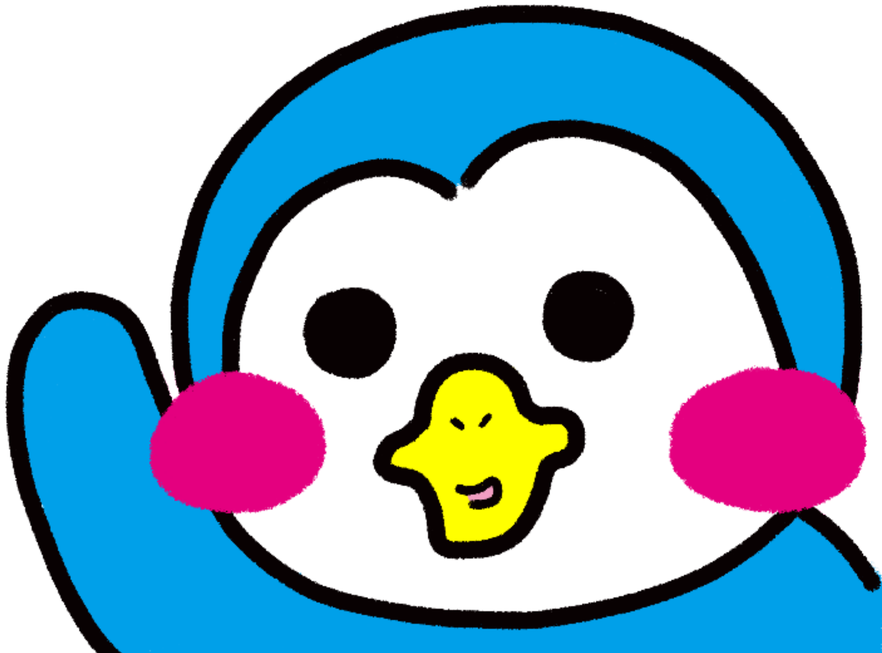
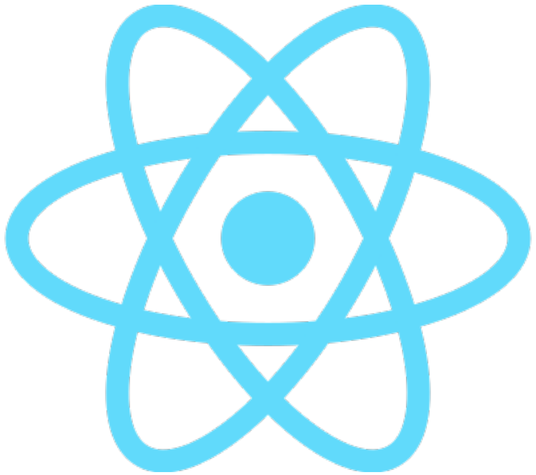
프론트엔드 역량 강화 과정

리액트(React) 소개



리액트를 소개합니다

리액트는 ‘사용자 인터페이스를 만들기 위한 자바스크립트 라이브러리’이다.
사용자 인터페이스(User Interface, UI)란 사용자와 컴퓨터 프로그램이 상호작용을 할 수 있도록 중간에서 입력과 출력의 중개자 역할을 하는 접점을 뜻하며, 리액트는 현존하는 UI 라이브러리 중에서 가장 많이 사용되고 있는 인기 라이브러리이다.



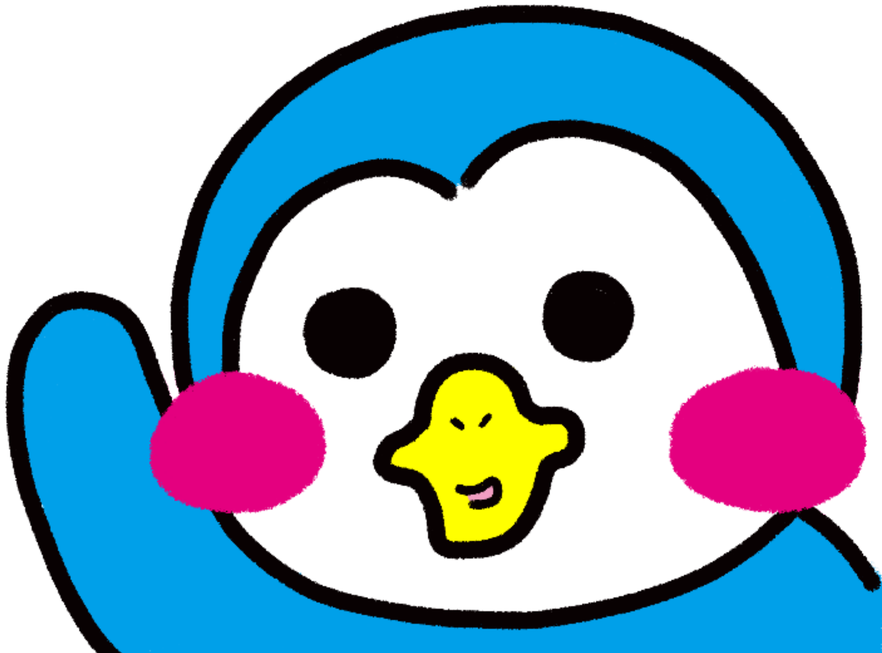
리액트의 특징

리액트는 SPA(Single Page Application) 방식의 개발을 지원하는 라이브러리이다. SPA란 하나의 페이지만 존재하는 웹사이트(웹앱)를 의미하며, 리액트는 가상 DOM을 통한 페이지 업데이트를 처리 함으로써 웹사이트 로딩 시간을 줄여 준다는 장점을 제공한다. 이 외에도 리액트는 다음과 같은 여러 이점들을 제공한다.

- 레이아웃 구성에 편의를 제공하는 컴포넌트 기반 구조
- 코드 작성 시간을 줄여 주는 재사용성
- 거대한 커뮤니티와 다양한 관련 라이브러리
- 모바일 앱 개발 프레임워크로의 영역 확장

프론트엔드 역량 강화 과정

create-react-app



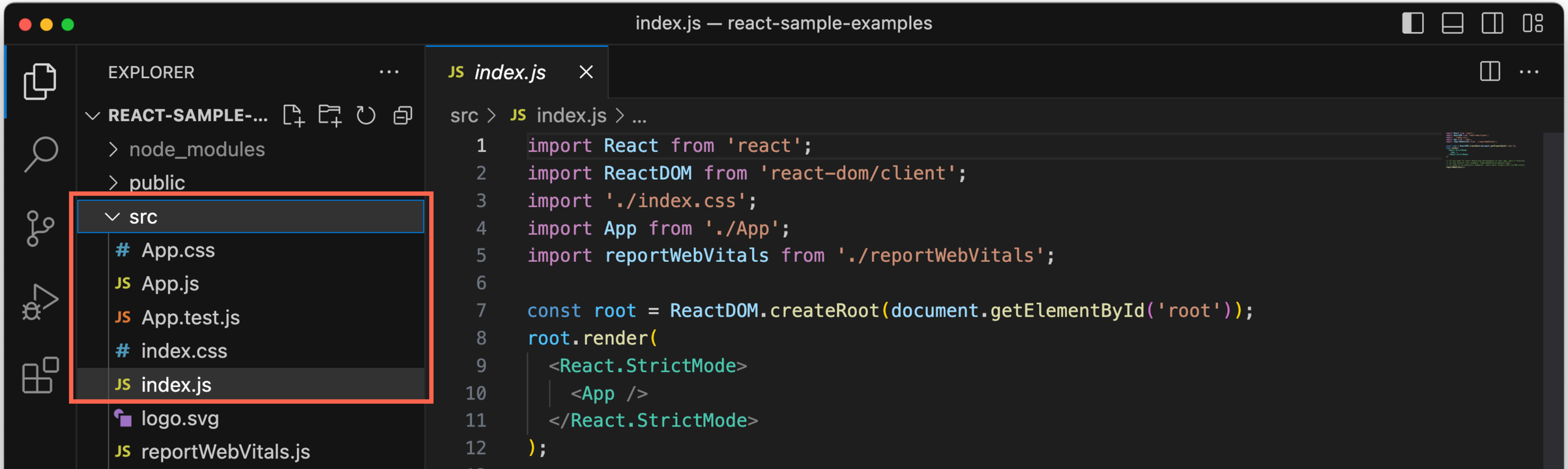
create-react-app, 줄여서 CRA

기존 웹사이트에 리액트를 적용하는 방식으로 개발을 진행할 수도 있지만, 일반적으로 리액트를 이용해 새로운 웹사이트를 만들 때는 그런 방식을 사용하지 않는다.
create-react-app(CRA)은 리액트로 웹 애플리케이션을 개발하는 데 필요한 모든 설정이 적용된 프로젝트를 생성해주는 개발 도구로, 대부분의 리액트 웹 개발은 이를 통해 진행된다.
CRA는 노드(node.js) 기반으로 동작한다. 따라서 CRA 사용을 위해서는 node.js 설치가 선행되어야 한다. CRA를 이용한 프로젝트 생성은 다음 명령어를 통해 진행한다.

```
npx create-react-app 프로젝트명
```

CRA 프로젝트 수정 방법

CRA 프로젝트에서 가장 중요한 것은 src 폴더이다. src 폴더에 있는 index.js 파일은 리액트 애플리케이션이 실행될 때 가장 먼저 실행되는 진입점(entry point) 역할을 수행한다. 리액트 애플리케이션 실행 명령어는 `npm start` 이다.



CRA 프로젝트 수정 방법

index.js 파일의 중간 부분 <App /> 태그는 UI 전체를 포함하는 태그이므로, 이를 삭제하면 웹 브라우저 화면에 아무것도 나타나지 않는다.

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

2회차 예제, src/App01.js

CRA 프로젝트 수정 방법

<App />의 실제 내용은 index.js 파일과 같은 디렉터리에 존재하는 App.js 파일에 존재한다. App.js 파일의 내용을 수정하여 <App /> 태그를 렌더링하는 것이 CRA 프로젝트의 기본적인 수정 방법이다.

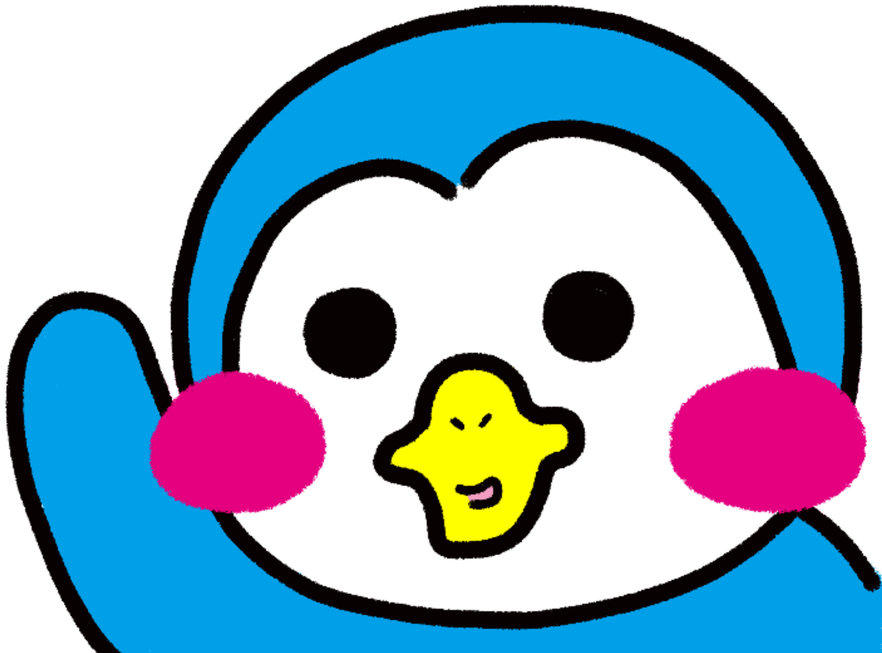
```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
```

리액트 컴포넌트를 수정하기 위해서는 JSX 문법을 이해해야 한다!

프론트엔드 역량 강화 과정

JSX



JSX란?

JSX는 자바스크립트의 확장 문법으로, 자바스크립트와 XML을 결합한 형태이다.
CRA 프로젝트에서 리액트 컴포넌트는 이 문법을 기반으로 만들어지므로, JSX는 리액트 애플리케이션 개발을 위해 반드시 알아야 하는 기술 중 하나이다. JSX는 코드가 간결하고, 마크업 형태가 포함되어 있어 가독성이 뛰어나다.

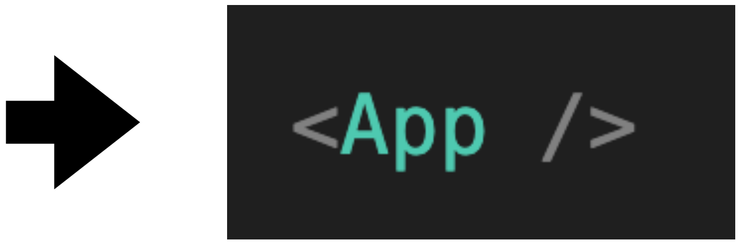
JSX로 작성된 코드는 웹 브라우저가 직접 해석할 수 없지만, CRA 프로젝트는 JSX를 순수(?)한 형태의 자바스크립트 코드로 변환해주는 기능까지 모두 설정되어 있기 때문에 걱정하지 않아도 된다.

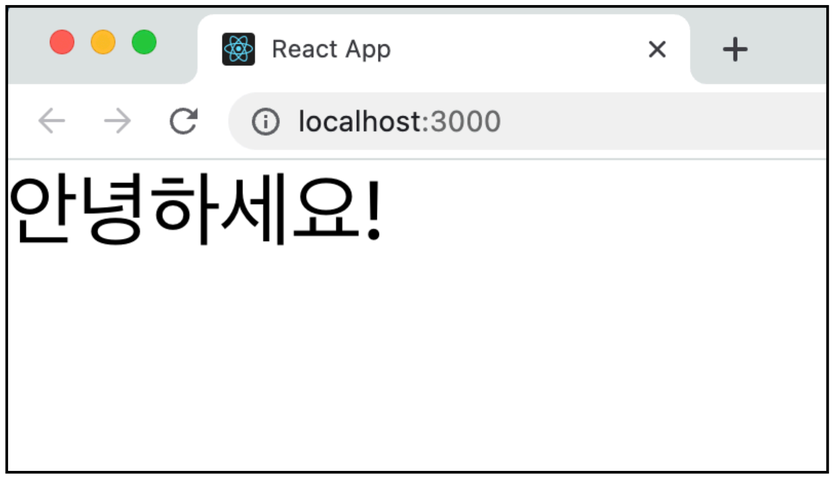
BABEL

JSX 문법

JSX는 자바스크립트 문법을 확장시킨 것이기 때문에, 모든 자바스크립트 문법을 지원한다. 여기에 추가로 마크업을 섞어서 사용하는 형태이다.

```
function App(){
  const element = <div>안녕하세요!</div>
  return element
}
```





2회차 예제, src/App02.js

JSX 문법 규칙

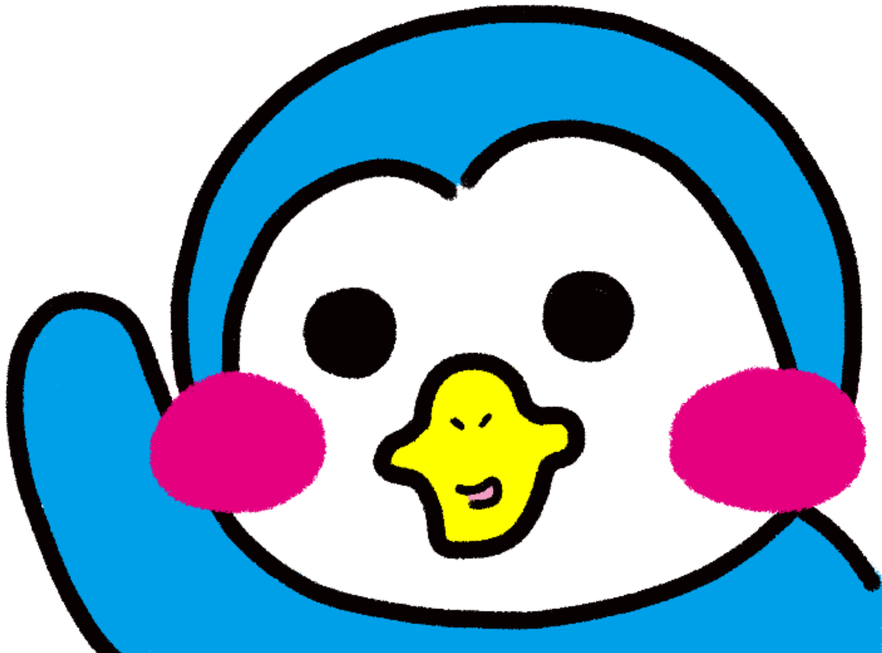
JSX 사용 시에는 다음 규칙들을 반드시 준수해야 한다.

- 태그는 반드시 닫혀야 한다. 단일 태그의 경우에도 닫는 표시는 해주어야 한다.
- 두 개 이상의 태그는 반드시 하나의 태그로 감싸져 있어야 한다.
- JSX 내부에 자바스크립트 변수를 표현할 때는 중괄호로 감싸야 한다.
- style 속성 정의 시에는 CSS 표기법이 아닌 객체 리터럴 표기법을 따른다.
- 주석 입력 시에는 자바스크립트의 주석을 입력하되, 중괄호로 감싸야 한다.
- class 속성은 className 이라는 이름으로 대체해야 한다.

2회차 예제, src/App03.js

프론트엔드 역량 강화 과정

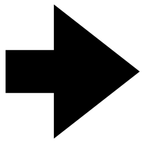
컴포넌트 & props



리액트 컴포넌트

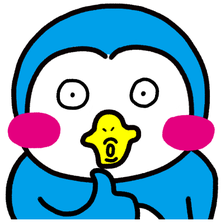
리액트는 사용자가 정의한 UI, 즉 '컴포넌트'를 만드는 기술이다. 리액트 애플리케이션은 버튼, 메뉴 바, 메시지 카드 등의 컴포넌트가 모여 하나의 웹 페이지를 구성하는 방식으로 만들어진다. 쉽게 말해, 리액트는 '사용자 정의 태그'를 만드는 기술인 셈이다!

```
function App(){
  const element = <div>안녕하세요!</div>
  return element
}
```



```
<App />
```

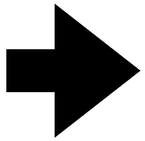
사용자가 App 이라는 이름의 컴포넌트를 만들고, 이를 마크업한 예시.
App 컴포넌트의 내부에는 외에도 다양한 태그 및 컴포넌트를 마크업할 수 있다.
이때 App 컴포넌트는 함수 형태로 작성되었으므로 '함수형 컴포넌트'라 표현할 수 있다.



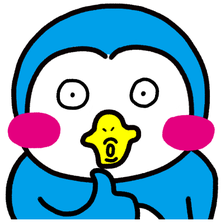
props 활용

한번 만든 컴포넌트는 원하는 만큼 재사용 가능하다. 그러나 같은 컴포넌트라 할지라도, 마크업 위치나 상황에 따라 세부 요소(색상, 크기 등)를 달리 표현하고 싶을 때가 있다. 그럴 때는 컴포넌트 마크업 시 부가적인 속성을 정의하고 활용하는 방식을 사용하면 되는데, 이때 속성을 가리켜 props라 한다.

```
function MyComponent(props){
  return <div>
    {props.value}
  </div>
}
```



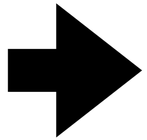
```
<MyComponent value="속성값" />
```



props 활용

컴포넌트 마크업 시 속성값으로 함수를 전달할 수도 있다.
이는 이벤트 핸들링에 도움이 되는 기능이다.

```
function MyComponent(props){
  return <div onClick={props.func}>
    나를 클릭해 봐
  </div>
}
```



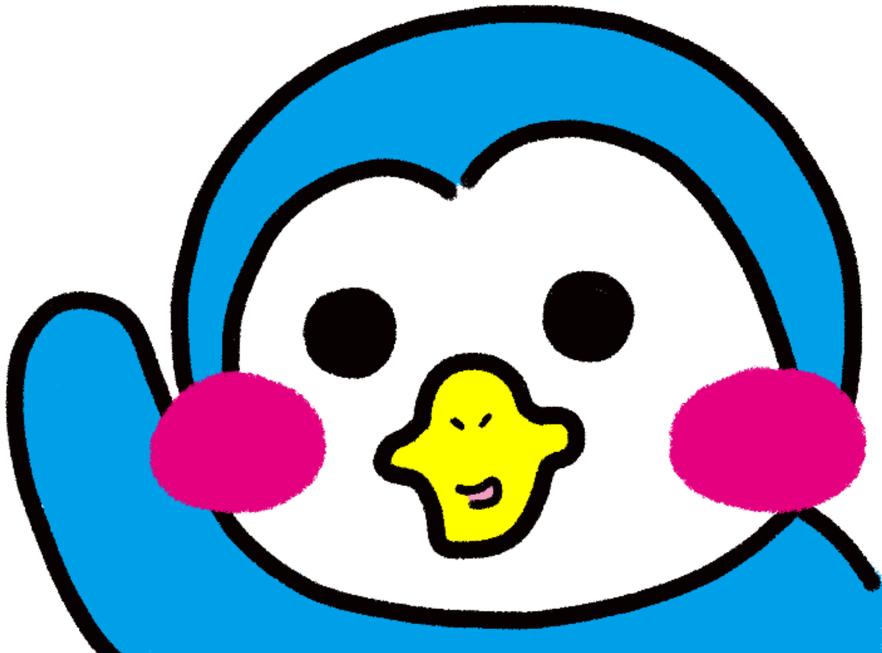
```
<MyComponent func={() => {
  alert("!")
}} />
```

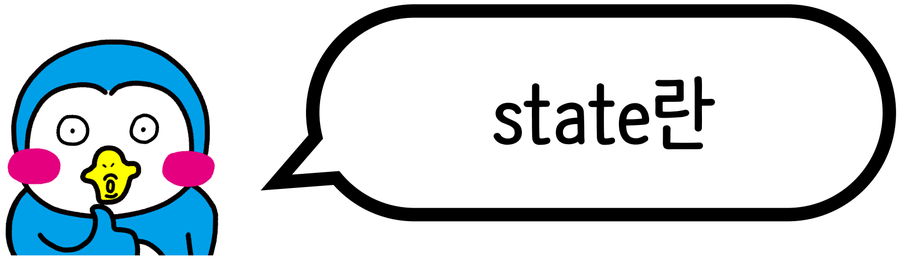
클릭하면 느낌표(!)가 쓰여진 경고창이 나타난다

2회차 예제, src/App04.js

프론트엔드 역량 강화 과정

state



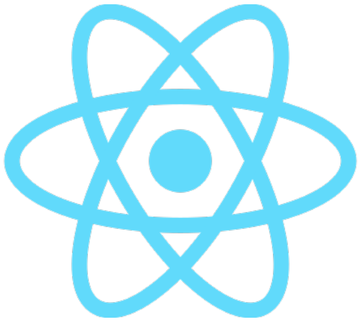


리액트에서 state란 리액트 컴포넌트의 상태를 뜻하는 용어이다. 정확히 표현하자면, state는 리액트 컴포넌트가 보유하고 있는 데이터라고 할 수 있다. state는 변경 가능한 데이터이며, state가 변경될 경우 컴포넌트가 다시 렌더링된다. 따라서 state는 렌더링이나 데이터 흐름에 사용되는 값을 관리하기 위한 도구라고 볼 수 있다. state는 사전에 정의된 것이 아니므로, 개발자가 직접 정의해서 사용해야 한다.

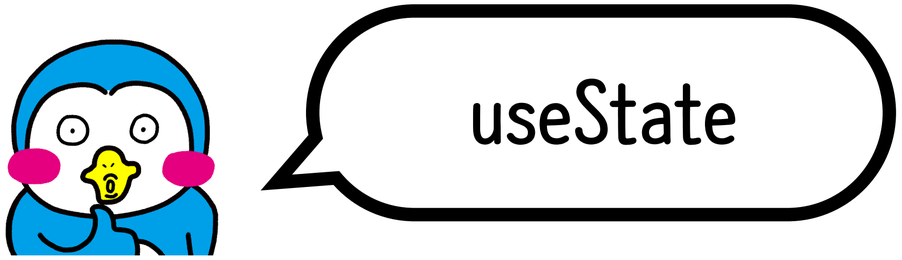
함수형 컴포넌트가 state를 가지도록 하기 위해서는, 리액트 라이브러리에 존재하는 useState 함수를 사용해야 한다!

리액트 훅(hooks)

리액트 컴포넌트에는 두 가지 종류가 있다. 하나는 클래스 컴포넌트이고, 다른 하나는 함수 컴포넌트이다. 과거에는 클래스 컴포넌트만이 state 관리를 지원하는 형태였기 때문에 클래스 컴포넌트가 주로 사용되었으나, 함수 컴포넌트에서 state 관리 등과 같은 기능을 지원하는 '리액트 훅'이 등장한 뒤로는 함수 컴포넌트가 주로 사용되고 있다. 리액트 훅은 리액트 컴포넌트의 기능을 지원하는 함수 모음의 공식 명칭이다.



앞서 언급한 바 있는 useState 함수가 바로 대표적인 리액트 훅 중 하나이다.
함수 컴포넌트는 스스로 state 관리를 수행할 수 없기 때문에
상태 관리를 위해서는 반드시 useState 함수를 사용해야 한다.




```
const [상태명, 상태변경함수] = useState(초깃값)
```

상태명은 state에 붙은 이름이다. 즉 상태를 나타내는 변수이다. 그러나 여기에 대한 직접적인 변경은 불가하다. state 변경을 위해서는 상태 변경 함수에 state의 새로운 값을 인자로 전달해주어야 한다.

```
let [count, setCount] = useState(0)

return <h1 onClick={() => setCount(1) }>
  {count}
</h1>
```

0으로 초기화 되었던 상태 count가,
<h1> 요소를 클릭하면 1이 된다!

2회차 예제, src/App05.js

2회차 예제, src/App06.js