



# React

---

00. **React** ??

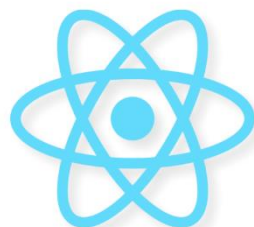
---



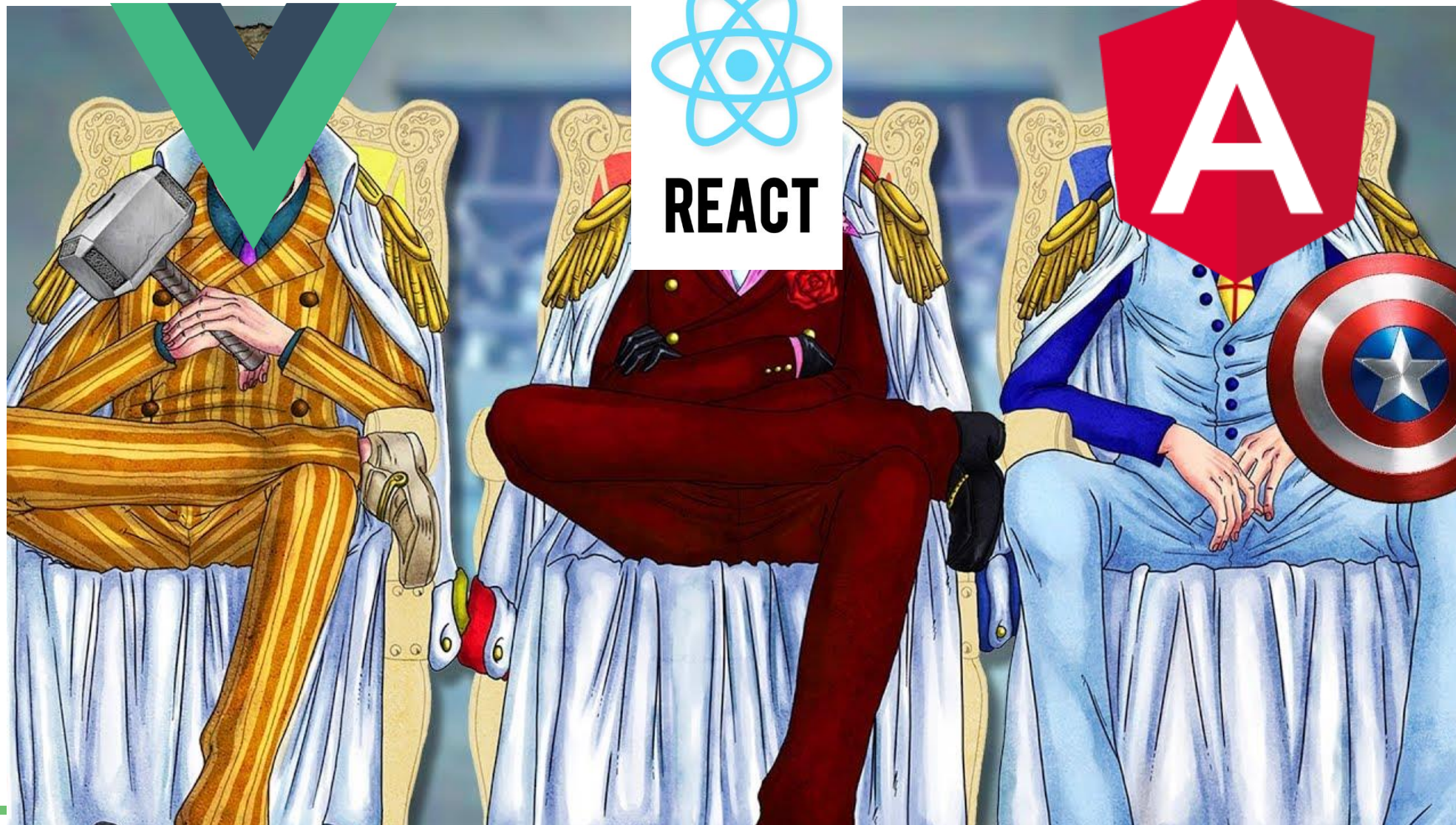
facebook

Google

x CODINGO™



REACT

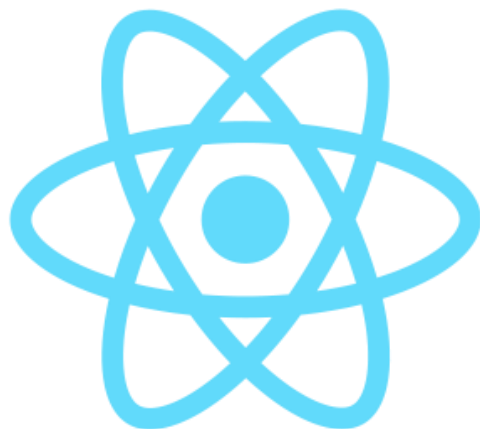


# 프론트엔드 3대장

- 다음 중 가장 많이 쓰이는 것은?



ANGULARJS



React JS



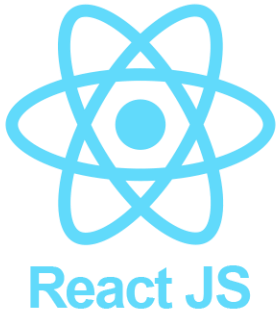
Vue.js

# Angular JS



- 구글에서 만든 JavaScript 기반의 오픈 소스 프레임워크
- 양방향 데이터 바인딩으로 양방향 웹 애플리케이션에 적합
- 2016년도 이후 점유율 하락 중
- Ex) 유튜브, 페이스북, 구글, 텔레그램 등등

# React JS



- 동적 사용자 인터페이스를 만들기 위해 2011년 페이스북에서 만든 오픈 소스 JavaScript 라이브러리
- 데이터 변경이 잦은 복잡하고, 규모가 큰 라이브러리에 적합
- Angular 보다 배우기 쉽다고 이야기 됨.
- Ex) 페이스북, 인스타그램, 넷플릭스, 야후, 드롭박스 등등



# Vue.js

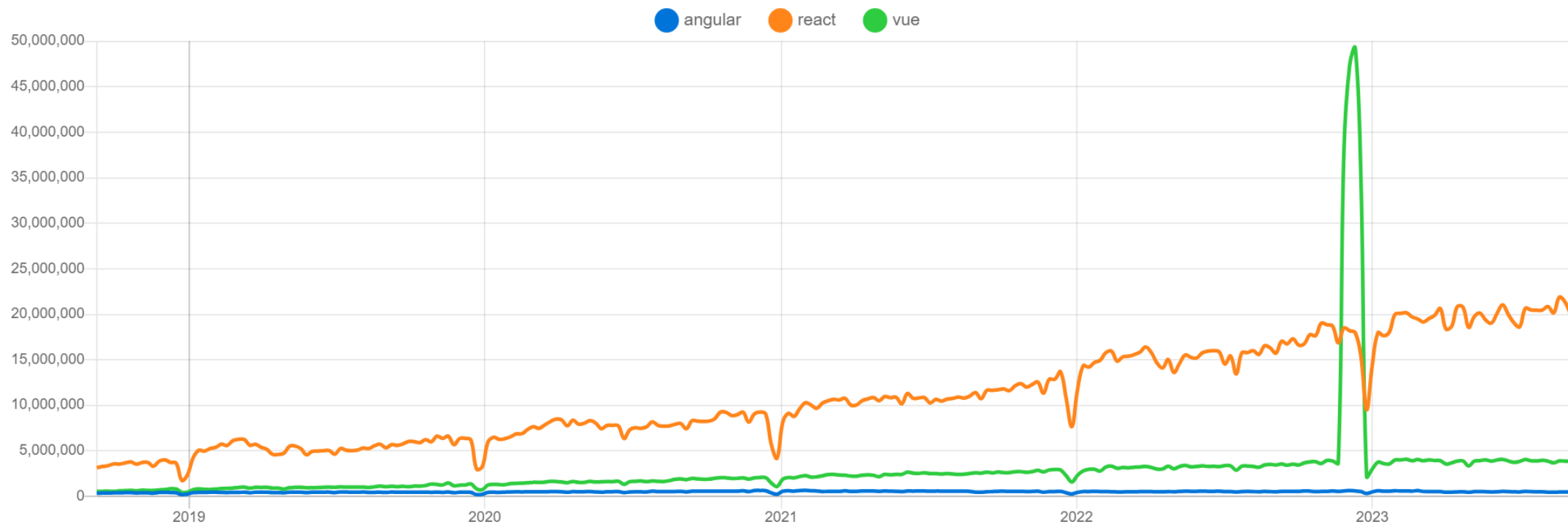


- 2013년 출시된 JavaScript 프레임워크
- Angular와 React의 장점을 수용한 프레임워크
- 중국어 기반으로 Reference가 적음.
- Ex) 샤오미, 알리바바, 깃랩, 어도비 등

# Angular vs. React vs. Vue.js

angular x react x vue x + @angular/core

Downloads in past 5 Years ▾

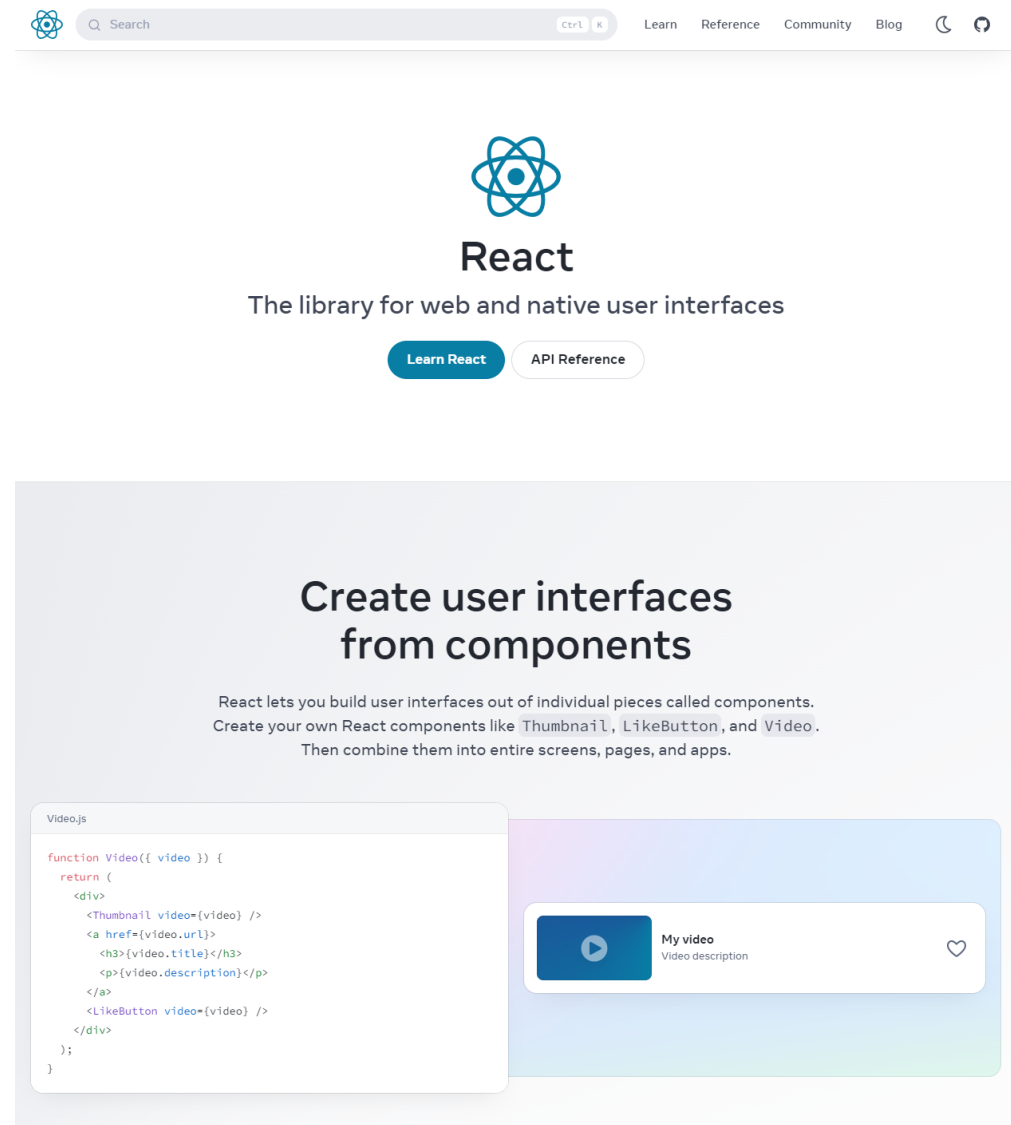


# React 란?



# React란?

- <https://react.dev/>
- The **library** for web and native **user interfaces**



# React란?

- 화면을 만들기 위한 자바스크립트 라이브러리
- 현재 프론트엔드 개발 환경에서 가장 많이 활용되고 있는 라이브러리
- 사용자와 상호 작용이 가능한 동적 UI 제작 가능

# React의 특징

1. Data Flow
2. Component 기반 구조
3. Virtual Dom
4. Props and State
5. JSX

# 특징 1. Data Flow

- 양방향 X **단방향** O 데이터 흐름
- Angular JS 처럼 양방향 데이터 바인딩은 규모가 커질수록 데이터의 흐름을 추적하기 힘들고 복잡해지는 경향이 있다.

## 특징 2. Component 기반 구조

- **Component** : 독립적인 단위의 소프트웨어 모듈로 소프트웨어를 독립적인 **하나의 부품**으로 만드는 방법
- React는 UI(View)를 여러 Component를 쪼개서 만든다.
- 한 페이지 내에서 여러 부분을 Component로 만들고 이를 조립해 화면을 구성

## 특징 2. Component 기반 구조

1. Component 단위로 쪼개져 있기 때문에, 전체 코드를 파악하기 쉽다.
2. 기능 단위, UI 단위로 캡슐화시켜 코드를 관리하기 때문에 **재사용성**이 높다.
3. 코드를 반복할 필요 없이 Component만 import 해서 사용하면 된다는 **간편함**이 있다.
4. 애플리케이션이 복잡해지더라도 **코드의 유지보수, 관리가 용이하다.**

캡슐화란?

데이터와, 데이터를 처리하는 행위를 묶고, 외부에는 그 행위를 보여주지 않는 것

## 특징 2. Component 기반 구조

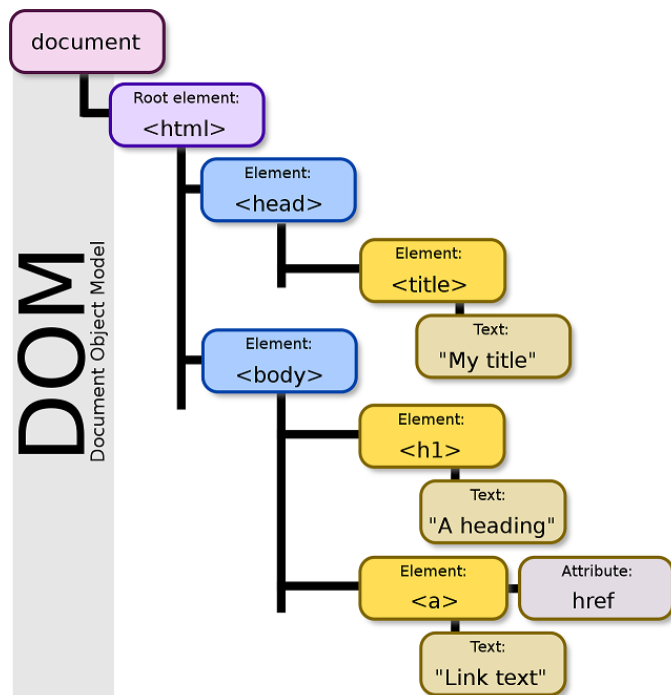
```
class App extends Component {
  render() {
    return (
      <Layout>
        <Header />
        <Navigation />
        <Content>
          <Sidebar></Sidebar>
          <Router />
        </Content>
        <Footer></Footer>
      </Layout>
    );
  }
}
```

- Header, Footer 같은 구조를 컴포넌트로 제작
- 컴포넌트를 조합해 Root Component (최상위 컴포넌트) 로 만들



# 특징 3. Virtual DOM

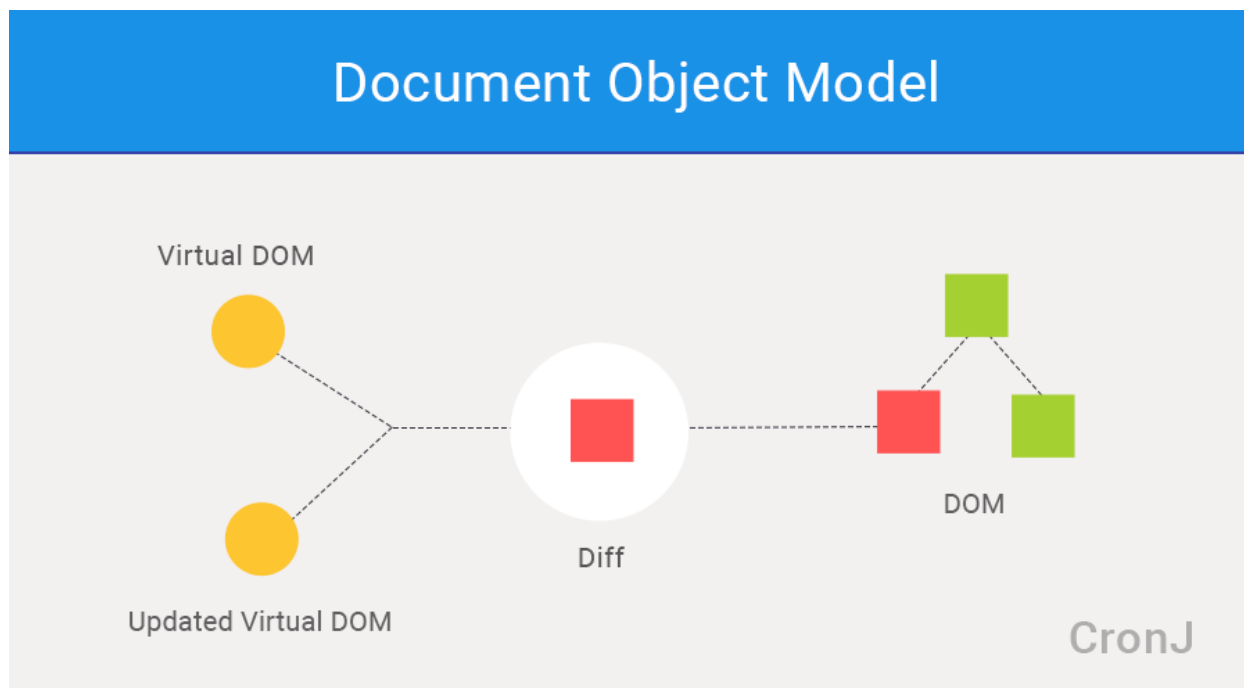
- DOM : Document Object Model ( 문서 객체 모델 )



- React는 DOM Tree 구조와 같은 구조를 Virtual DOM 으로 가지고 있다.
- 이벤트가 발생할 때마다 Virtual DOM을 만들고 다시 그릴 때 실제와 전후 상태를 계속 비교 -> 앱의 효율성과 속도 개선

## 특징 3. Virtual DOM

- 이벤트가 발생할 때마다 Virtual DOM을 만들고 다시 그릴 때 실제와 전후 상태를 계속 비교 -> 앱의 효율성과 속도 개선



# 특징 4. Props and State

- Props
  - 부모 컴포넌트에서 자식 컴포넌트로 전달해주는 데이터
  - 자식에서는 props 변경 불가능, props를 전달한 최상위에서만 변경 가능
- State
  - 컴포넌트 내부에서 선언되고 내부에서 값을 변경
  - 클래스형 컴포넌트에서만 사용 가능, 각각의 state는 독립적 (함수형 컴포넌트에서는 NO!)

# 특징 5. JSX

- React 에서 JSX 사용이 필수는 아니지만 편의성을 위해 대부분의 프로젝트에서 JSX를 사용
- JSX = Javascript + XML
- JavaScript를 확장한 문법으로 React Element를 생성

더 자세한 건 React 프로젝트를 생성한 후에 배워보자!

# Node.js & NPM

# Node.js & NPM

- Node.js는 React를 사용하기 쉽게 하는 도구를 내장하고 있음
- Node.js와 함께 설치되는 NPM(Node Package Manager)를 이용하면 프론트엔드 프로젝트를 위한 다양한 패키지를 쉽게 설치하고 관리 가능
  - 리액트도 하나의 패키지!
- 자바스크립트 런타임인 Node.js 기반으로 만들어진 Webpack과 Babel 같은 도구를 자주 사용

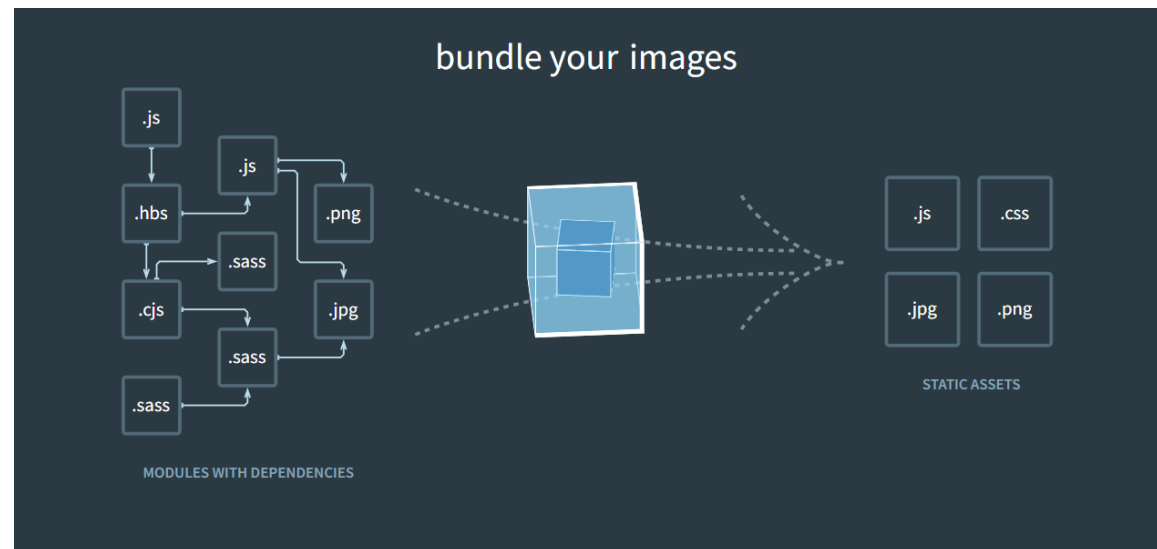
# Babel *BABEL*

- 옛날에는 큰 기능을 하지 않는 자바스크립트 컴파일러(해석기)였지만 리액트에서는 중요한 역할
- 최신 JavaScript 문법을 이전 버전의 JavaScript 문법으로 변환해주어 다양한 브라우저에서 호환성에 문제 없이 코드를 실행할 수 있음



# Webpack webpack

- 자바스크립트 모듈 번들러 (Module Bundler)
- 여러 개의 파일과 모듈을 하나의 파일로 합쳐줌
- 코드 변경사항 발생 시 바로 반영되는 기능



# React 프로젝트 생성

# 방법1. CDN 사용

```
<script crossorigin
src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
```



여러분이 사용하고 있는 코드 편집기를 사용하길 원한다면, [이 HTML 파일을 다운로드](#)하고 편집한 다음 브라우저의 로컬 파일 시스템에서 열 수도 있습니다. 런타임 코드 변환이 느리므로 간단한 데모에만 이 코드를 사용하는 것이 좋습니다.

## 웹사이트에 React를 추가하기

1분 안에 HTML 페이지에 React를 추가할 수 있습니다. 그리고 조금씩 React의 비중을 늘리거나 몇 개의 동적 위젯에 포함할 수 있습니다.

## 새 React 앱 만들기

React 프로젝트를 시작한다면 [스크립트 태그를 사용한 간단한 HTML 페이지](#)를 만드는 것이 최고의 방법일 것입니다. 설치하는 데 1분밖에 걸리지 않습니다!

그러나 애플리케이션이 커진다면 보다 통합된 설정을 고려하는 것이 좋습니다. 대규모 애플리케이션

<https://ko.reactjs.org/>

## 방법2. npx create-react-app 사용

- 리액트 앱(프로젝트) 이름 규칙
  - 대문자 사용 불가능
  - 단어 여러 개 사용시 하이픈(-) 기호로 구분
- npx
  - npm의 자식 명령어로 npm보다 가볍게 패키지를 구성
  - npm 버전이 5.2 이상일 때, node 버전이 14 이상일 때 사용 가능

```
$ npm install create-react-app -g
```

// create-react-app 모듈 사용할 수 있도록 global로 설치

```
$ npx create-react-app test-app
```

React App Name

# React 프로젝트 실행하기

```
npm start
```

```
Compiled successfully!
```

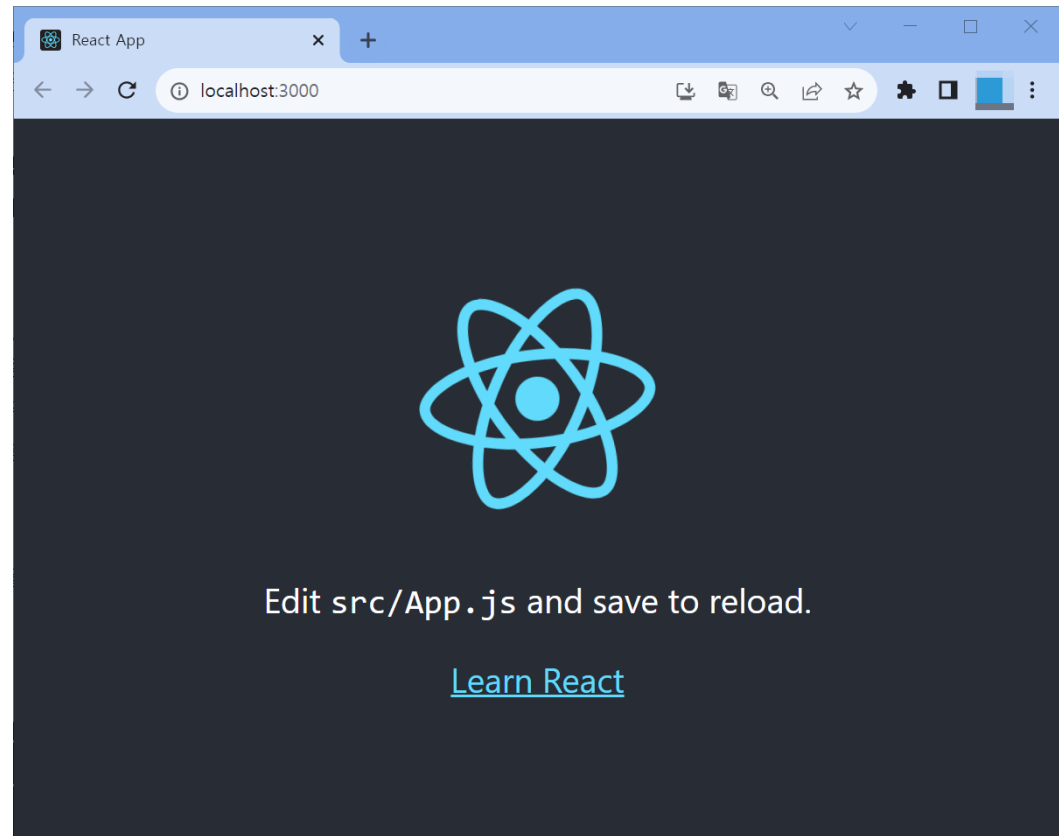
```
You can now view todo-app in the browser.
```

```
Local: http://localhost:3000
```

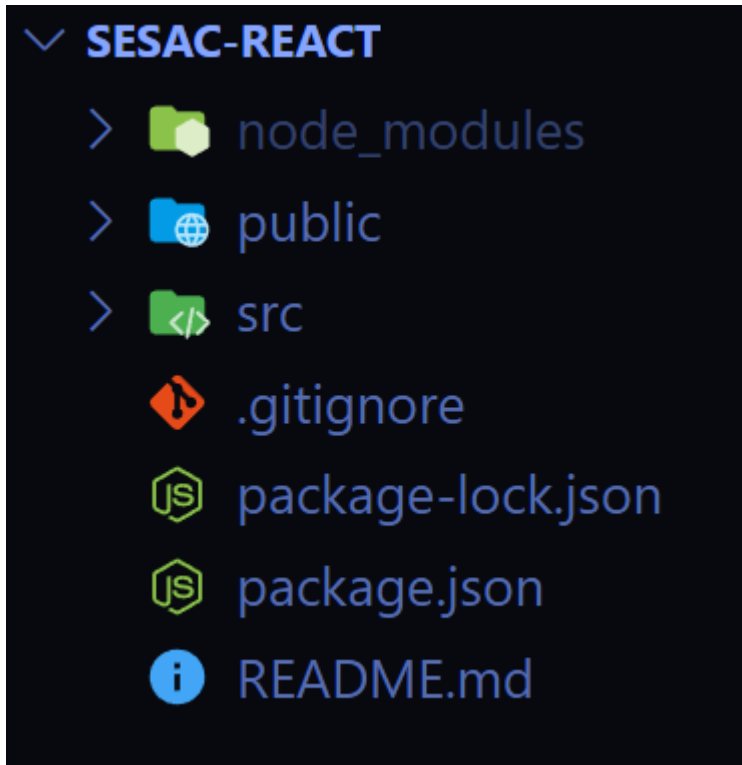
```
On Your Network: http://192.168.10.42:3000
```

```
Note that the development build is not optimized.  
To create a production build, use npm run build.
```

```
webpack compiled successfully
```



# create-react-app으로 생성한 프로젝트의 구조



**/node\_modules** : npm 을 시작하면 생기는 폴더, 내부에 모듈과 관련된 정보가 들어있음

**/public** : index.html & favicon.ico! → 가상DOM을 위한 html 파일이 들어있는 곳

**/src** : 실제 React 코드(컴포넌트)를 작성하는 곳



.gitignore

package-lock.json, package.json

README.md

# 정리

- `npx create-react-app [프로젝트 이름]` 로 프로젝트 만들기
- 해당 프로젝트로 이동(`cd`) 후 `npm start` 를 하면 만들어진 페이지를 `localhost:3000` 에서 볼 수 있음
- 앞으로 `src` 폴더 안에 `js` 이용해서 컴포넌트를 만들어볼 예정!