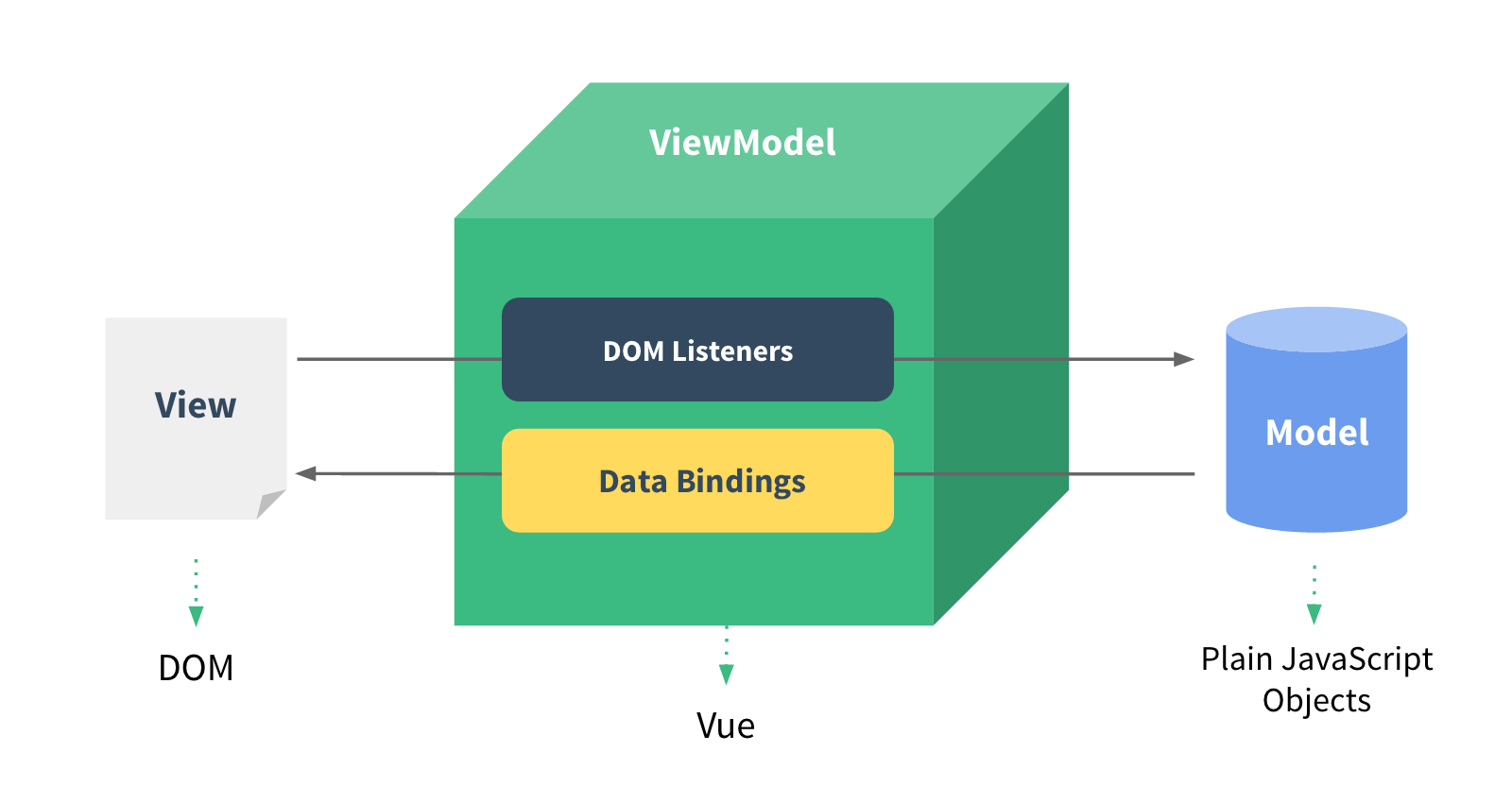
1. **Vue.js란 무엇인가?**

MVVM 패턴의 ViewModel 레이어에 해당하는 화면단 라이브러리



* **데이터 바인딩**과 **화면 단위를 컴포넌트 형태로 제공하며, 관련 API 를 지원**하는데에 궁극적인 목적이 있음
* Angular에서 지원하는 **양방향 데이터 바인딩** 을 동일하게 제공
* 하지만 **컴포넌트 간 통신**의 기본 골격은 React의 **단방향 데이터 흐름(부모 -> 자식)**을 사용
* 다른 프런트엔드 프레임워크(Angular, React)와 [비교](https://vuejs.org/v2/guide/comparison.html)했을 때 상대적으로 가볍고 빠름.
* 문법이 단순하고 간결하여 **초기 학습 비용이 낮고 누구나 쉽게 접근 가능**

1. **MVVM 패턴이란?**

[위키](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel)에 명시된 것처럼, Backend 로직과 Client 의 마크업 & 데이터 표현단을 분리하기 위한 구조로 전통적인 MVC 패턴의 방식에서 기인하였다. 간단하게 생각해서 화면 앞단의 회면 동작 관련 로직과 뒷단의 DB 데이터 처리 및 서버 로직을 분리하고, 뒷단에서 넘어온 데이터를 Model 에 담아 View 로 넘어주는 중간 지점이라고 보면 되겠다.



1. **Vue.js 시작하기**

다른 주요 프런트엔드 프레임워크(Angular, React)와 비교했을 때 뷰위 가장 큰 강점은 바로 시작하기가 정말 쉽다는 점이다.

*<!DOCTYPE html>*

<html>

<head>

<title>Vue.js Sample</title>

</head>

<body>

<div *id*="app">

{{ message }}

</div>

<script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

<script>

new Vue({

el: "#app",

data: {

message: "Hello Vue.js!"

}

});

</script>

</body>

</html>

CDN으로 코드 땡겨오고 바로 Vue 인스턴스를 하나 생성하여 간단한 페이지를 만들어보았다. 기존의 구현된 시스템에 적용하기도 훨씬 수월할 것으로 보인다.

1. **Vue Instance**

인스턴스는 Vue.js로 화면을 개발하기 위해 꼭 생성해야 하는 필수 단위이다.

**Vue Instance 생성자**

Vue 생성자 함수를 이용하여 인스턴스를 생성하는 방법은 아래와 같다.

new Vue({

*// instance option properties*

});

Vue 객체를 생성할 때 아래와 같이 *data, template, el, methods, life cycle hook* 등의 **인스턴스 옵션 속성**을 포함할 수 있다.

new Vue({

*// instance option properties*

template: "",

el: "",

methods: {}

*// ...*

});

**Vue Instance 라이프싸이클 초기화**

인스턴스가 생성될 때 아래의 초기화 작업을 수행한다.

* 데이터 관찰
* 템플릿 컴파일
* DOM 에 객체 연결
* 데이터 변경시 DOM 업데이트

이 초기화 작업 외에도 개발자가 의도하는 커스텀 로직을 아래와 같이 추가할 수 있다.

new Vue({

data: {

a: 1

},

created: function() {

*// this 는 vm 을 가리킴*

console.log("a is: " + this.a);

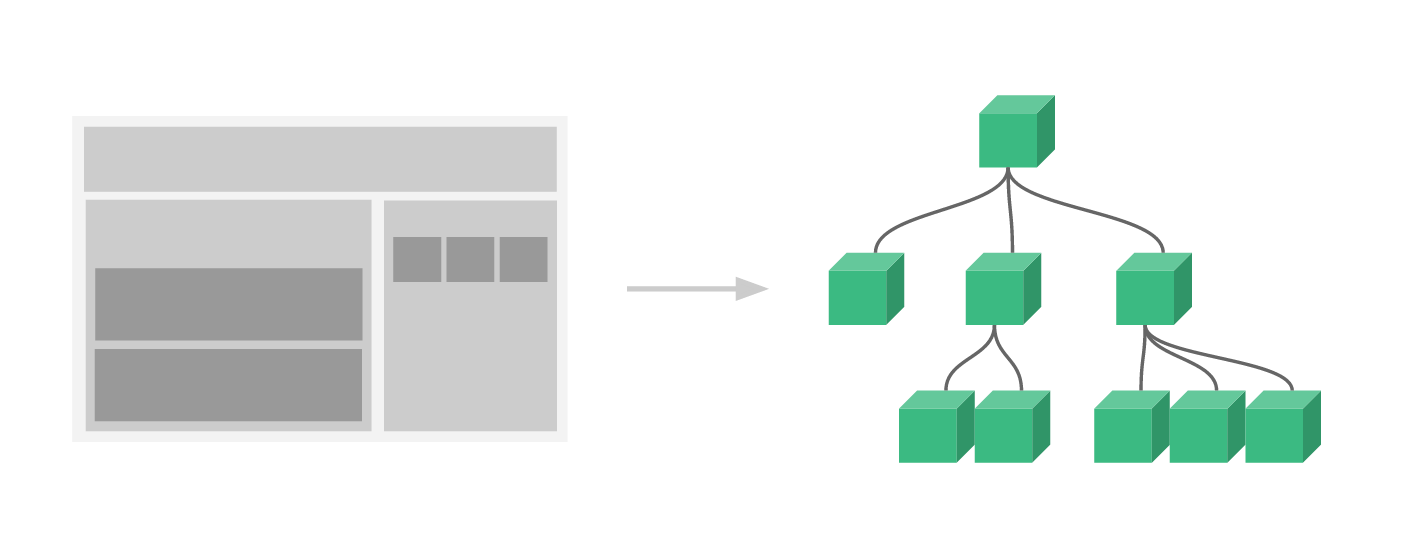
}

});

위 created 이외에도 라이프싸이클 단계에 따라 mounted, updated, destroyed 등을 사용할 수 있다. 이 라이프싸이클 초기화 메서드로 커스텀 로직을 수행하기 때문에 뷰에서는 따로 Controller를 갖고 있지 않다.

1. **Vue Components**

화면의 영역을 일정한 단위로 쪼개어 재활용 가능한 형태로 관리하는 것이 컴포넌트



컴포넌트 등록은 아래와 같은 코드로 생성 가능하다.

<div *id*="app">

<my-component></my-component>

</div>

new Vue({

el: "#app",

*// 컴포넌트 등록 코드*

components: {

*// '컴포넌트 이름': 컴포넌트 내용*

"my-component": {

template: "<div>A custom component!</div>"

}

}

});

**Global or Local Component**

아래의 컴포넌트 등록 방식은 전역 컴포넌트 등록 방식이다.

Vue.component('my-component', {

*// 컴포넌트 내용*

template: '',

...

})

아래와 같이 지역 컴포넌트로도 등록할 수 있다.

var cmp = {

*// 컴포넌트 내용*

template: '',

...

}

new Vue({

components: {

'my-cmp' : cmp

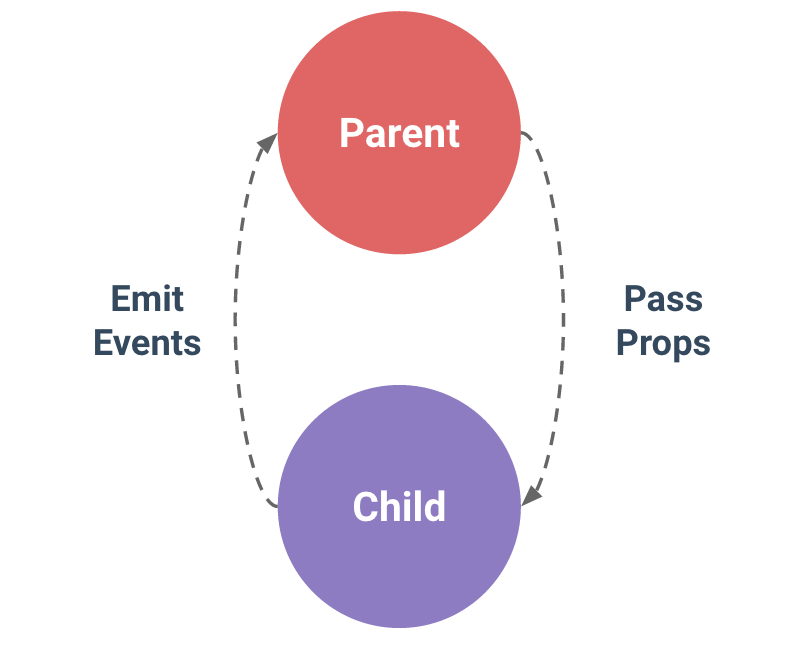
}

})

**부모와 자식 컴포넌트 관계**

컴포넌트 관계도에서 상-하 관계에 있는 컴포넌트의 통신은

* 위에서 아래로는 데이터(props)를 내리고
* 아래에서 위로는 이벤트를 올린다(event emit)



**Props**

프롭스는 상위 컴포넌트에서 하위 컴포넌트로 내리는 데이터 속성을 의미한다. 이렇게 하는 이유는 모든 컴포넌트가 각 컴포넌트 자체의 스코프를 갖고 있어 다른 컴포넌트의 값을 바로 참조할 수 없기 때문이다.

*<!-- 상위 컴포넌트 -->*

<div *id*="app">

*<!-- 하위 컴포넌트에 상위 컴포넌트가 갖고 있는 message를 전달함 -->*

<child-component *v-bind:propsdata*="message"></child-component>

</div>

*// 하위 컴포넌트*

Vue.component("child-component", {

*// 상위 컴포넌트의 data 속성인 message를 propsdata라는 속성으로 넘겨받음*

props: ["propsdata"],

template: '<p>{{ propsdata }}</p>'

});

*// 상위 컴포넌트*

var app = new Vue({

el: "#app",

data: {

message: "Hello Vue! from Parent Component"

}

});

**주의할 점: props 변수 명을 카멜 기법(aBow)으로 정의하면 html 태그에서 사용할 때는 케밥 기법(-)으로 선언해야 한다. 아래는 만약 프롭스 속성 명을 카멜 기법인 passedData로 선언했을 때의 주의 메시지**

props-parsing-rules-between-components

**같은 레벨의 컴포넌트 간 통신**

동일한 상위 컴포넌트를 가진 하위 컴포넌트들 간의 통신은 아래와 같이 해야 한다.

* Child(하위) -> Parent(상위) -> Children(하위 2개)

**참고 : 컴포넌트 간의 직접적인 통신은 불가능하도록 되어 있는게 Vue 의 기본 구조**

**Event Bus**

상위 - 하위 관계가 아닌 컴포넌트 간의 통신을 위해 **Event Bus**를 활용할 수 있다.

Event Bus를 사용하기 위해 새로운 뷰 인스턴스를 아래와 같이 생성한다.

*// 화면 개발을 위한 인스턴스와 다른 별도의 인스턴스를 생성하여 활용*

var eventBus = new Vue();

new Vue({

*// ...*

});

이벤트를 발생시킬 컴포넌트에서 $emit() 호출

eventBus.$emit("refresh", 10);

이벤트를 받을 컴포넌트에서 $on() 이벤트 수신

*// 이벤트 버스 이벤트는 일반적으로 라이프 사이클 함수에서 수신*

new Vue({

created: function() {

eventBus.$on("refresh", function(data) {

console.log(data); *// 10*

});

}

});

만약, eventBus의 콜백 함수 안에서 해당 컴포넌트의 메서드를 참고하려면 vm 사용

new Vue({

methods: {

callAnyMethod() {

*// ...*

}

},

created() {

var vm = this;

eventBus.$on("refresh", function(data) {

console.log(this); *// 여기서의 this는 이벤트 버스용 인스턴스를 가리킴*

vm.callAnyMethod(); *// vm은 현재 인스턴스를 가리킴*

});

}

});

1. **Vue Routers**

뷰를 이용하여 싱글 페이지 애플리케이션을 제작할 때 유용한 [라우팅 라이브러리](https://router.vuejs.org/en/). 뷰 코어 라이브러리와 함께 공식 라이브러리로 지원되고 있다.

설치는 NPM과 CDN 방식 모두 지원한다.

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

npm install vue-router --save

**라우터 특성**

Vue 라우터는 기본적으로 '루트 URL'/#/'라우터 이름'의 구조로 되어 있다.

example.com/#/user

여기서 ‘#’ 값을 제외하고 싶으면 아래와 같이 mode 속성을 추가한다.

new VueRouter({

mode: "history"

});

**Nested Routers**

라우터로 화면을 이동할 때 네스티드 라우터를 이용하여 지정된 하위 컴포넌트를 표시할 수 있다. 이 때 컴포넌트의 구조는 가장 큰 상위의 컴포넌트가 하위의 컴포넌트를 포함하는 Parent - Child 형태와 같다.

*<!-- localhost:5000 -->*

<div *id*="app">

<router-view></router-view>

</div>

*<!-- localhost:5000/home -->*

<div>

<p>Main Component rendered</p>

<app-header></app-header>

</div>

*// 'localhost:5000/home'에 접근하면 Main과 Header 컴포넌트 둘다 표시된다.*

{

path : '/home',

component: Main,

children: [

{

path: '/',

component: AppHeader

},

{

path: '/list',

component: List

},

]

}

**Named Views**

특정 URL로 이동했을 때 여러 개의 컴포넌트를 동시에 표시할 수 있는 방법이다.

<div *id*="app">

<router-view *name*="appHeader"></router-view>

<router-view></router-view>

<router-view *name*="appFooter"></router-view>

</div>

{

path : '/home',

*// Named Router*

components: {

appHeader: AppHeader,

default: Body,

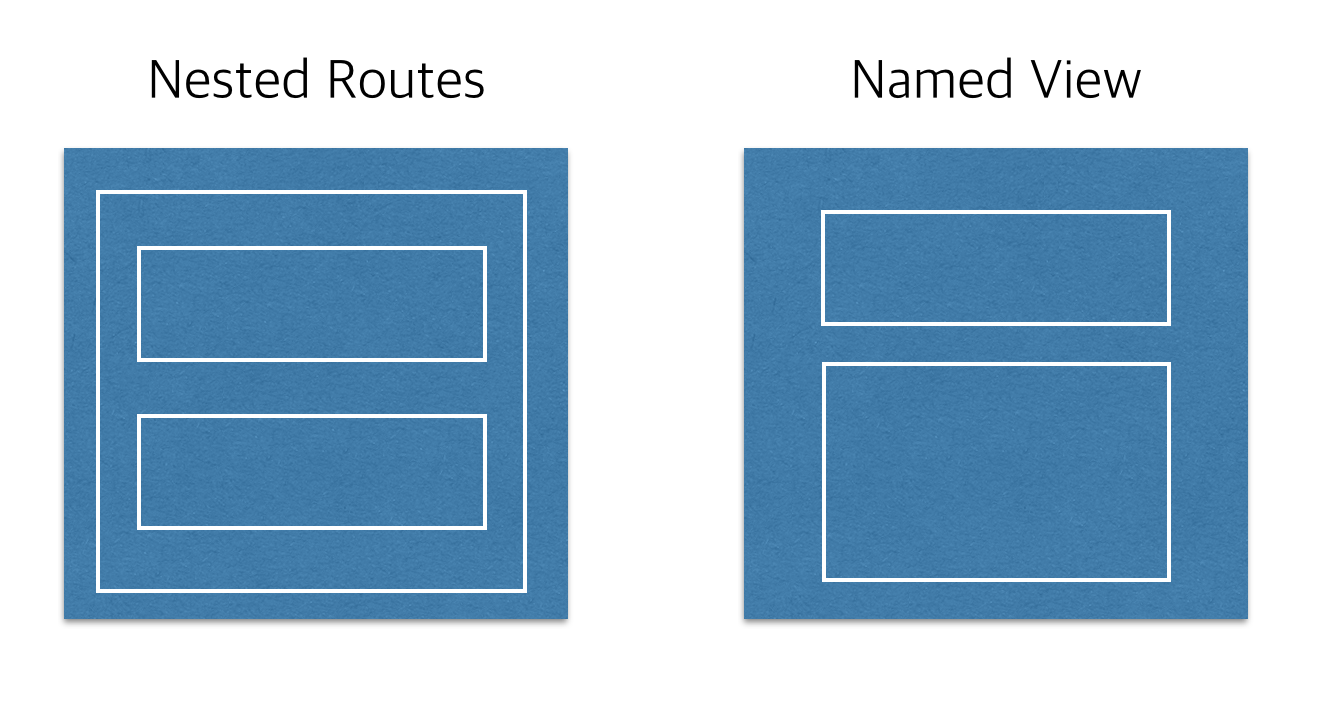
appFooter: AppFooter

}

},

**Nested Router vs Named Views**

* 특정 URL에 지정된 1개의 컴포넌트가 여러 개의 하위 컴포넌트를 갖는 것을 Nested Router
* 특정 URL에 여러 개의 컴포넌트를 영역 별로 지정하여 렌더링 하는 것을 Named View



1. **Axios**

Vue에서 가장 많이 사용하는 [HTTP 통신 라이브러리](https://github.com/axios/axios)이다. CDN과 NPM 설치 방식을 모두 지원하며 사용하기 좋은 속성과 API가 많다. 무엇보다도 [Promise](https://joshua1988.github.io/web-development/javascript/promise-for-beginners/) 기반이라 코드를 간결하게 작성하기 용이하다.

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

npm install axios

설치 후 컴포넌트에서 아래와 같은 코드로 사용한다.

methods: {

fetchData: function() {

axios.get('URL 주소');

}

}

1. **Vue Template**

템플릿이란 뷰로 화면을 조작하기 위해 제공되는 문법이다. 뷰 인스턴스에서 관리하는 데이터를 화면에 연결하는 데이터 바인딩과 화면의 조작을 편하게 할 수 있는 디렉티브로 나뉜다.

**Data Binding**

콧수염 문법인 “{{ }}”를 활용하여 인스턴스의 data, computed, props 속성을 연결할 수 있다. 그리고 간단한 자바스크립트 표현식도 화면에 표시할 수 있다.

<div>{{ str }}</div>

<div>{{ number + 1 }}</div>

<div>{{ message.split('').reverse().join('') }}</div>

**Directive**

HTML 태그의 속성에 v- 접두사가 붙은 특별한 속성으로 화면의 DOM 조작을 쉽게할 수 있는 문법들을 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>

**Filters**

화면에 표시되는 텍스트의 형식을 편하게 바꿀 수 있도록 고안된 기능이며 | 을 이용하여 여러 개의 필터를 적용할 수 있다.

*<!-- message 값에 capitalize 필터를 적용하여 첫 글자를 대문자로 변경 -->*

{{ message | capitalize }}

HTML

Copy

new Vue({

filters: {

capitalize: function(value) {

if (!value) return "";

value = value.toString();

return value.charAt(0).toUpperCase() + value.slice(1);

}

}

});

1. **Single File Component**

특정 화면 영역의 HTML, CSS, JS 코드를 한 파일에서 관리할 수 있는 방법. 파일 확장자는 vue이며 HTML 파일에서 뷰 개발을 진행했을 때의 한계점을 극복할 수 있는 방법이기도 하다. 한계점은 아래와 같다.

1. **모든 컴포넌트에 고유의 이름**을 붙여야 함
2. js 파일에서 template 안의 html 의 **문법 강조가 되지 않음**
3. js 파일상에서 **css 스타일링 작업이 거의 불가**
4. ES5 를 이용하여 계속 앱을 작성할 경우 **Babel 빌드가 지원되지 않음**

싱글 파일 컴포넌트로 개발하려면 Webpack과 같은 번들링 도구가 필요하다. 싱글 파일 컴포넌트의 기본 골격은 다음과 같다.

<template>

*<!-- HTML -->*

</template>

<script>

*// Javascript*

</script>

<style>

*/\* CSS \*/*

</style>

1. **Vue Loader**

싱글 파일 컴포넌트를 브라우저에서 실행할 수 있게 자바스크립트 파일로 변환해주는 웹팩 로더. 뷰 로더를 사용하면 다음과 같은 장점이 있다.

1. ES6 지원
2. <style> 과 <template> 에 대한 각각의 웹팩 로더 지원. ex) sass, jade
3. 각 .vue 컴포넌트의 스코프로 좁힌 css 스타일링 지원
4. 웹팩의 모듈 번들링에 대한 지원과 의존성 관리가 제공
5. 개발 시 Hot Module Replacement(HMR) 지원
6. **Vue CLI**

뷰 프로젝트를 생성하기 위한 명령어 도구이다. 아래의 명령어로 CLI를 시스템 레벨에 설치할 수 있다. 그리고 [CLI를 설치](https://cli.vuejs.org/guide/installation.html)하기 위해서는 [Node.js LTS 버전](https://nodejs.org/en/)이 설치되어 있어야 한다.

npm install -g @vue/cli

CLI가 설치되고 나면 아래의 명령어로 프로젝트를 생성할 수 있다.

vue create 프로젝트 이름

명령어를 입력하고 나면 Preset을 선택하라고 나오는데 Default를 선택하면 된다. 프로젝트가 생성되면 콘솔에 아래와 같은 형식으로 안내된 명령어를 입력하여 웹 애플리케이션을 실행할 수 있다.

cd 프로젝트 폴더 이름

npm run serve

1. **Virtual DOM in Vue.js**

리액트와 마찬가지로 뷰도 Virtual DOM을 사용한다. Virtual DOM은 화면을 조작하기 위한 기반 기술이다. 화면의 DOM을 추가하거나 삭제하는 등의 변경이 일어날 때 마다 화면을 다시 그리는 것이 아니라, 자바스크립트 객체로 DOM의 모양을 잡아 놓고 화면의 렌더링 횟수를 최소화하여 브라우저의 부하를 줄인다.