



React, Sass 실습을 통한

모던 프론트엔드 시작하기

멋쟁이 사자처럼 삼육대 - 프론트엔드 심화세션

유경수

```
<세션 주차="2">
```

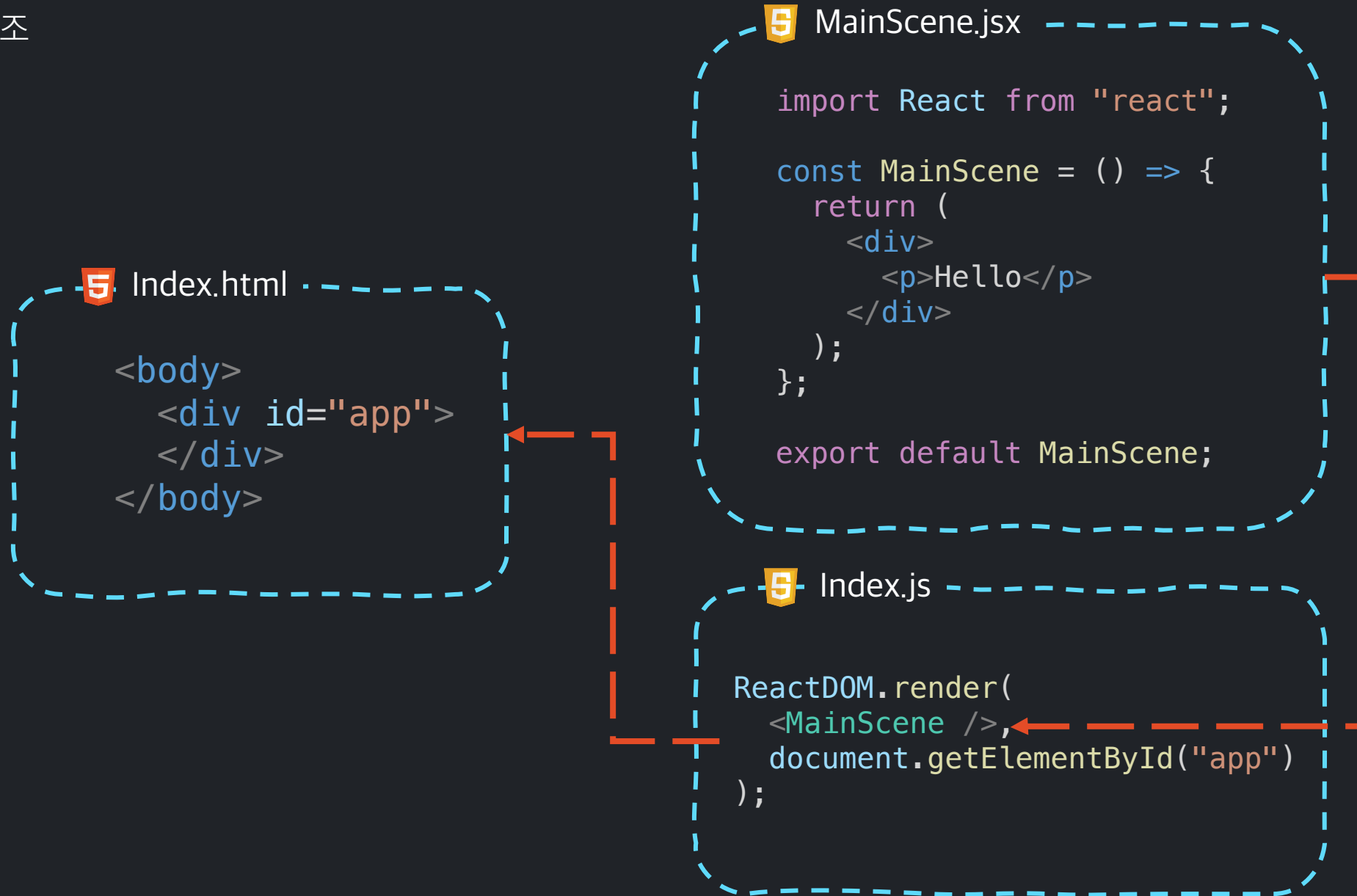
```
  <목차 내용="REACT 개발 환경 만들어보기" />
```

```
  <목차 내용="Class? Hooks?" />
```

```
  <목차 내용={ [ 'state' , '실습' , <Counter/> ] } />
```

```
<세션/>
```

React의 구조

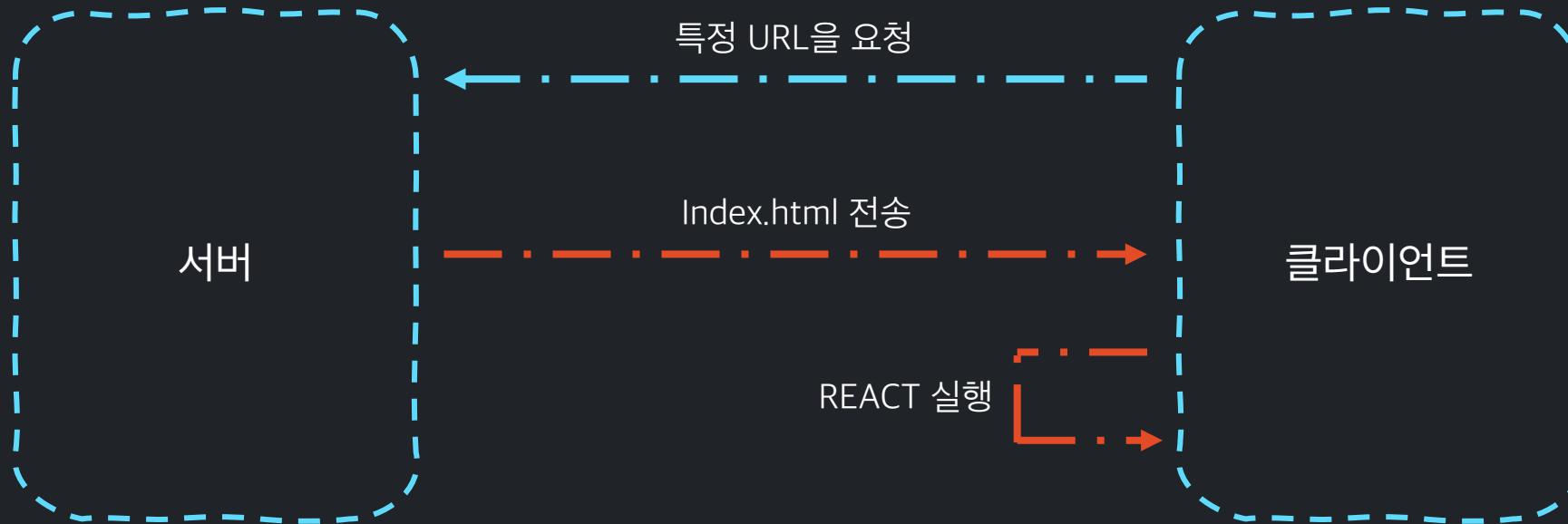


React의 구조

 Index.html

```
<body>
  <div id="app">
    <div>
      <p>Hello</p>
    </div>
  </div>
</body>
```

React의 실행 순서



React 컴포넌트의 특이사항

 MainScene.jsx

```
import React from "react";

const MainScene = () => {
  return (
    <div>
      <p>Hello</p>
    </div>
  );
};

export default MainScene;
```

일반 브라우저에서 지원하지 않음!

서로 의존성 주입이 수행되지 않음!

React 컴포넌트의 특이사항 해결하기

BABEL

일반 브라우저에서 지원하지 않음!

-> 새로운 문법의 내용을 브라우저에서
지원하도록 변환



서로 의존성 주입이 수행되지 않음!


-> 흩어진 파일들을 모아서 하나의 JS파일로
변환

React 컴포넌트의 특이사항 해결하기



프로젝트 폴더 구성 Preview


✓  public


 index.html

✓  src


✓  components

 App.jsx

 index.js

 .babelrc

 webpack.config.js

 package.json

React 컴포넌트가 생성될 html

화면에 보여줄 컴포넌트

컴포넌트를 로드 해, index.html 에 삽입해주는 코드

코드를 변환해줄 babel 설정

코드를 번들링해줄 webpack 설정

필요한 패키지와 실행 명령을 저장해줄 파일

개발 환경 세팅 - npm

```
npm init
```

```
-- npm의 package.json 초기화하기
```

```
npm i react react-dom
```

```
-- react / react-dom 설치하기
```

```
npm i @babel/core @babel/preset-env @babel/preset-react
```

```
-- 구문 변환을 위한 babel 설정 설치하기
```

```
npm i babel-loader html-webpack-plugin path webpack webpack-cli webpack-dev-server
```

```
-- webpack관련 항목 설치하기
```

개발 환경 세팅 - babel

BABEL .babelrc

```
{  
  "presets": ["@babel/preset-env", "@babel/preset-react"]  
}
```

개발 환경 세팅 - webpack




webpack.config.js


```
const path = require("path");
const HtmlWebpackPlugin = require("html-webpack-plugin");

module.exports = {
  entry: "./src/index.js",
  output: {
    path: path.resolve(__dirname, "build"),
    filename: "main.js",
  },
  devServer: {
    port: 3000,
    open: true,
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /node_modules/,
        use: {
          loader: "babel-loader",
        },
      },
    ],
  },
  plugins: [
    new HtmlWebpackPlugin({
      template: path.resolve(__dirname, "public/index.html"),
      filename: "index.html",
    }),
  ],
};
```

개발 환경 세팅 - component

 /public/index.html


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Document</title>
  </head>
  <body>
    <div id="app"></div>
  </body>
</html>
```

 /src/components/App.jsx

```
import React from "react";

const App = () => {
  return (
    <div>
      <p>Hello Your First React App</p>
    </div>
  );
};

export default App;
```

 /src/index.js

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App.jsx";

ReactDOM.render(<App />,
  document.getElementById("app"));
```

개발 환경 세팅 - npm 스크립트 작성

```
npm package.json
```

```
...  
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1",  
  "build": "webpack --mode=production",  
  "start": "webpack-dev-server --mode=development"  
},
```

개발 환경 실행

```
> npm run build
```

Build 폴더에 번들링된 파일이 생성됨
완성된 리액트 프로젝트를 배포할 때 사용

```
> npm run start
```

React가 적용된 html을 위한 서버를 실행
관련 파일들이 변경될 때 마다,브라우저에서 자동으로 새로고침

Class로 구성한 컴포넌트

```
import React from "react";
```

```
// 설치된 react 모듈을 불러옵니다
```

```
class App extends React.Component {  
  render() {  
    return (  
      <div>  
        <p>First React App</p>  
      </div>  
    );  
  }  
}
```

```
// React.Component를 상속 받습니다
```

```
// React.Component의 render 함수를 Override  
// 합니다
```

```
// HTML 요소를 반환합니다
```

```
export default App;
```

```
// React.Component를 상속받은 App 을  
// 내보냅니다
```


Class와 상속

```
class Animal {  
  constructor(name) {  
    this.name = name;  
  }  
  
  speak() {  
    console.log(`${this.name} makes a noise.`);  
  }  
}  
  
let animal = new Animal("Elephant");  
animal.speak(); // Elephant makes a noise.
```

Class와 상속

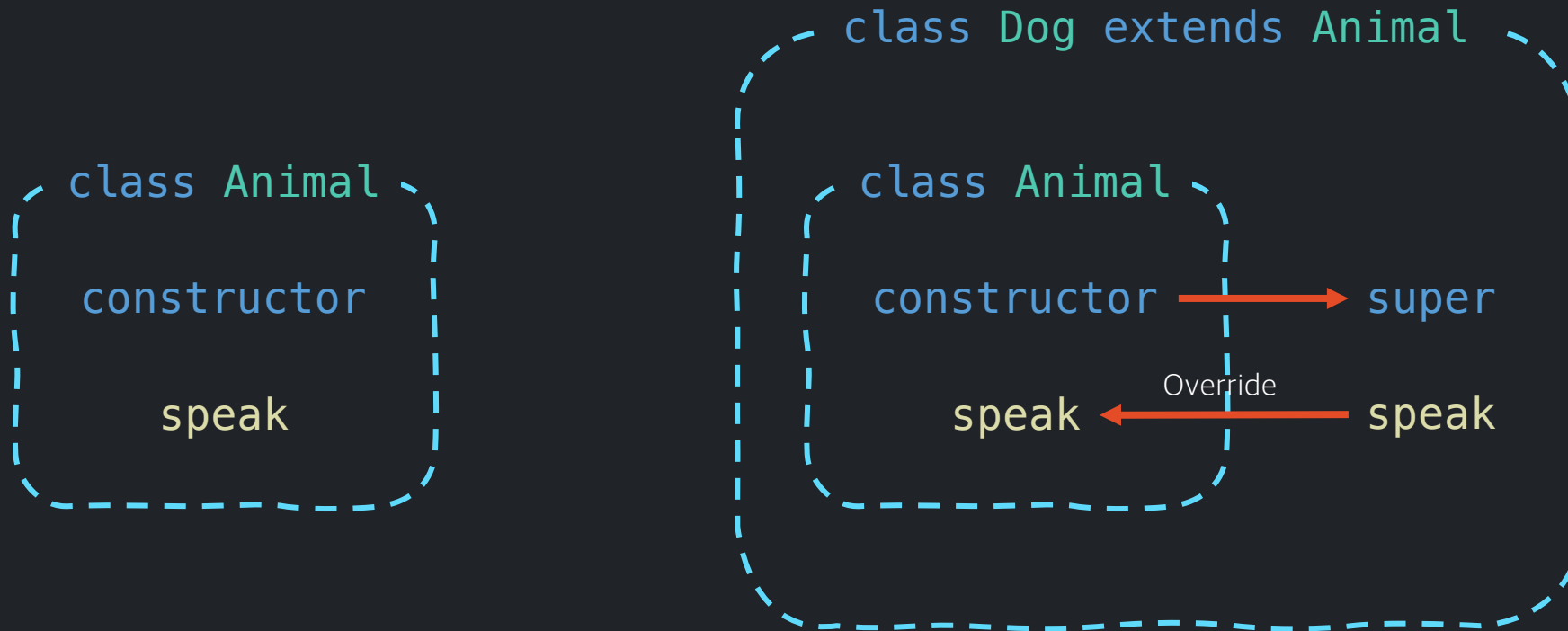
```
class Animal {  
  constructor(name) {  
    this.name = name;  
  }  
  
  speak() {  
    console.log(`${this.name} makes a noise.`);  
  }  
}
```

```
let animal = new Animal("Elephant");  
animal.speak(); // Elephant makes a noise.
```

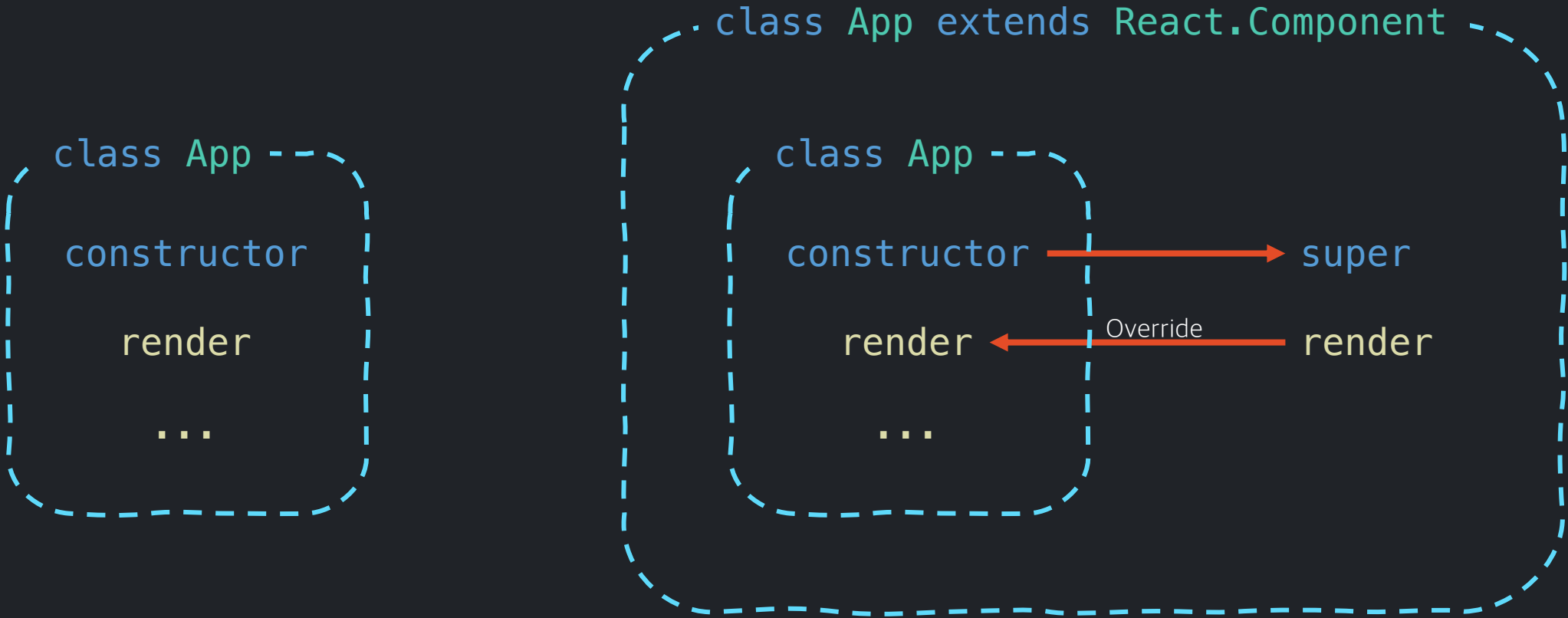
```
class Dog extends Animal {  
  constructor(name) {  
    // name을 파라미터로 부모 클래스의 생성자를 호출  
    super(name);  
  }  
  
  speak() {  
    console.log(`${this.name} barks.`);  
  }  
}
```

```
let d = new Dog("Mitzie");  
d.speak(); // Mitzie barks.
```

Class와 상속



Class와 상속



Class와 상속

```
class React.Component
```

```
  constructor
```

```
  render
```

```
  componentDidMount
```

```
  componentDidUpdate
```

```
  componentWillUnmount
```

```
  ...
```

Hook

```
import React from "react";

class App extends React.Component {
  render() {
    return (
      <div>
        <p>First React App</p>
      </div>
    );
  }
}

export default App;
```

```
import React from "react";

const App = () => {
  return (
    <div>
      <p>First React App</p>
    </div>
  );
};

export default App;
```

```
import React, { useState, useEffect } from "react";

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

  useEffect(() => {
    console.log(`current count : ${count}`);
  }, [count]);

  return (
    <div>
      <button type='button' onClick={() => setCount(count + 1)}>up</button>
      <button type='button' onClick={() => setCount(count - 1)}>down</button>
      <p>count : {count}</p>
    </div>
  );
};

export default App;
```