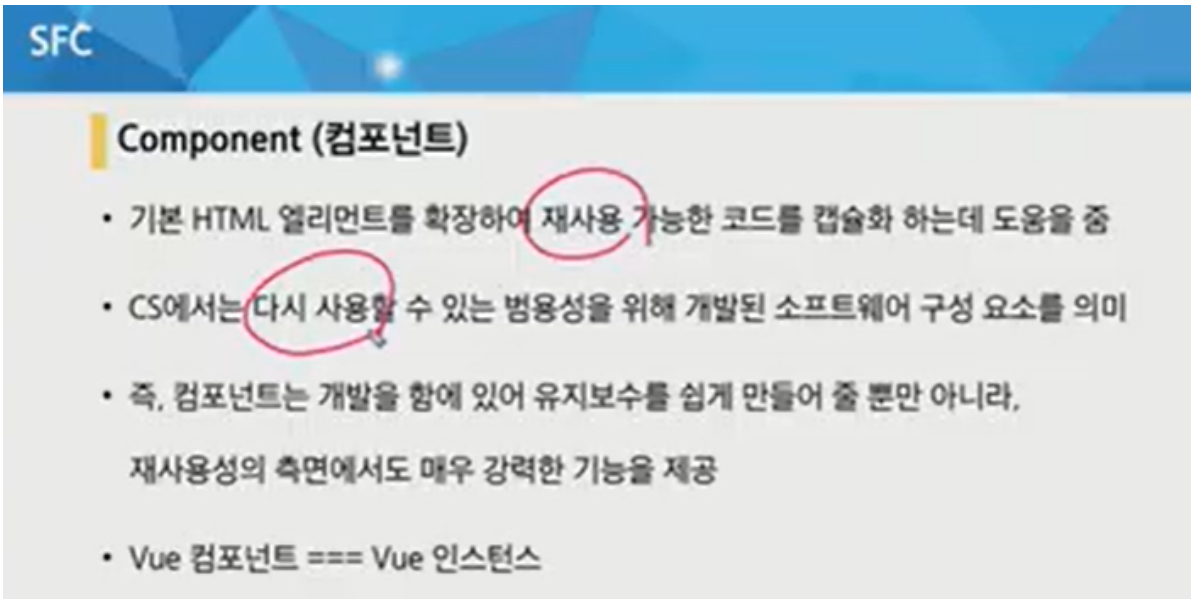
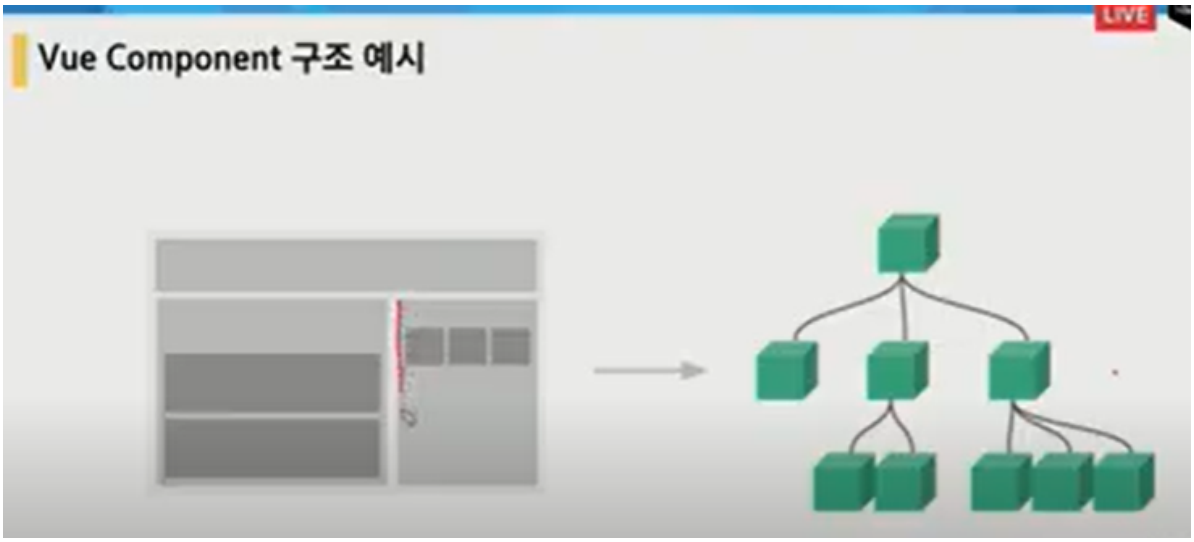
SFC



const app = new Vue()

이렇게 했을 때 app이 component

구조

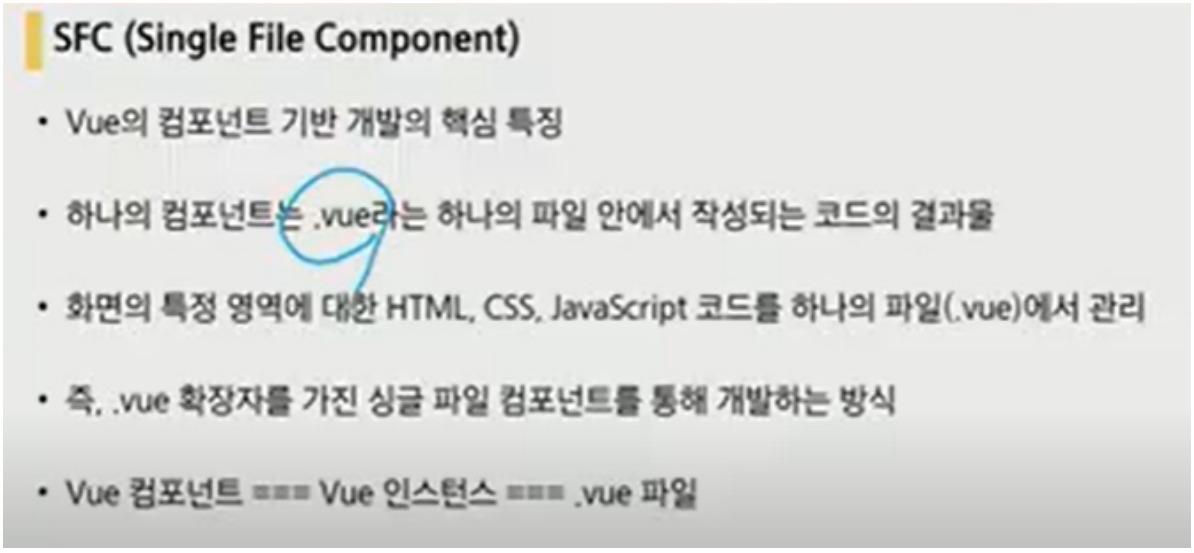


쪼개면 쪼갤수록 유지 보수에 용이

상하관계가 있음(큰 네모 안에 작은 네모 여러개)

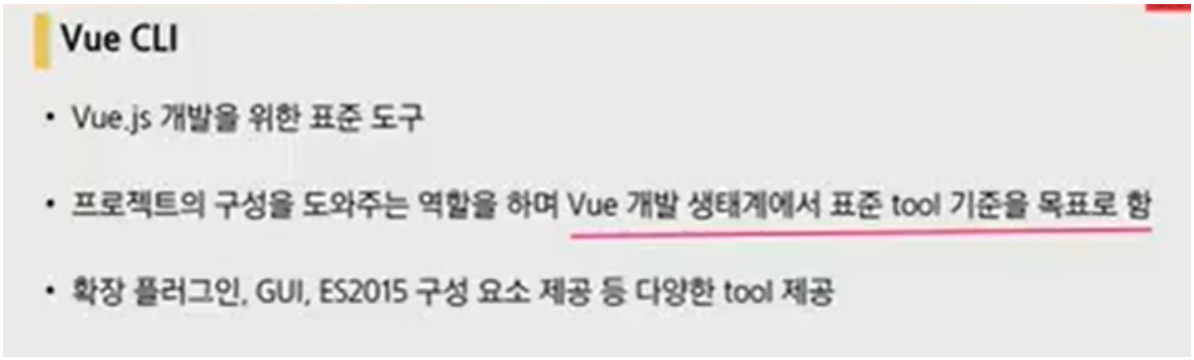
SFC

하나의 파일(vue)이 하나의 컴포넌트



Vue CLI

command line, 명령어

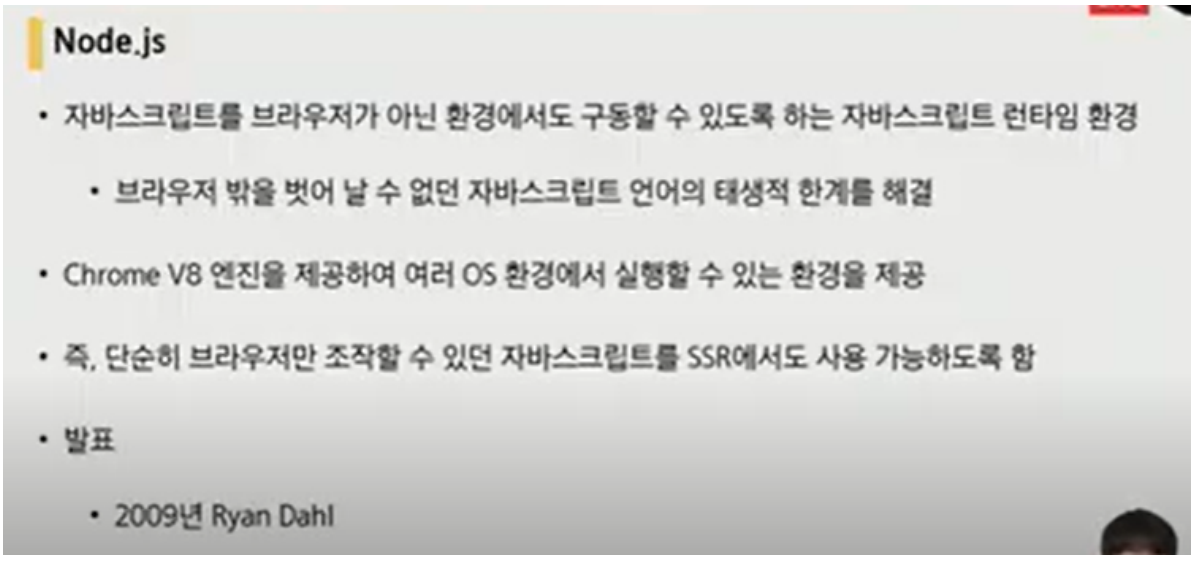


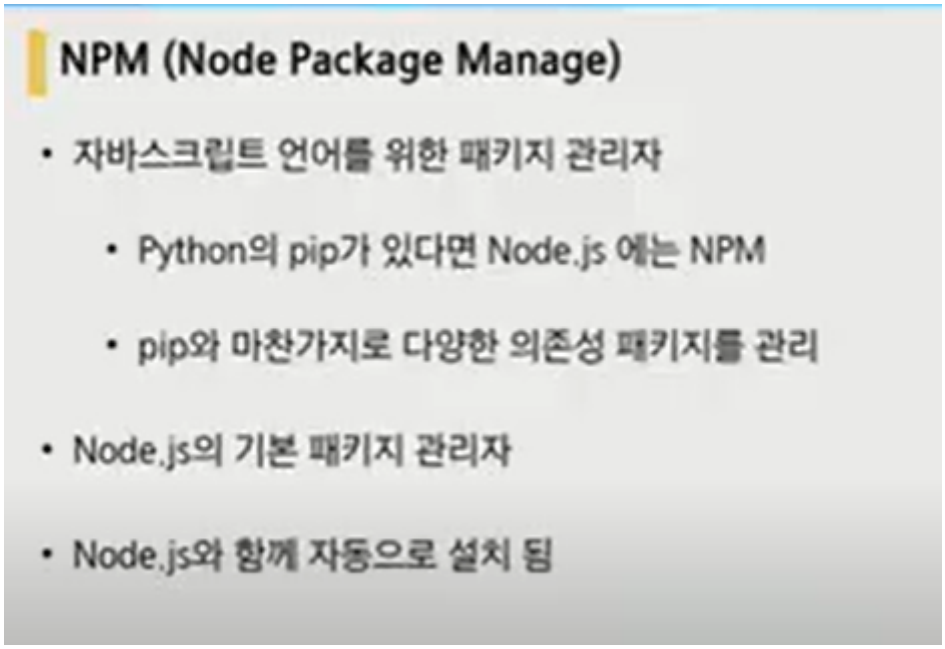
Node.js

Vue cli를 사용하려면 node.js를 설치해야함

자바는 브라우저 안에서만 동작한다는 한계가 있었음

노드js가 탄생하면서 serversiderendering이 가능해지고 지금같은 인기를 갖게된 계기





Vue CLI 설치

설치

vue 3이 릴리즈 되긴 했지만 vue 2로 설치

Babel

Babel

- JavaScript Transcomiler
- 자바스크립트의 신버전 코드를 구버전으로 번역/변환 해주는 도구
- 자바스크립트 역사에 있어서 파편화와 표준화의 영향으로 작성된 코드의 스펙트럼이 매우 다양
  - 최신 문법을 사용해도 브라우저의 버전별로 동작하지 않는 상황이 발생
  - 같은 의미의 다른 코드를 작성하는 등의 대응이 필요해졌고 이러한 문제를 해결하기 위한 도구
- 원시 코드(최신 버전)를 목적 코드(구 버전)으로 옮기는 번역기가 등장하면서 개발자는 더 이상 내 코드가 특정 브라우저에서 동작하지 않는 상황에 대해 크게 고민하지 않을 수 있음

Babel 동작 예시

JavaScript Copy

```
// Babel Input: ES2015 arrow function
[1, 2, 3].map((n) => n + 1);

// Babel Output: ES5 equivalent
[1, 2, 3].map(function(n) {
  return n + 1;
});
```

원시 코드 → compile → 목적 코드

Webpack

Webpack

- static module bundler
- 모듈 간의 의존성 문제를 해결하기 위한 도구

Module

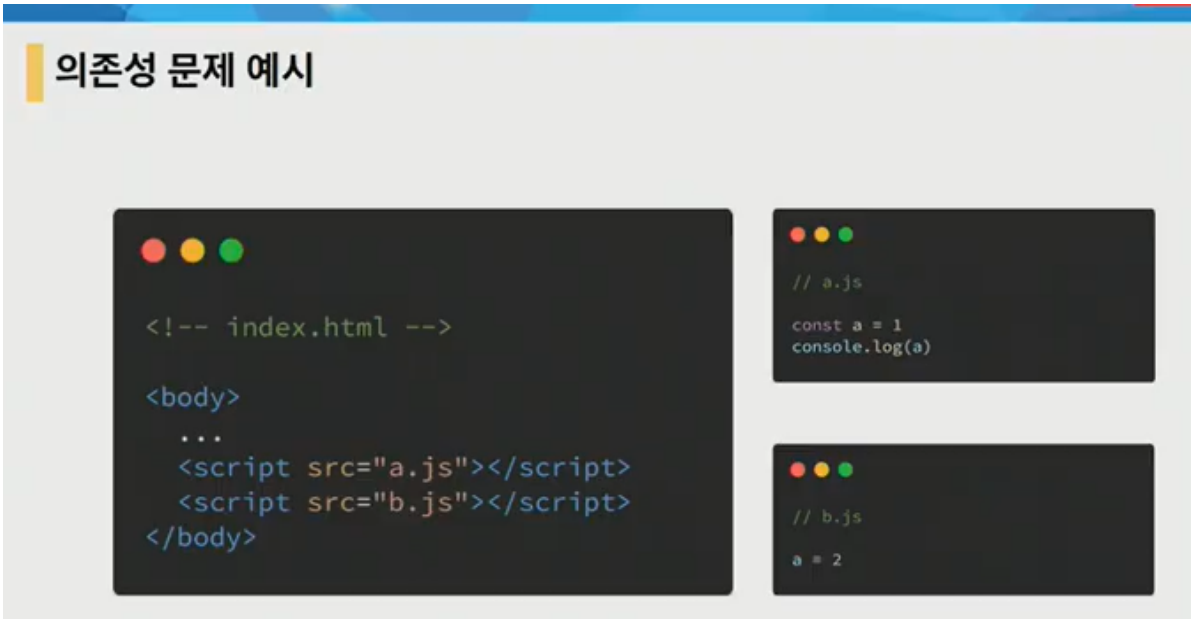
- 모듈은 단지 파일 하나를 의미 (ex. 스크립트 하나 === 모듈 하나)
- 배경
  - 브라우저만 조작할 수 있었던 시기의 자바스크립트는 모듈 관련 문법 없이 사용 되어짐
  - 하지만 자바스크립트와 애플리케이션이 복잡해지고 크기가 커지자
    전역 스코프를 공유하는 형태의 기존 개발 방식의 한계점이 드러남
  - 그래서 라이브러리를 만들어 필요한 모듈을 언제든지 불러오거나
    코드를 모듈 단위로 작성하는 등의 다양한 시도가 이루어짐
- 과거 모듈 시스템
  - AMD, CommonJS, UMD
- 모듈 시스템 2015년 표준으로 등재 되었으며 현재는 대부분의 브라우저와 Node.js가 모듈 시스템을 지

Module 의존성 문제

- 모듈의 수가 많아지고 라이브러리 혹은 모듈 간의 의존성(연결성)이 깊어지면서 특정한 곳에서 발생한 문제가 어떤 모듈 간의 문제인지 파악하기 어려워짐 (의존성 문제)
- Webpack은 모듈 간의 의존성 문제를 해결하기 위해 존재하는 도구

Bundler

- 모듈 의존성 문제를 해결해주는 작업이 Bundling이고 이러한 일을 해주는 도구가 Bundler이고, Webpack은 다양한 Bundler 중 하나
- 모듈들을 하나로 묶어주고 묶인 파일은 하나(혹은 여러 개)로 만들어짐
- Bundling된 결과물은 더 이상 서-순에 영향을 받지 않고 동작하게 됨
- Bundling 과정에서 문제가 해결되지 않으면 최종 결과물을 만들어 낼 수 없기 때문에 유지 & 보수의 측면에서도 매우 편리해짐
  - snowpack, parcel, rollup.js 등의 webpack 이외에도 다양한 모듈 번들러 존재
- Vue CLI는 이러한 Babel, Webpack에 대한 초기 설정이 자동으로 되어 있음



const는 재할당, 재선언이 불가능하므로 오류 발생

src:webpack에 빌드되는 정적 파일들

components: 하위 컴포넌트가 들어감

app.vue: 최상위 컴포넌트

main.js: webpack이 빌드를 시작할때 가장 먼저 불러오는 시작점, 뷰의 전역에서 사용하는 모듈들을 등록

gitignore은 기본적으로 만들어줌

gitinit을 안했는데 자동으로 master가 생김(git 초기화가 저절로 생김)

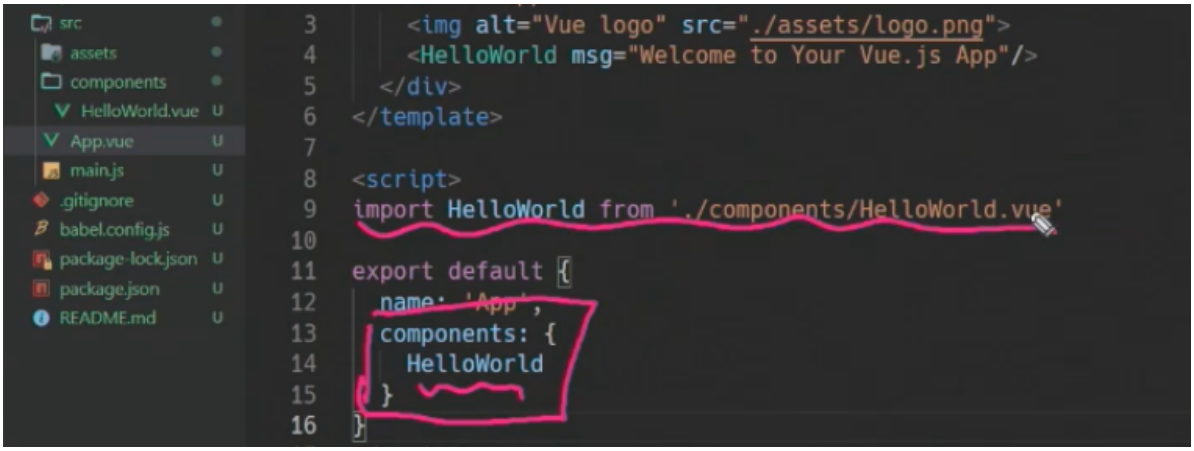
package.json: requirements.txt와 비슷한 역할, 우리가 건들일 x

package-lock.json: 팀원들과 동일한 환경(종속성)을 유지하기 위한 파일, 우리가 건들일 x

app.vue

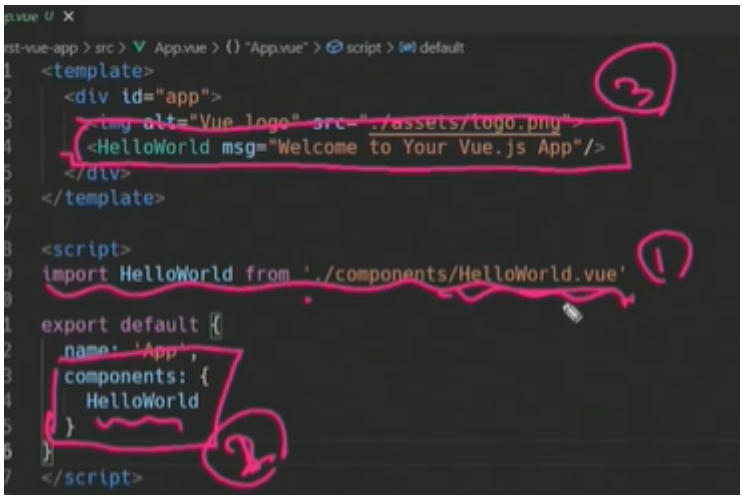
template(html) - script(js) - style(css)

.vue가 하나의 컴포넌트



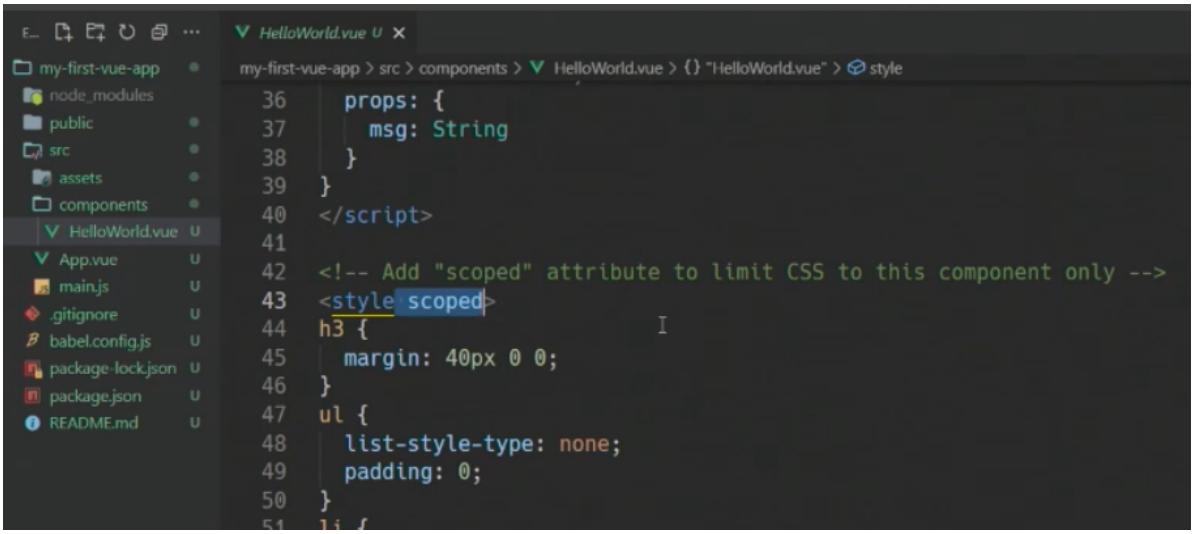
app.vue와 하위 컴포넌트의 연결 확인

불러와서 - 등록하고 - 사용



style scope 뜻

이 스타일은 딱 이 .vue에서 한정되게 동작한다는 뜻



component 만들기

new component 만듦

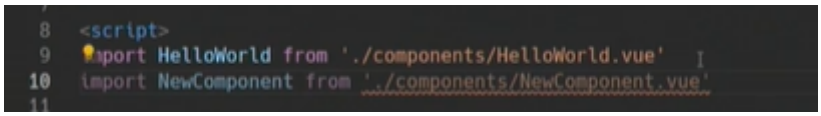
vue 하고 엔터하면 3등분 자동완성

template 안에는 하나의 최상위 태그가 있다

app 이름과 동일하게 name 설정

꼭 그럴 필요는 없지만 그러지 않을 이유도 없음

1. 불러오기



2. 등록하기

3. 사용하기(닫는 태그 작성)