

# Vue.js Tutorial

---



1. Vue.js 소개
2. Vue.js 설치
3. Vue CLI
4. Single File Component
5. Vue 인스턴스(Instance)
6. Vue 컴포넌트(Components)
7. Vue 라우터(Routers)
8. Axios
9. Vue Template

## Vue.js란?

SPA(Single Page Application)를 사용하는 Javascript 프론트엔트 프레임워크

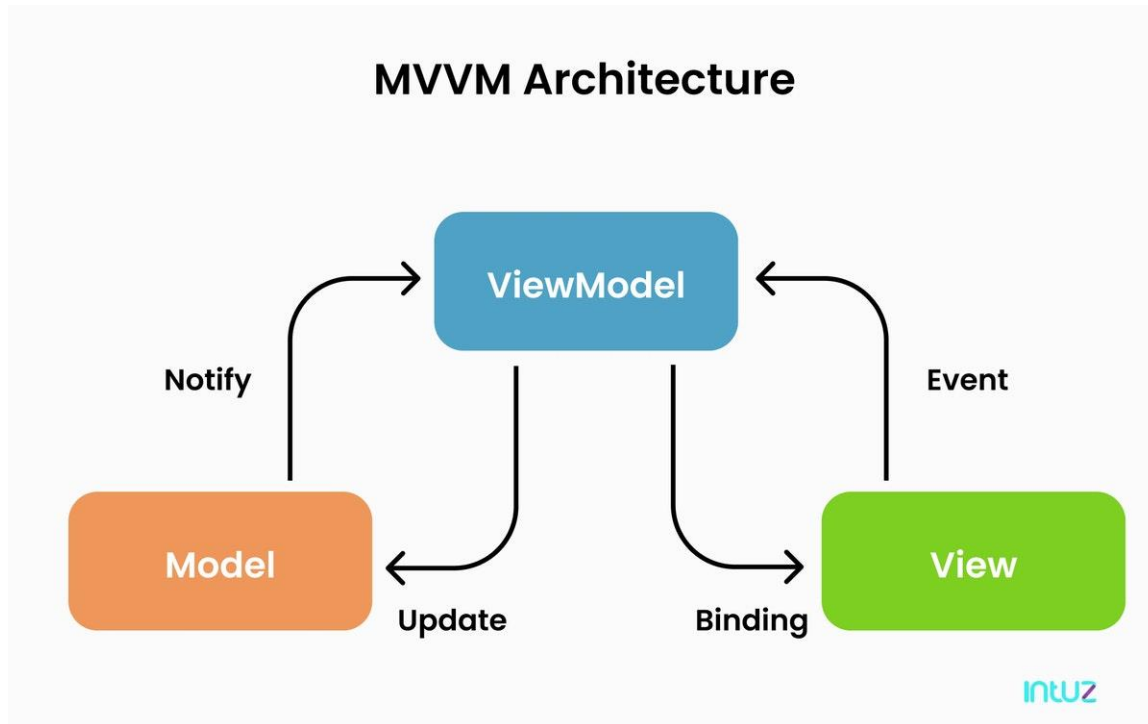
MVVM 패턴의 ViewModel 레이어에 해당하는 화면단 라이브러리



\* SPA(Single Page Application) :

하나의 페이지를 사용하는 애플리케이션. 서버로부터 새로운 페이지를 가져오는 것이 아닌, 하나의 페이지에서 내용을 동적으로 변경하는 웹앱.

## MVVM 패턴



### 낮은 의존성과 양방향 통신

View와 Model 사이의 의존성이 없고 양방향 통신이 가능하다.

### 높은 확장성

각 각의 부분이 독립적이기 때문에 모듈화 하여 개발할 수 있습니다.

\* MVVM = Model View View Model

Model : 어플리케이션에서 사용되는 데이터와 그 데이터를 처리하는 부분

View : 사용자에서 보여지는 UI 부분

View Model : View를 표현하기 위한 Model이자 View를 나타내기 위한 데이터 처리를 하는 부분

# Node.js 설치

nodejs

홈 | ABOUT | 다운로드 | 문서 | 참여하기 | 보안 | CERTIFICATION | 뉴스

🌙

다운로드

최신 LTS 버전: 18.15.0 (includes npm 9.5.0)

플랫폼에 맞게 미리 빌드된 Node.js 인스톨러나 소스코드를 다운받아서 바로 개발을 시작하세요.

LTS  
대다수 사용자에게 추천

현재 버전  
최신 기능

Windows Installer

node-v18.15.0-x64.msi

macOS Installer

node-v18.15.0.pkg

Source Code

node-v18.15.0.tar.gz

Windows Installer (.msi)

Windows Binary (.zip)

macOS Installer (.pkg)

macOS Binary (.tar.gz)

Linux Binaries (x64)

Linux Binaries (ARM)

Source Code

32-bit	64-bit
32-bit	64-bit
64-bit / ARM64	
64-bit	ARM64
64-bit	
ARMv7	ARMv8
node-v18.15.0.tar.gz	

그 밖의 플랫폼

Docker Image

Linux on Power LE Systems

Linux on System z

AIX on Power Systems

Official Node.js Docker Image

64-bit

64-bit

64-bit

<https://nodejs.org/ko/download> 에 접속 후 자신의 운영체제에 맞는 버전 다운로드

# Node.js 설치 확인 및 Vue.js 설치

Node.js 설치 확인

```
@LAPTOP-DG:/opt$ node -v  
v10.19.0  
@LAPTOP-DG:/opt$ |
```

\$ node -v

Vue.js 설치 및 설치확인

```
@LAPTOP-DG:/opt$ npm install vue  
^LAPTOP-DG:~$ npm vue -v  
6.14.4
```

\$ npm install vue

\$ npm vue -v

## Vue CLI 설치

### NPM 방식의 설치

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

### Vue 프로젝트 생성


```
vue create 프로젝트 이름
```

```
Vue CLI v5.0.8
? Please pick a preset: (Use arrow keys)
> Default ([Vue 3] babel, eslint)
  Default ([Vue 2] babel, eslint)
  Manually select features
```

### 애플리케이션 실행

```
cd 프로젝트 폴더 이름
npm run serve
```

### Single File Component : HTML, CSS, JS 코드를 모두 관리할 수 있는 Vue 확장자 파일



```
<template>
  <!-- HTML 내용 -->
  <!-- 화면에 표시 할 요소를 정의하는 영역 -->
</template>

<script>
  // Javascript 내용
  // Vue Component 내용을 정의하는 영역
</script>

<style>
  /* CSS 내용*/
  /* 템플릿에 추가한 HTML 태그의 CSS 스타일을 정의하는 영역 */
</style>
```

\* .vue 파일은 하나의 모듈



**Vue 인스턴스(Instance)** : 인스턴스 옵션과 라이프사이클 단계에 따른 명령을 작성하는 코드

```
<script>
  new Vue({
    data: {
      a: 1
    },
    created: function() {
      // this 는 vm 을 가리킴
      console.log("a is: " + this.a);
    }
  });
</script>
```

### 인스턴스 옵션

data, template, method 등 관리

### 라이프사이클 단계

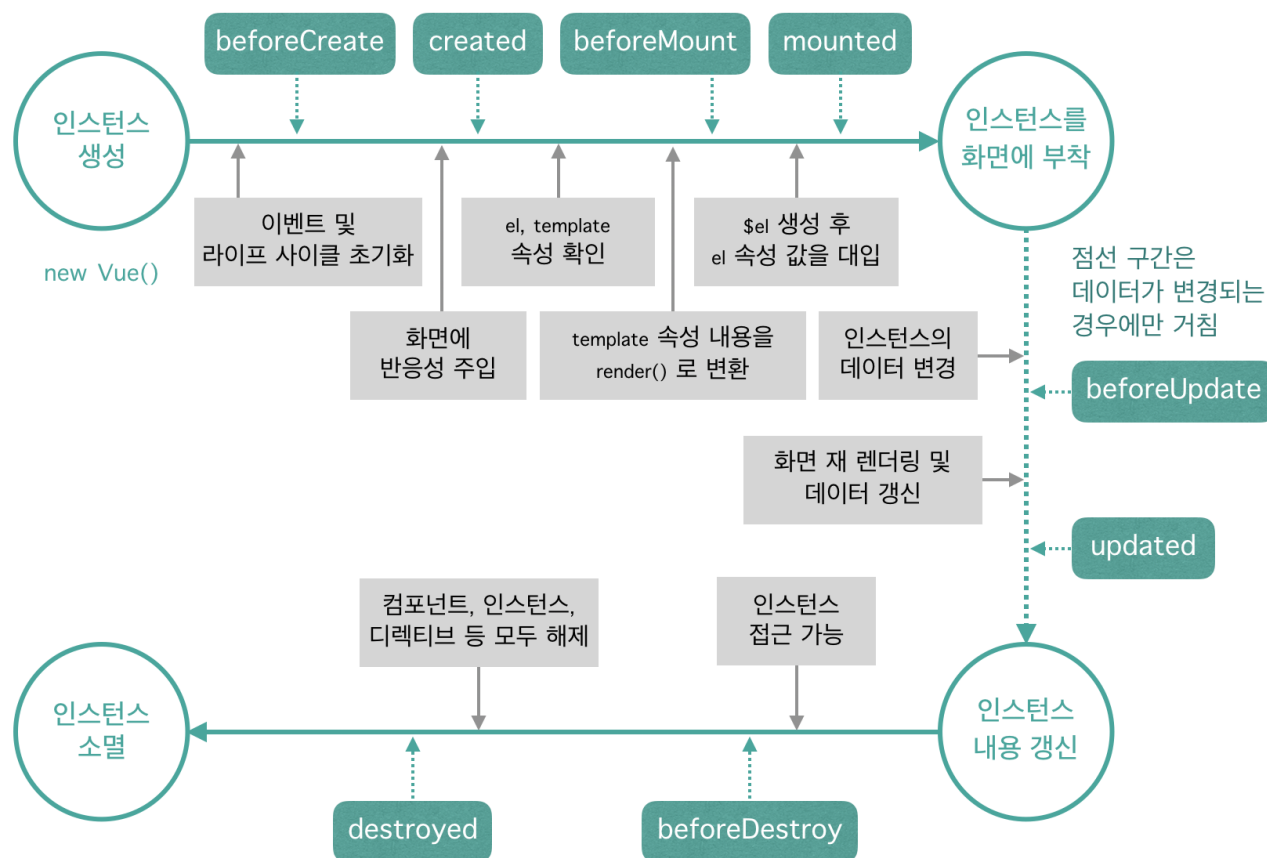
beforeCreate, created

beforeMount, mounted

beforeUpdate, updated

beforeDestroy, destroyed

\* **라이프사이클(Lifecycle)** : Vue 인스턴스가 생성된 후 눈에 보여지고, 사라지기까지의 단계




**Vue 컴포넌트(Components)** : 화면의 영역을 일정한 단위로 쪼개어 재사용 가능한 형태로 관리하는 것

```
<template>
  <div id="app">
    <my-component></my-component>
  </div>
</template>
```

```
<script>
  new Vue({
    el: "#app",
    // 컴포넌트 등록 코드
    components: {
      // '컴포넌트 이름': 컴포넌트 내용
      "my-component": {
        template: "<div>A custom component!</div>"
      }
    }
  });
</script>
```

### 전역 컴포넌트



```
<script>
  Vue.component('my-component', {
    // 컴포넌트 내용
    template: '',
    ...
  })
</script>
```

### 지역 컴포넌트



```
<script>
  var cmp = {
    // 컴포넌트 내용
    template: '',
    ...
  }

  new Vue({
    components: {
      'my-cmp' : cmp
    }
  })
</script>
```

# Vue 라우터(Routers) : 뷰에서 사용되는 라우팅 라이브러리

## 설치방법

```
<script src="https://unpkg.com/vue-router/dist/vue-router.js"></script>
```

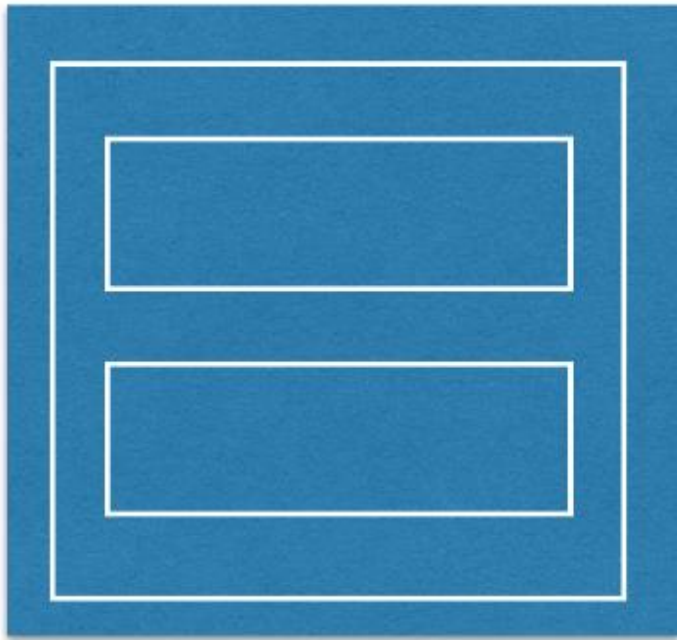
CDN 방식

```
npm install vue-router --save
```

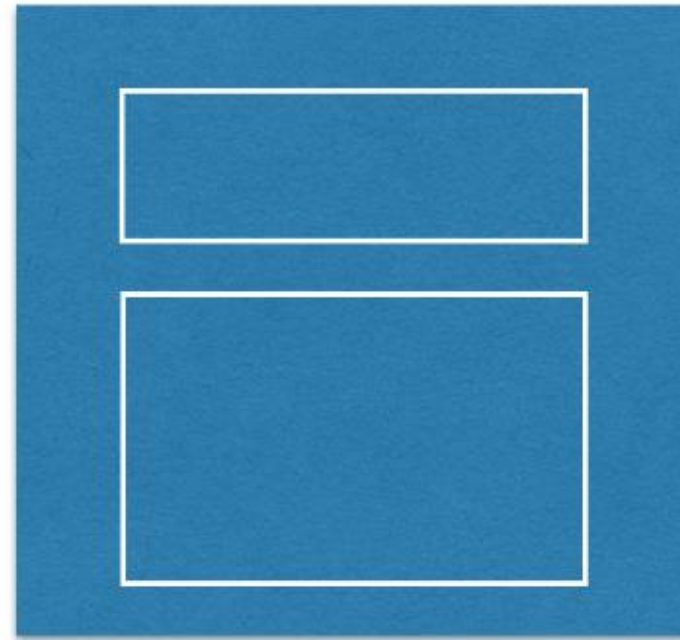
NPM 방식

**Nested Routers** : 라우터로 화면을 이동에서 하위 컴포넌트를 표시 할 때 사용

Nested Routes



Named View



### Nested Routers : 라우터로 화면을 이동에서 하위 컴포넌트를 표시 할 때 사용

```

<!-- localhost:5000 -->
<div id="app">
  <router-view></router-view>
</div>

<!-- localhost:5000/home -->
<div>
  <p>Main Component rendered</p>
  <app-header></app-header>
</div>

```

.vue 파일

```

{
  path : '/home',
  component: Main,
  children: [
    {
      path: '/',
      component: AppHeader
    },
    {
      path: '/list',
      component: List
    },
  ],
}

```

index.js 파일

**Named Views** : 특정 URL로 이동했을 때 여러 개의 컴포넌트를 동시에 표시할 수 있는 방법

```
<div id="app">
  <router-view name="appHeader"></router-view>
  <router-view></router-view>
  <router-view name="appFooter"></router-view>
</div>
```

.vue 파일

```
{
  path : '/home',
  // Named Router
  components: {
    appHeader: AppHeader,
    default: Body,
    appFooter: AppFooter
  }
},
```

index.js 파일



**Axios** : Vue에서 가장 많이 사용되는 HTTP 통신 라이브러리

### 설치방법


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

CDN 방식

```
npm install axios
```

NPM 방식

### 사용예제



```
methods: {  
  fetchData: function() {  
    axios.get('URL 주소');  
  }  
}
```

# Vue 템플릿(Template) : 템플릿이란 뷰로 화면을 조작하기 위해 제공되는 문법 (데이터바인딩 + 디렉티브)

## 데이터 바인딩(Data Binding)

콧수염 문법인 “{{ }}”를 활용하여 인스턴스의 data, computed, props 속성 연결을 가능하게 한다.

```
<div>{{ str }}</div>
<div>{{ number + 1 }}</div>
<div>{{ message.split('').reverse().join('') }}</div>
```

## 디렉티브(Directive)

HTML 태그 속성에 v- 접두사가 붙은 특별한 속성으로 화면의 DOM 조작을 쉽게 할 수 있는 문법을 제공한다.

```
<!-- seen의 진위 값에 따라 p 태그가 화면에 표시 또는 미표시 -->
<p v-if="seen">Now you see me</p>
<!-- 화면에 a 태그를 표시하는 시점에 뷰 인스턴스의 url 값을 href에 대입 -->
<a v-bind:href="url"></a>
<!-- 버튼에 클릭 이벤트가 발생했을 때 doSomething이라는 메서드를 실행 -->
<button v-on:click="doSomething"></button>
```