React

프론트엔드 7주차

김보성, 서동섭

리액트 설치

1. Node.js설치
- https://nodejs.org/en/

2. 설치 확인
- 터미널에서 nom -v, node -v

3. 프로젝트 생성
- npx create-react-app 프로젝트 이름

💡 에러가 났을 경우!

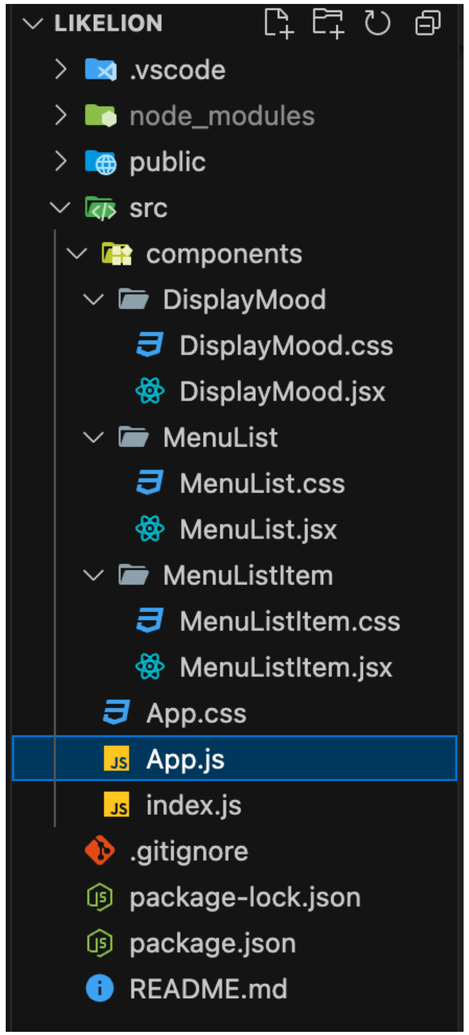
1. 아래 명령어로 실행하세요.
    - npx create-react-app@latest my-app
2. 아래 처럼 에러가 났을 경우에는 `npm uninstall -g create-react-app` 을 해주시면 됩니다.

```
You are running `create-react-app` 5.0.0, which is behind the latest release (5.0.1).

We no longer support global installation of Create React App.

Please remove any global installs with one of the following commands:
- npm uninstall -g create-react-app
- yarn global remove create-react-app

The latest instructions for creating a new app can be found here:
https://create-react-app.dev/docs/getting-started/

PS C:\Users\cheesegom\Desktop\likelion\react\009>
```

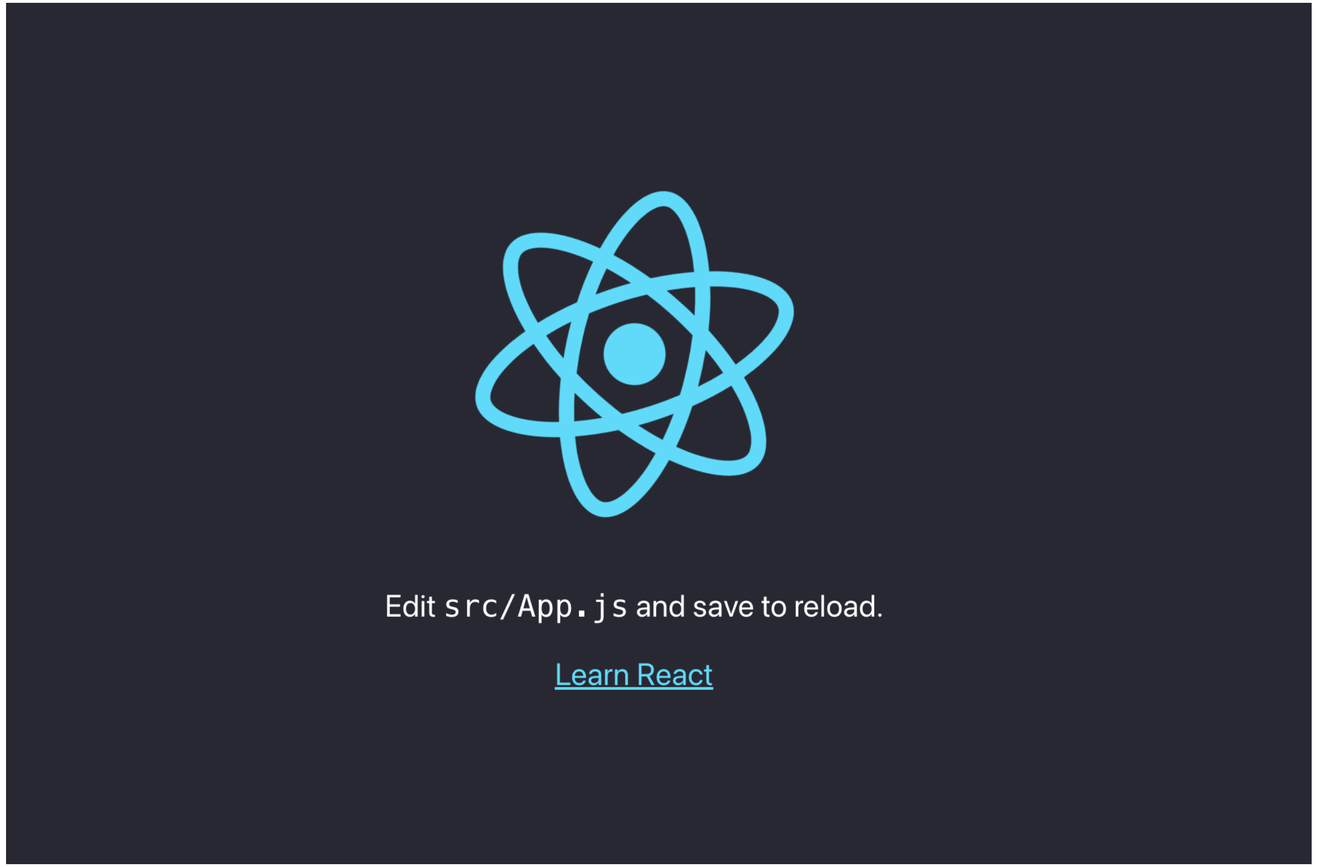
3. 설치가 계속 되지 않을 경우는 아래 명령어를 통해 캐쉬를 비어주세요.

```
npm cache verify
npm cache clear --force
```

프로젝트 확인



터미널 ->
Npm start 입력



리액트 특징

- JavaScript 라이브러리
- 사용자 인터페이스를 구축하는 데 사용
- 효과적인 방식으로 복잡한 UI 구성
- 가상 DOM을 사용하여 빠른 업데이트
- 컴포넌트 기반 아키텍처로 코드 재사용성 높음
- 다양한 라이브러리나 프레임워크 조합 가능

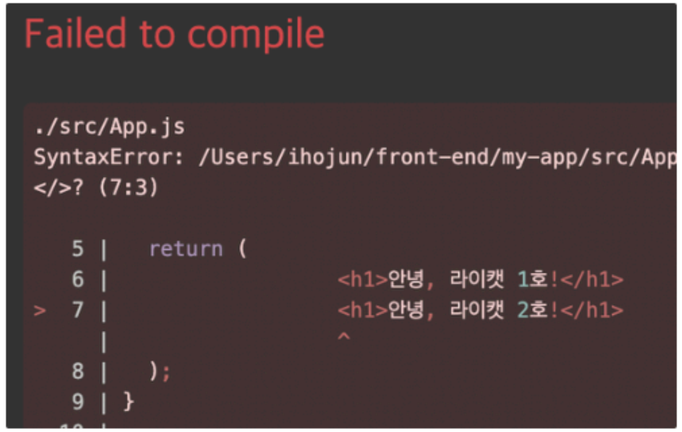
왜 JSX인가?

- JavaScript XML의 약자로 리액트에서 사용하는 특수 문법
- HTML과 매우 유사하지만 다른 언어
- JavaScript에서 HTML을 사용하여 UI를 만드는 것 같은 편리한 개발
- UI 코드의 가독성과 유지보수 높음

JSX 문법

- 컴포넌트의 최상위 부모 요소는 1개 즉 , 단일 해야함

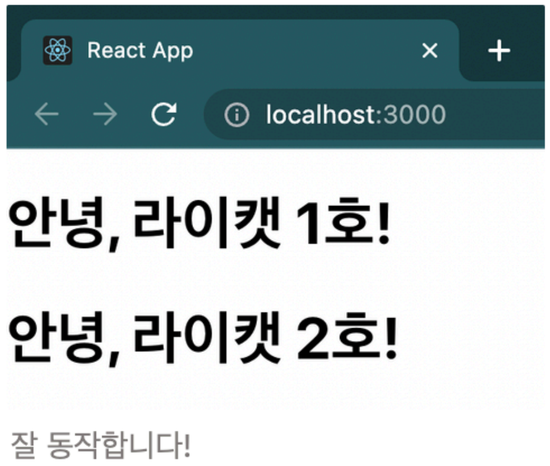
```
function App() {
  return (
      <h1>안녕, 라이캣 1호!
</h1>
      <h1>안녕, 라이캣 2호!
</h1>
  );
}
```



JSX 문법

- 컴포넌트의 최상위 부모 요소는 1개 즉 , 단일 해야함

```
function App() {
  return (
    <div>
      <h1>안녕, 라이캣 1호!</h
      <h1>안녕, 라이캣 2호!</h
    </div>
  );
}
```



- <Fragment> or <> </>

- 의미없는 컨테이너 생성 X
- 실제 DOM에 나타나지 않고 그룹화를 해줌

JSX가 HTML와의 차이점

- 모든 프로퍼티(class, id 등등) 이름은 카멜 케이스 대문자 구분을 따름
  - Ex) onclick -> onClick

- 문자를 제외한 속성값은 {} 사용

- class -> className ->

- 인라인 스타일은 객체 형태로 사용
  - 객체: {key: value}의 형태

- css에서 '-' 사용 x
  - Ex) background - color -> backgroundColor 사용

컴포넌트(components)

- 묶음의 단위 -> 함수랑 유사
- 첫 글자는 대문자(중요)
- 재사용성, 유지보수, 모듈화

** import 방법 **

- import 컴포넌트 이름 from ‘경로’;

1. Export function()
   - import {컴포넌트 이름} from ‘경로’

2. Export default 컴포넌트 이름;
- import 컴포넌트 이름 from ‘경로’;

컴포넌트(components)

App.js M userInfo.jsx U

src > userInfo.jsx > ...

```jsx
import React from 'react';

const UserInfo = () => {
  return (
    <div>
      <h1>John</h1>
      <p>lion@kmu.ac.kr</p>
    </div>
  );
};

export default UserInfo;
```

UserInfo.jsx

App.js M userInfo.jsx U

src > App.js > ...

You, 29 seconds ago | 1 author (You)

```jsx
import React from 'react';
import UserInfo from './userInfo';

function App() {
  return <UserInfo />;
}

export default App;
```

App.js

컴포넌트 흐름

App.js M | userInfo.jsx U

src > App.js > ...

You, 8 seconds ago | 1 author (You)

```jsx
import React from 'react';
import UserInfo from './userInfo';

function App() {
  const user = { name: 'John', email: 'lion@kmu.ac,kr' };

  return <UserInfo user={user} />;
}
export default App;
```

App.js

App.js M | userInfo.jsx U

src > userInfo.jsx > ...

```jsx
import React from 'react';

const UserInfo = ({ user }) => {
  return (
    <div>
      <h1>{user.name}</h1>
      <p>{user.email}</p>
    </div>
  );
};

export default UserInfo;
```

UserInfo.jsx

props

컴포넌트 흐름

App.js M | userInfo.jsx U

src > App.js > ...

You, 1 minute ago | 1 author (You)

```jsx
import React from 'react';
import UserInfo from './userInfo';

function App() {
  const users = [
    { name: 'John', email: 'lion@kmu.ac.kr' },
    { name: 'Jane', email: 'tiger@kmu.ac.kr' },
    { name: 'Doe', email: 'bear@kmu.ac.kr' },
  ];

  return (
    <div>
      {users.map((user, index) => (
        <UserInfo key={index} user={user} />
      ))}
    </div>
  );
}

export default App;
```

App.js

** map(배열명, index) **

- 배열의 크기만큼 반복하여 내부 값들을 사용

- 인덱스 순서 상관없이 자체 함수에서 순서를 차례대로 꺼내서 사용

Props & UseState

App.js M × count.jsx U index.js

src > App.js > App

You, 38 seconds ago | 1 author (You)

```jsx
import React, { useState } from 'react';
import Count from './count';

function App() {
  const [count, setCount] = useState(0);

  return (
    <>
      <Count count={count} />
      <button onClick={() => setCount(count + 1)}>
        Click      You, 38 seconds ago • Uncommitted c
      </button>
    </>
  );
}

export default App;
```

App.js M × count.jsx U × index.js

src > count.jsx > ...

```jsx
import React from 'react';

const Count = ({ count }) => {
  return <div>{count}</div>;
};

export default Count;
```

- ** UseState **

- 변하는 값을 담음

- input, button클릭 등 이벤트 발생시 변화하는 값을 담는데 사용

- -

(코드 예시)

- count : 현재 상태 값

- setCount: 상태를 업데이트 하는 함수

Props & UseState

** props **

- 컴포넌트를 만들 때 넣어줄 수 있는 속성의 집합

```
function App() {
 return (
   <div>
     <Licat name="gary" />
     <Time />
   </div>
 );
}
```

```
function Licat(props) {
  const someStyle = {backgroundColor:"black", color:"white"};
  return(
    <div>
      <h1 style={someStyle}>안녕, {props.name} 1호</h1>
      <h1>안녕, {name} 2호!</h1>
      <div className="newClass"/>
    </div>
  )
}
```

Quiz!!!

안녕, 1호

안녕, 2호!

년 : 2024

월/일 : 6/3

시간 : 16시 35분 57초

❤️즐거운 실습시간 ❤️

- input 활용
- 컴포넌트 2개 생성 후 Import
- Time.jsx, -> props 없음

- Hello.jsx -> input에서 받은 name을 props로

```
<input
    type="text"
    onChange={(e) => setName(e.target.value)}
></input>
```