딩이란 서버에서 받아 온 데이터를 html에 꽂아 넣는것  
쌩 html에서는 DOM을 사용하여 변경  
{ 변수명 }를 사용하여 데이터 바인딩

```
File Edit Selection View Go Run Terminal Help
src > JS App.js > App
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  let posts = '강남 고기 맛집';

  return (
    <div className="App">
      <div className="black-nav">
        <div>개발 Blog</div>
      </div>
      <h4> { posts } </h4>
    </div>
  );
}

export default App;
```

**2. 리액트에서 데이터 바인딩 쉽게하는 법  
{ 변수명 }**

{ } 안에 함수도 들어갈 수 있다.

```
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  let posts = '강남 고기 맛집';

  return (
    <div className="App">
      <div className="black-nav">
        <div>개발 Blog</div>
      </div>
      <img src={ logo } />
      <h4> { posts } </h4>
    </div>
  );
}

export default App;
```

2. 리액트에서 데이터 바인딩 쉽게하는 법  
src, id, href 등의 속성에도 { 변수명, 함수 등 }

import 해온 logo 도 {} 을 사용하여 이미지를 불러올 수 있다. src, id, href 등의 속성에도 가능

상상하는 모든 곳에 {} 로 변수 집어넣기 가능

- JSX 에서 style 속성을 집어넣을 때

```
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  let posts = '강남 고기 맛집';

  return (
    <div className="App">
      <div className="black-nav">
        <div style={ { color : 'blue' } }>개발 Blog</div>
      </div>
      <h4> { posts } </h4>
    </div>
  );
}

export default App;
```

3. JSX에서 style 속성 집어넣을 때  
style={ object 자료형으로 만든 스타일 }

중괄호와 object 자료형으로 입력해야한다.

```
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  let posts = '강남 고기 맛집';

  return (
    <div className="App">
      <div className="black-nav">
        <div style={ { color : 'blue', fontSize : '30px' } }>개발 Blog</div>
      </div>
      <h4> { posts } </h4>
    </div>
  );
}

export default App;
```

camelCase 작명관습에 따라  
속성명을 바꿔주세요

3. JSX에서 style 속성 집어넣을 때  
style={ object 자료형으로 만든 스타일 }

```
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  let posts = { color: 'blue', fontSize: '30px' };
  return (
    <div className="App">
      <div className="black-nav">
        <div style={posts}>개발 Blog</div>
      </div>
      <h4> { posts } </h4>
    </div>
  );
}

export default App;
```

이렇게도 가능하다. 상상하는 이상 모든 것이 가능

## 4) 중요한 데이터는 변수말고 리액트 state로

- 데이터 `const, let` 변수에 보관 or state에 보관

The screenshot shows the Visual Studio Code interface with the `App.js` file open. The code uses the `useState` hook to manage state. A browser window on the left displays the application's UI, which includes a header "개발 Blog" and a main content area with the text "강남 고기 맛집" and "2월 17일 발행".

```

File Edit Selection View Go Run Terminal Help
src > JS App.js - blog - Visual Studio Code
JS App.js
import React, { useState } from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  let posts = '강남 고기 맛집';
  return (
    <div className="App">
      <div className="black-nav">
        <div>개발 Blog</div>
      </div>
      <div className="list">
        <h3> { posts } </h3>
        <p>2월 17일 발행</p>
        <hr/>
      </div>
    </div>
  );
}

export default App;

```

**1. `{ useState }` 상단에 첨부**

**리액트의 데이터 저장공간 state 만드는 법**

The screenshot shows the Visual Studio Code interface with the `App.js` file open. There is a syntax error at the line where `useState` is used, indicated by a red squiggle under the code. A browser window on the left displays the application's UI, which includes a header "개발 Blog" and a main content area with the text "강남 고기 맛집" and "2월 17일 발행".

```

File Edit Selection View Go Run Terminal Help
src > JS App.js - blog - Visual Studio Code
JS App.js
import React, { useState } from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  useState('남자 코트 추천'); [a,b]
  let posts = '강남 고기 맛집';
  return (
    <div className="App">
      <div className="black-nav">
        <div>개발 Blog</div>
      </div>
      <div className="list">
        <h3> { posts } </h3>
        <p>2월 17일 발행</p>
        <hr/>
      </div>
    </div>
  );
}

export default App;

```

**1. `{ useState }` 상단에 첨부**

**2. `useState(데이터)`**

**리액트의 데이터 저장공간 state 만드는 법**

위와 같이 `useState()` 함수를 사용하면 Array 가 하나 출력된다. Array [a,b] 중 a에는 '남자 코트 추천'이 들어가고, b에는 a를 변경할 수 있는 함수가 들어가게 된다.

```

Failed to compile

./src/App.js
Line 10:8  Parsing error: Identifier 'a' has already been declared

8 |
9 |
> 10 |   let [a,b] = useState('남자 코드 추천');
  |   ^
11 |
12 |   let posts = '강남 고기 맛집';
13 |

This error occurred during the build time and cannot be dismissed.

```

(참고) ES6 destructuring 문법

```

array, object에 있던 자료를  
변수에 쉽게 담고 싶을 때

```

```

JS App.js
File Edit Selection View Go Run Terminal Help
src > JS App.js > App
import React, { useState } from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  var [a,b] = [10, 100];
  var a = 10
  var b = 100

  let [a,b] = useState('남자 코드 추천');

  let posts = '강남 고기 맛집';
  return (
    <div className="App">
      <div className="black-nav">
        <div>개발 Blog</div>
      </div>
      <div className="list">
        <h3> { posts } </h3>
        <p>2월 17일 발행</p>
        <hr/>
      </div>
    </div>
  );
}

export default App;

```

디스트럭쳐링 신문법을 사용하여 useState로 출력한 배열의 값을 각각 a, b라는 변수에 지정해준다.

## state는

- ⇒ 변수 대신 쓰는 데이터 저장공간
- ⇒ useState( )를 이용해서 만들어야함.
- ⇒ useState( ) 안에 인자로 문자, 숫자, array, object 중요한 정보 모두 저장가능하다.

## state에 데이터를 저장해놓는 이유:

- ⇒ 웹이 App처럼 동작하게 만들고 싶어서
- ⇒ state는 변경되면 HTML이 새로고침 없이 자동으로 재렌더링이 된다. 그냥 변수로 지정하게 되면 변수가 변경되어도 자동 재렌더링이 안된다.

자주 바뀌고 중요한 데이터는 변수말고 state로 저장해서 쓰자.

잘 안바뀌는 블로그 제목등 이런 것은 변수에 넣어도 상관없다.

## 5) 리액트 state변경하는 법 & 버튼에 기능개발을 해보자.

- eslint warning 보기 싫을 때 끄는 방법

The screenshot shows a browser window displaying a simple blog application with posts. In the VS Code editor, the file 'App.js' is open. It contains the following code:

```
/* eslint-disable */
import React, { useState } from 'react';
import logo from './logo.svg';
```

- 리액트 이벤트 핸들러

The screenshot shows a browser window displaying a blog application with three posts. In the VS Code editor, the file 'App.js' is open. The code includes event handling for the first post's like button:

```
/* eslint-disable */
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 posts = '강남 고기 맛집';

  return (
    <div className="App">
      <div className="black-nav">
        <div>개발 Blog</div>
      </div>
      <div className="list">
        <h3> { 글제목[0] } <span onClick={()=>[]}>👍</span> 0 </h3>
        <p>2월 17일 발행</p>
        <hr/>
      </div>
      <div className="list">
        <h3> { 글제목[1] } </h3>
        <p>2월 18일 발행</p>
        <hr/>
      </div>
      <div className="list">
```

On the right side of the code editor, there is a callout with the text "리액트" at the top, followed by two lines of explanatory text:

onClick={ 클릭될 때 실행할 함수 }  
onClick={ ()=>{ 실행할 내용 } }

The screenshot shows a browser window titled "React App" displaying a blog application. The sidebar lists posts: "남자코트 추천" (0 likes, 2월 17일 발행), "강남 우동맛집" (0 likes, 2월 18일 발행), and "파이썬독학" (0 likes, 2월 19일 발행). The main content area shows the React code for rendering this list.

```

File Edit Selection View Go Run Terminal Help
src > App.js > App
/* eslint-disable */
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 [] = useState();
  let posts = '강남 고기 맛집';

  return (
    <div className="App">
      <div className="black-nav">
        <div>개발 Blog</div>
      </div>
      <div className="list">
        <h3> { 글제목[0] } <span onClick={ }>👍</span> 0 </h3>
        <p>2월 17일 발행</p>
        <hr/>
      </div>
      <div className="list">
        <h3> { 글제목[1] } </h3>
        <p>2월 18일 발행</p>
        <hr/>
      </div>
    </div>
  );
}

export default App;

```

**▼ state로 만드는게 좋겠군**

**Q. 따봉 누를 때마다 1 증가시키기**

The screenshot shows the same blog application and VS Code editor. The code has been modified to include logic for incrementing the like count when the "thumb-up" button is clicked.

```

File Edit Selection View Go Run Terminal Help
src > App.js > App
/* eslint-disable */
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 posts = '강남 고기 맛집';

  return (
    <div className="App">
      <div className="black-nav">
        <div>개발 Blog</div>
      </div>
      <div className="list">
        <h3> { 글제목[0] } <span onClick={ ()=>{ 따봉 + 1 } }>👍</span> { 따봉 } </h3>
        <p>2월 17일 발행</p>
        <hr/>
      </div>
      <div className="list">
        <h3> { 글제목[1] } </h3>
        <p>2월 18일 발행</p>
        <hr/>
      </div>
    </div>
  );
}

export default App;

```

**state 변경방법이 따로 있습니다  
따봉이랑 같이 만들었던 따봉변경()**

**Q. 따봉 누를 때마다 1 증가시키기**

따봉변경( 대체할 데이터 )

## 6) 숙제 해설: 블로그 글 수정버튼 만들기

- return 문 밖에 함수를 만들어서 onClick 에 넣을 수 있다.

```
<div>>> 개발 Blog</div>
</div>
<button onClick={ 제목바꾸기 } >버튼</button>
<div className="list">
```

onClick 안에 함수를 넣을 땐 소괄호는 빼고 넣어야한다. 소괄호를 넣게 되면 함수를 클릭을 하지 않아도 바로 실행하게 된다.

The screenshot shows a React application titled "개발 Blog" running in a browser at localhost:3000. The application displays a list of posts:

- 여자코드 추천 0
  - 2월 17일 발행
- 강남 우동맛집
  - 2월 18일 발행
- 파이썬독학
  - 2월 19일 발행

The code in the VS Code editor (App.js) is as follows:

```
import './App.css';

function App() {
  let [글제목, 글제목변경] = useState(['남자코드 추천', '강남 우동맛집', '파이썬독학']);
  let [따봉, 따봉변경] = useState(0);
  let posts = '강남 고기 맛집';

  function 제목바꾸기(){
    글제목변경( ['여자코드 추천', '강남 우동맛집', '파이썬독학'] );
  }

  return (
    <div className="App">
      <div className="black-nav">
        <div>개발 Blog</div>
      </div>
      <button onClick={ 제목바꾸기 } >버튼</button>
      <div className="list">
        <h3> { 글제목[0] } <span onClick={ ()=>{ 따봉변경(따봉 + 1) } }>👍</span>
        {따봉} </h3>
        <p>2월 17일 발행</p>
        <hr/>
      </div>
      <div className="list">
```

A callout box on the right side of the code editor contains the text: "Q. 버튼을 누르면 첫째 제목이 '여자코드 추천'" (Q. When you click the button, the first title becomes '여자코드 추천').

```

import './App.css';

function App() {
  let [글제목, 글제목변경] = useState(['남자코트 추천', '강남 우동맛집', '파이썬독학']);
  let [파봉, 파봉변경] = useState(0);
  let posts = '강남 고기 맛집';

  function 제목바꾸기(){
    var newArray = [...글제목];
    newArray[0] = '여자코트 추천';
    글제목변경( newArray );
  }

  return (
    <div className="App">
      <div className="black-nav">
        <div>개발 Blog</div>
      </div>
      <button onClick={ 제목바꾸기 }>버튼</button>
      <div className="list">
        <h3> { 글제목[0] } <span onClick={ ()=>{ 파봉변경(파봉 + 1) } }>👍</span>
        {파봉} </h3>
        <p>2월 17일 발행</p>
        <hr/>
      </div>
    </div>
  );
}

export default App;

```

설정된 [데이터]를 만듭니다  
근데 state를 deep copy해서  
수정하세요

state 직접 건들 ↵ ↵ 변경함수 사용

```
글제목[0] = 123123;
```

... (스프레이드 오퍼레이터) 를 사용하여 state의 카피본을 하나 만들어서 변경함수를 이용 해서 변경해야함 즉, 레퍼런스 타입이 아닌 별개의 어레이를 만든다.

카피본 없이 그냥 같다고 해버리면 값 공유(Reference data type) 참조형자료 가 일어나서 오류가 뜸

---

리액트 대 원칙: 모든 state 변수는 immutable data 여야 한다. 즉, 직접 수정이 되면 안된다.

---

변경함수내 인자는 useState 인자 타입과 같아야 함

## 7) React Component : 많은 div들을 한 단어로 줄이고 싶은 충동이 들 때

원래 페이지 구분은 라우터를 사용해야하는데 그건 나중에 배울 것

---

제목을 누르면 상세페이지를 띄우는 실습을 진행(모달창)

return 안에 div 여러개 불가능 즉, 하나의 태그만 생성 가능

자식으로 묶어서 여러개로 만드는 것은 가능

---

즉, html 덩어리를 한 단어로 줄여서 쓸 수 있다.

html 보기 싫은 경우 ⇒ 리액트의 컴포넌트로 묶자.

---

컴포넌트는 App 함수 밖에 또 다른 함수로 만들어줘야 한다.

---

컴포넌트에서 div 대신 묶는 용으로 쓰는 태그는 <> </>

컴포넌트를 만들면 관리가 쉬워짐 하지만 너무 많아도 관리가 힘드니 이럴 땐 redux 를 이용

---

어떤걸 컴포넌트로 만드는게 좋을까?

- ⇒ 반복적으로 출현하는 HTML 덩어리들
  - ⇒ 자주 변경되는 HTML UI들
  - ⇒ 다른 페이지를 만들 때도 컴포넌트로 만듦
- 

컴포넌트를 많이 만들면 단점:

- ⇒ state 쓸 때 복잡해짐
  - ⇒ 다른 함수에서 쓰인 state는 지역변수로써 사용 x
  - ⇒ 이럴경우 props 라는 문법을 사용하여 상속받아서 써야한다.
- 

## 8) 클릭하면 동작하는 UI(모달창) 만드는 법

컴포넌트는 일종의 자바스크립트 표현식과 같다. 그러므로 온갖 곳에 집어넣을 수 있다.

ex) if 문 안, 반복문 등

---

JSX 안에서 if 문은 못쓴다. if 대신 삼항연산자를 사용해야한다. (조건) ? true : false

```
if 대신 삼항연산자

<div>
  <div className="list">
    <h3> { 글제목[2] } </h3>
    <p>2월 19일 발행</p>
    <hr/>
  </div>

  {
    1 < 3 ? console.log('맞아요') : console.log('틀려요')
  }
</div>
}

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

JSX에서는 변수명 넣듯이 중괄호 {} 안에 써야한다.

```
if 대신 삼항연산자

<div>
  <div className="list">
    <h3> { 글제목[2] } </h3>
    <p>2월 19일 발행</p>
    <hr/>
  </div>

  {
    1 < 3
    ? console.log('맞아요')
    : console.log('틀려요')
  }
</div>
}

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

이런식으로 많이 쓴다.

관습같다. 아무것도 없는 것을 표현할 땐 null 사용

```

  <p>제목</p>
  <p>날짜</p>
  <p>상세내용</p>

  {
    1 < 3
    ? <Modal></Modal>
    : null
  }
}

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

```

◀ 텅빈 HTML이라는 뜻

리액트에선 UI를 만들 때 state데이터를 이용한다. UI 가 보임/안보임 정보를 state로 저장해둔다.

```

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 [파봉, 파봉변경] = useState(0);

  let [modal, modal변경] = useState(false); 모달창을 켜고 닫는 스위치

  let posts = '강남 고기 맛집';

```

리액트에선 UI를 만들 때  
state 데이터를 이용합니다

## 9) map : 많은 div들을 반복문으로 줄이고 싶은 충동이 들 때 { map( ) }

반복문 쓰는 법

중괄호를 사용하여 넣자.

중괄호는 항상 변수 or 함수만 가능

for 은 못 쓰는 대신 .map() 함수를 사용

어레이의 데이터들을 모두 2씩 곱해서 새로운 어레이 만들기

```
let [글제목, 글제목변경] = useState(['남자코트 추천', '강남 우동맛집', '파이썬독학']);
let [따봉, 따봉변경] = useState(0);

let [modal, modal변경] = useState(false);

var 어레이 = [2,3,4];
💡 어레이.map(function(a){
... return a * 2
});           I

이 자리에 [4,6,8] 남음

let posts = '강남 고기 맛집';
```

array 내의 모든 데이터에  
똑같은 작업을 시켜주고 싶을 때  
.map()

map 함수 안에 콜백함수를 쓴다. a 는 array 하나하나의 데이터를 의미한다.

```
let [글제목, 글제목변경] = useState(['남자코트 추천', '강남 우동맛집', '파이썬독학']);
let [따봉, 따봉변경] = useState(0);

let [modal, modal변경] = useState(false);
var 어레이 = [2,3,4];
var 뉴어레이 = 어레이.map(function(a){
  return a * 2
});

let posts = '강남 고기 맛집';
```

array 내의 모든 데이터에  
똑같은 작업을 시켜주고 싶을 때  
.map()

새로운 어레이를 정의해준다.

.map() 은 유사 반복문이다.

따봉을 누를때마다 3개가 동시에 변경되는 이유:

⇒ 따봉state 를 한개의 숫자인데 그걸 모두 공유하고 있기 때문이다.

---

for 반복문을 쓰고 싶다면?

return 밖에 함수를 하나 만든다.

```
// 반복문 for 쓰는 법
function 반복된UI() {
  let 어레이 = [];

  for (let i = 0; i < 3; i++) {
    어레이.push(<div>안녕</div>)
  }

  return 어레이
}
```

```
<h3>이 를 </span onClick={()=> 모달모달(모달+1)}>클릭</span>
  {따봉} </h3>
  <p>9월 4일 발행</p>
  <hr/>
</div>
)

}

{
  반복된UI()
}

{
  modal === true
  ? <Modal></Modal>
  : null
}
<button onClick={ 모달스위치 }>버튼</button>
</div>
);
}
```

만든 함수를 return 안에서 바로 실행해준다.

for in / for of 도 똑같이 사용이 가능하다.

