Vue 프로젝트

**학습목표**

@vue/cli 설치

vue 프로젝트 구조 이해

**목차**

[1. Visual Studio Code 확장 설치 2](#_Toc73325565)

[2. Vue 프로젝트 생성 4](#_Toc73325566)

[1) 기초 vue 앱 4](#_Toc73325567)

[2) 고급 vue 앱 5](#_Toc73325568)

[3) 용어 정리 6](#_Toc73325569)

[4) @vue/cli 설치 7](#_Toc73325570)

[5) vue\_app1 프로젝트 생성 7](#_Toc73325571)

[6) vue\_app1 프로젝트 실행 8](#_Toc73325572)

[7) 포트 번호 변경 9](#_Toc73325573)

[3. 소스코드 수정 10](#_Toc73325574)

[1) 프로젝트 폴더 구조 10](#_Toc73325575)

[2) vue.config.js 생성 11](#_Toc73325576)

[3) App.vue 수정 12](#_Toc73325577)

[4) HelloWorld.vue 수정 13](#_Toc73325578)

[4. 프로젝트 빌드 & 배포 14](#_Toc73325579)

[1) 프로젝트 빌드 14](#_Toc73325580)

[5. 요약 15](#_Toc73325581)

[1) 기초 Vue 프로젝트 15](#_Toc73325582)

[2) Vue.js프로젝트 15](#_Toc73325583)

[6. 부록: vue.config.js 설명 16](#_Toc73325584)

# Visual Studio Code 확장 설치

Visual Studio Code의 기능을 확장해 주는 플러그인들이 매우 많다.

Vue 개발에 편리한 확장 플러그인들을 설치하자.

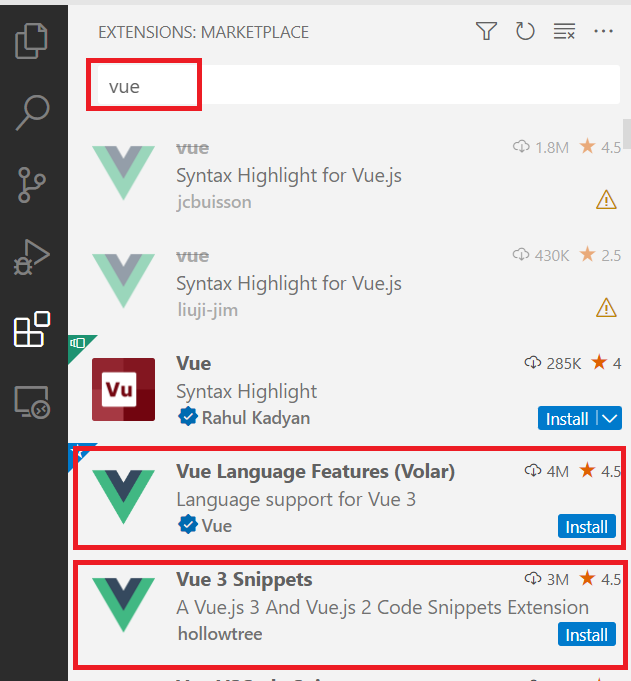
### Vetur 설치

Graphical user interface, application

Description automatically generated

install 버튼을 클릭하자 마자 빠르게 설치된다.

### Vue Language Features 설치, Vue 3 Snippets 설치



# Vue 프로젝트 생성

## 기초 vue 앱

지금까지 만든 vue 예제들은, 기초 vue 앱 예제들이다.

기초 vue 앱의 특징은 다음과 같다.

### ES6 버전의 표준 javascript 문법으로 구현한다

javascript 표준 문법에도 버전이 있다.

vue.js 엔진은 ES6 버전의 javascript 문법으로 구현되었다.

internet explorer version 11은 이 문법을 제대로 지원하지 못한다.

그래서 vue.js는 ie11에서 실행되지 않는다.

### CDN (Contents Delivery Network 서비스) 이용

<script src="https://unpkg.com/vue@next"></script>

html 문서의 head 부분에 위 태그를 넣어주면,

웹브라우저가 vue 라이브러리를 위 URL의 서버로부터 받아와서 실행한다.

널리 사용되는 라이브러리 파일들을, 이런 방식으로 제공해주는 서비스를 CDN 이라고 한다.

<script src="https://unpkg.com/axios/dist/axios.min.js"></script>

axios 라이브러리도 웹브라우저가 CDN 서비스 서버로부터 받아와서 실행한다.

### 프로젝트를 빌드(build)할 필요가 없다

javascript 소스코드 파일을 컴파일하거나 변환할 필요가 없기 때문에,

프로젝트 빌드 작업도 필요 없다.

### vue 컴포넌트를 구현하기 불편하다.

우리는 아직 vue 컴포넌트 구현 방법을 배우지 않았다.

기초 vue 앱에서는 컴포넌트를 구현하기 불편하기 때문이다.

## vue 전용 프로젝트

지금부터 우리가 만들 vue 예제들은, 고급 vue 앱 예제들이다.

고급 vue 앱의 특징은 다음과 같다.

### 확장된 javascript 문법으로 구현한다

컴포넌트를 구현하기에 편한 확장된 javascrpt 문법으로 구현한다.

그런데 이 확장된 javascript 문법은 웹브라우저가 이해할 수 없기 때문에,

표준 javascript 문법으로 변환해야, 웹브라우저에서 실행될 수 있다.

이렇게 변환하는 작업을 트랜스파일(transpile)이라고 한다.

### 트랜스파일(transpile) 해야한다

고급 vue 프로젝트는, 실행하기 전에 소스 코드 파일을 트랜스파일 해야 한다.

확장된 문법의 vue 소스코드 파일의 확장자는 \*.vue 이다. 이 파일을 표준 javascript 파일(\*.js)로 변환해야 한다.

소스 코드를 기계어 코드나 바이트 코드로 변환하는 작업을 컴파일(compile)이라고 하고,

소스 코드를 다른 문법의 소스 코드로 변환하는 작업을 트랜스파일(transpile)이라고 한다.

### ie11에서도 실행된다

트랜스파일 결과 생성된 javascript 파일은, ie11도 지원하는 javascript 문법을 사용한다. (ES5 문법)

그래서 거의 대부분의 웹브라우저에서 실행 가능하다.

### 압축(minify) 한다

javascript 파일의 크기가 너무 크면,

웹브라우저가 서버로부터 그 파일을 받아갈 때도 무겁고,

웹브라우저가 그 파일을 분석해서 실행할 때도 무겁다.

변수 이름이나 함수 이름을 짧게 줄이고, 연속된 공백이나 줄바꿈 문자를 공백 문자 한개로 줄이면,

기능 변화는 없지만, javascript 파일의 크기를 많이 줄일 수 있다.

이렇게 줄이는 작업을 minify 이라고 한다.

이렇게 줄인 javascript 파일의 이름은 \*.min.js 형태인 것이 관례이다.

### npm 패키지 이용

CDN 서비스로 vue 라이브러리와 axios 라이브러리를 이용하지 않는다.

(1) vue 라이브러리와 axios 라이브러리를 개발 PC로 다운로드하고, (라이브러리 == npm 패키지)

(2) 개발자가 구현한 소스 코드 파일 뿐만 아니라, 라이브러리 파일들도 transpile 하고 minify 해서,

(3) 배포할 파일을 만든다.

이 배포 파일에는 필요한 라이브러리도 들어있고, 내가 구현한 소스코드 들어있다.

### 프로젝트를 빌드(build)해야 한다.

build = transpile + minify

개발자가 구현한 소스 코드 파일과 라이브러리 파일들을 transpile 하고 minify 해서,

배포할 파일을 만든다. => 빌드(build) 한다.

웹브라우저는 이 배포 파일만 서버로부터 받아와서 실행하면 된다.

이 배포 파일은 transpile 되었기 때문에 웹브라우저 호환성도 높고 (internet explorer에서도 실행된다)

minify 되었기 때문에 사이즈도 작다.

### vue 컴포넌트를 구현하기 편하다.

vue 컴포넌트를 구현해야 한다면, 고급 vue 앱 방식으로 개발하는 것이 좋다.

## 용어 정리

### transpile

소스 코드를 다른 문법의 소스 코드로 변환하는 작업을 트랜스파일(transpile)이라고 한다.

### minify

변수 이름이나 함수 이름을 짧게 줄이고, 연속된 공백이나 줄바꿈 문자를 공백 문자 한개로 줄여서,

구현된 기능 변화 없이 javascript 파일의 크기만 줄이는 작업.

### babel

javascript 소스 코드 transpile & minify 도구

### webpack

javascript 소스 코드 파일과 라이브러리(npm 패키지) 파일들을

배포 파일로 묶어주는 도구

### eslint

javascript 소스 코드 오류를 찾아서 알려주는 도구

|  |
| --- |
| **주의**  git bash 쉘에서 작업할 것이라면 꼭 Visual Studio Code 터미널에서 git bash 쉘을 열어야 한다.  Visual Studio Code 터미널이 아닌, git bash 창에서 명령을 실행하면,  화살표 키가 잘 작동하지 않는다. |

## @vue/cli 설치

### @vue/cli

고급 vue 프로젝트를 생성, 빌드, 실행, 테스트하기 위한 도구이다.

### CLI

Command Line Interface 약자

쉘에서 명령을 입력하는 방식을 발한다.

CLI의 반댓말은 GUI (Graphic User Interface)

### 설치 명령

|  |
| --- |
| npm install -g @vue/cli |

### 설치 확인

다음 명령을 실행해서 @vue/cli 설치를 확인하자.

|  |
| --- |
| vue --version |

|  |
| --- |
|  |

## vue\_app1 프로젝트 생성

프로젝트를 생성할 적당한 폴더로 이동하자.

|  |
| --- |
| cd /c/PJ/frontend |

아래 명령을 실행해서 프로젝트를 생성하자.

|  |
| --- |
| vue create vue\_app1 |

|  |
| --- |
|  |

Default ([Vue 3] babel, eslint) 항목이 선택된 상태에서 엔터 키를 누르자.

이 설정에는 babel 도구와 eslint 도구 설정이 포함된다.

조금 기다리면 프로젝트 생성이 완료된다.

@vue/cli 도구가 프로젝트 설정의 많은 부분을 자동으로 해주기 때문에,

프로젝트 생성 작업이 쉬워졌다.

## vue\_app1 프로젝트 실행

|  |
| --- |
| cd vue\_app1 |

프로젝트 폴더로 이동하기

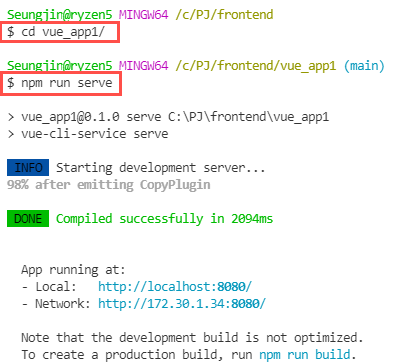
Visual Studio Code에서 프로젝트 디렉토리를 열어야 한다.

Vue 프로젝젝트를 실행하기 전에 먼저 빌드해야 한다.

빌드하기 전에는 live-server로 실행할 수 없다.

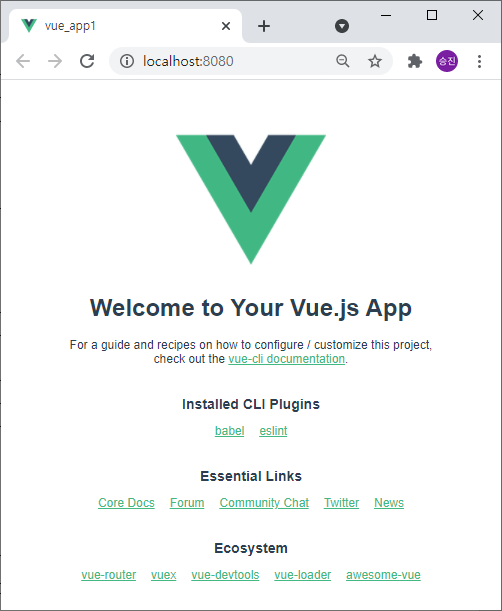
아래 명령으로 실행하면, 프로젝트를 빌드하지 않고도 실행할 수 있다. 이 명령을 기억하자.

|  |
| --- |
| npm run serve |



웹브라우저 창에서 다음 URL을 열자

http://localhost:8080



vue cli가 만들어준 페이지가 실행되고 있는 화면이다.

현재 vue cli 개발 서버에서 프로젝트가 실행되고 있다.

## 포트 번호 변경

컴퓨터에 오라클 DBMS나 톰캣이 설치되어 있다면, 8080 포트가 이미 사용중일 수 있다.

vue 프로젝트가 사용할 포트 번호를 8082로 변경하는 방법은 다음과 같다.

### package.json

|  |
| --- |
| ... 생략 ...    "scripts": {      "serve": "vue-cli-service serve --port 8082",      "build": "vue-cli-service build",      "lint": "vue-cli-service lint"    },  ... 생략 ... |

package.json 파일을 열고 위의 내용을 찾아서,

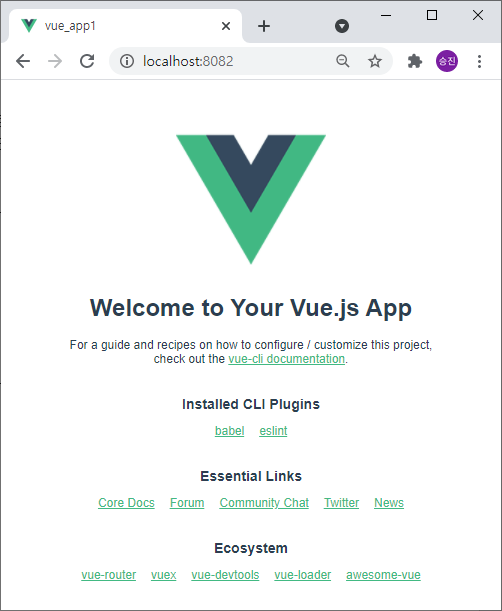
노란색으로 칠한 부분을 추가하자.

### 서버 재시작

npm run serve 명령이 실행중인 터미널 창에서 Ctrl+C 키를 눌러서 종료하자.

그리고 다시 npm run serve 명령을 실행하자.

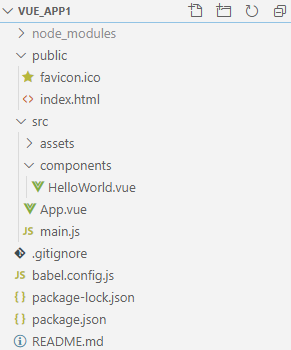
http://localhost:8082



# 소스코드 수정

## 프로젝트 폴더 구조

visual studio code에서 vue\_app1 폴더를 열자



vue cli 가 자동으로 생성한 프로젝트 소스 파일들이다.

### node\_modules 폴더

npm 패키지(라이브러리) 파일들이 다운로드 되는 폴더이다.

이 폴더는 git repository에 등록되지 않아야 한다.

### public/index.html 파일

앱의 시작에 해당하는 html 파일이다.

이 파일의 내용을 열어보면, <div id="app"> 태그 외에는 별 내용이 없다.

### src 폴더

소스코드 파일이 위치하는 폴더이다.

### src/main.js

앱의 시작에 해당하는 javascript 파일이다.

이 파일에서 App.vue 컴포넌트를 사용하고 있다.

### App.vue 파일

\*.vue 파일은 확장된 문법으로 구현된 vue 컴포넌트 소스 코드 파일이다.

마치 java 프로젝트의 소스 코드가 java 클래스인 것처럼,

vue 프로젝트의 소스 코드가 vue 컴포넌트이다.

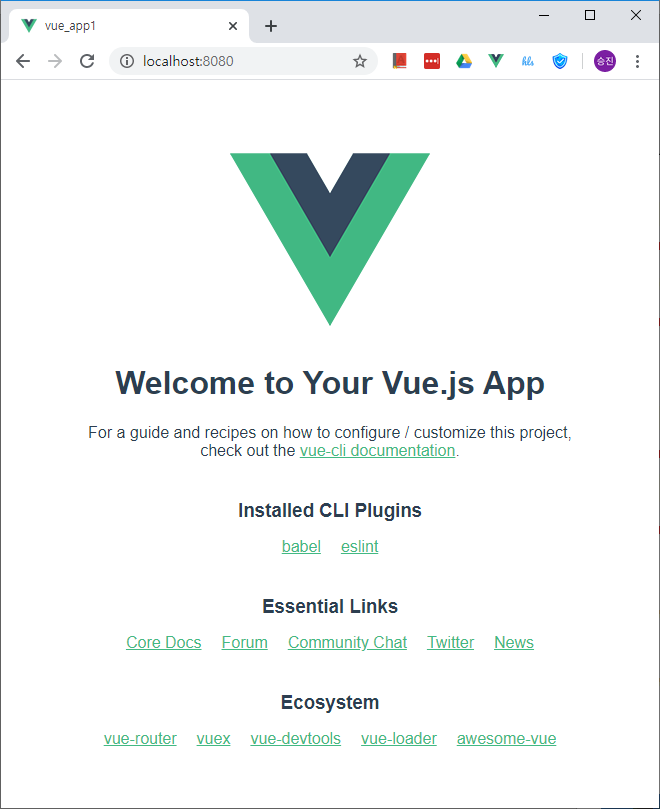
App.vue 컴포넌트는, 앱의 시작 컴포넌트이다.

App.vue 컴포넌트에서 HelloWord.vue 컴포넌트를 사용하고 있다.

### src/components 폴더

vue 컴포넌트 소스 코드 파일이 위치할 폴더이다.

### HelloWord.vue



이 화면을 구현한 vue 컴포넌트이다.

### .gitignore

git repository에 등록하지 않을 파일/폴더 목록이 들어있는 설정 파일이다.

### babel.config.js

babel transpiler 설정 파일이다.

### package.json

javascript 프로젝트 설정 파일이다.

## vue.config.js

|  |
| --- |
| const { defineConfig } = require('@vue/cli-service')  module.exports = defineConfig({    publicPath: './',    configureWebpack: {      devtool: 'source-map'    },    transpileDependencies: true  }) |

프로젝트 폴더에 vue.config.js 파일을 수정하자.

이것은 vue cli 설정 파일이다.

노란색으로 칠한 부분을 추가하자.

이 설정에 대한 설명은 뒤에서...

## App.vue 수정

|  |
| --- |
| <template>    <div id="app">      <HelloWorld message="첫번째 Vue.js App 입니다" />    </div>  </template>  <script>  import HelloWorld from './components/HelloWorld.vue'  export default {    name: 'App',    components: {      HelloWorld    }  }  </script>  <style scoped>  div#app { padding: 30px; margin: 30px auto; max-width: 400px;        border: 1px solid #ccc; box-shadow: 3px 3px 3px #aaa; }  </style> |

vue 컴포넌트 소스 코드 파일은 \*.vue 이다.

이 파일의 구조는 다음과 같다.

|  |
| --- |
| <template>  html 태그  </template>  <script>  javascript 소스코드  <script>  <style>  CSS 서식  </stype> |

<HelloWorld message="첫번째 Vue.js App 입니다" />

HelloWorld 컴포넌트를 여기에 출력한다.

HelloWorld 컴포넌트에 전달한 파라미터는 message="첫번째 Vue.js App 입니다" 이다.

import HelloWorld from './components/HelloWorld.vue'

HelloWorld.vue 파일에서 HelloWord 컴포넌트를 import 한다.

export default {

  name: 'App',

  components: {

    HelloWorld

  }

}

이 컴포넌트의 이름은 App 이다.

이 컴포넌트는 내부에서 HelloWorld 컴포넌트를 사용한다.

<style scoped>

이 서식은 이 컴포넌트에만 적용된다.

## HelloWorld.vue 수정

|  |
| --- |
| <template>    <div>      <h1>안녕하세요</h1>      <p>{{ message }}</p>    </div>  </template>  <script>  export default {    name: 'HelloWorld',    props: {      message: String    }  }  </script>  <!-- Add "scoped" attribute to limit CSS to this component only -->  <style scoped>    p { color: blue; }  </style> |

export default {

  name: 'HelloWorld',

  props: {

    message: String

  }

}

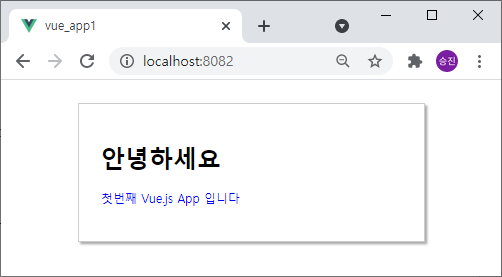
이 컴포넌트의 이름은 Helloworld 이다.

이 컴포넌트의 파라미터는 message 이다. 컴포넌트의 파라미터를 props 라고 부른다.

  <HelloWorld message="첫번째 Vue.js App 입니다" />

<p>{{ message }}</p>

message props 값을 여기에 출력한다.



# 프로젝트 빌드 & 배포

## 프로젝트 빌드

|  |
| --- |
| npm run build |

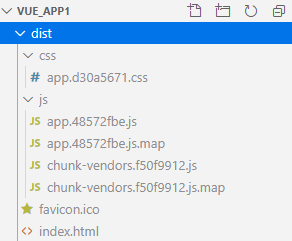
프로젝트 빌드 명령은 다음과 같다.

|  |
| --- |
|  |

위 명령에 의해서 프로젝트 소스 코드 파일이 transpile 되고 minify 된다.

빌드 결과로 생성되는 파일이 프로젝트 폴더 아래 dist 폴더에 생성된다.

### dist 폴더



실행 서버(production server)에 배포할 때,

dist 폴더 아래의 파일들만 복사하면 된다.

예를 들어 Spring Boot 프로젝트의 webapp 폴더에, dist 폴더 아래의 파일들만 복사하면 된다.

# 요약

## 기초 Vue 프로젝트

우리가 지금까지 만들었던 Node.js 프로젝트에서 Vue 앱을 개발했던 방식.

CND 서비스를 이용하여 Vue 엔진을 사용함.

표준 javascript 문법으로 구현한다.

빌드할 필요 없다.

컴포넌트를 구현하기 어렵다.

보통 소스코드 파일 한 개로 구현하기 때문에, 간단한 기능을 구현할 때 유용하다.

## Vue.js프로젝트

Vue CLI 도구로 프로젝트를 생성하고 빌드하고 실행함.

확장된 문법으로 구현한다.

빌드를 해야 한다.

컴포넌트 단위로 단위로 구현하기 때문에, 복잡한 기능을 구현할 때 유용하다.

컴포넌트 소스 코드 파일을 재사용하기 좋다.

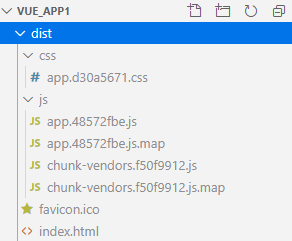
오픈소스 컴포넌트를 활용할 수 있다.

# 부록: vue.config.js 설명

|  |
| --- |
| const { defineConfig } = require('@vue/cli-service')  module.exports = defineConfig({    publicPath: './',    configureWebpack: {      devtool: 'source-map'    },    transpileDependencies: true  }) |

프로젝트 폴더의 vue.config.js 파일은 vue cli 설정 파일이다.

### publicPath: './'



빌드 결과 생성된 css 파일과 js 파일들에 대한 링크가 index.html 파일에 있다.

이 링크에서 사용하는 파일 경로는 기본적으로 절대 경로이다.

예:

|  |
| --- |
| <link rel=icon href=/favicon.ico>  <link href=/css/app.ccf402d9.css rel=preload as=style>  <link href=/js/app.7a33bb6c.js rel=preload as=script>  <link href=/js/chunk-vendors.252a99fb.js rel=preload as=script>  <link href=/css/app.ccf402d9.css rel=stylesheet> |

이렇게 절대 경로가 사용되었으면, 이 파일들을 서버에 배포할 때, 프로젝트 루트 폴더(/)에 복사해야 한다.

프로젝트 루트 폴더 아래의 서브 폴더에 복사하면, 링크된 파일을 찾지 못하는 문제가 발생한다.

publicPath: './' 이 설정이 있으면, 링크에서 사용하는 파일 경로는 상대 경로이다.

예:

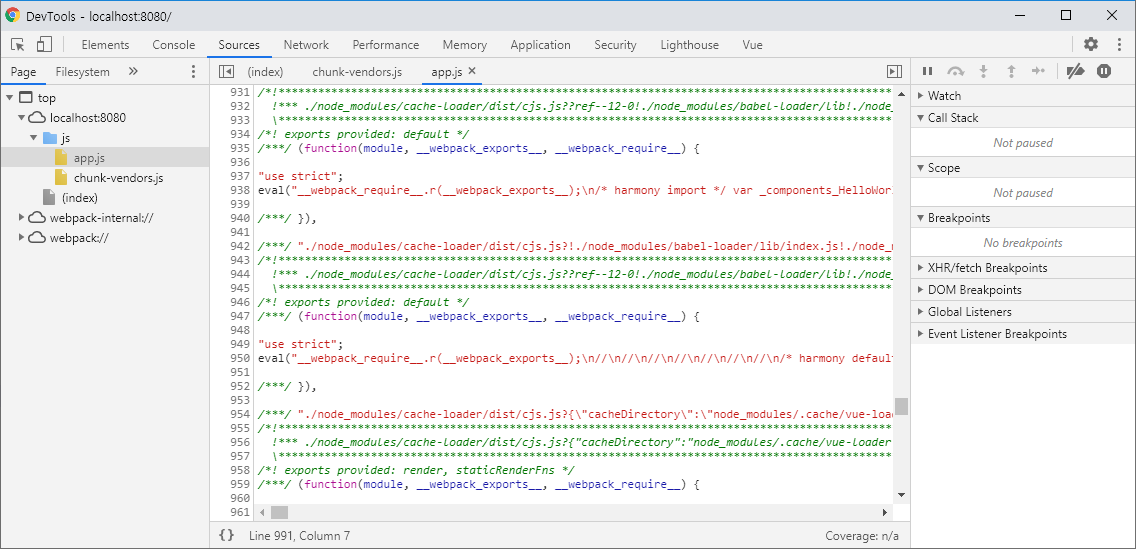
|  |
| --- |
| <link rel=icon href=favicon.ico>  <link href=css/app.ccf402d9.css rel=preload as=style>  <link href=js/app.7a33bb6c.js rel=preload as=script>  <link href=js/chunk-vendors.252a99fb.js rel=preload as=script>  <link href=css/app.ccf402d9.css rel=stylesheet> |

이렇게 상대 경로가 사용되었으면, 이 파일들을 서버에 배포할 때, 프로젝트 루트 폴더 아래에 폴더를 만들고 그 아래에 복사해도 된다.

### configureWebpack: { devtool: 'source-map' }

웹브라우저에서 실행되는 javascript 파일은, 트랜스 파일된 결과 \*.js 파일이다.

그래서 웹브라우저의 개발자창의 Sources 탭에서 보이는 소스코드도 트랜스파일된 결과 \*.js 파일이다.



configureWebpack: { devtool: 'source-map' }

위 설정이 있으면, 웹브라우저 개발자창의 Sources 탭에서 \*.vue 소스코드를 볼 수 있다.

웹브라우저 개발자창에서 디버거 기능을 실행할 때 꼭 필요한 기능이다.

