

# Vue.js

# WIFI 비번 : 20160919

교재/자료 공유용 오픈카톡방 : [open.kakao.com/o/](https://open.kakao.com/o/)  
(카톡에 입력하지말고 브라우저 주소창에 입력)

- Node.js 설치
- VS code 에디터 준비
- Yarn 설치
- PC용 카톡

# Vue.js

## 사용하는 이유

1. 데이터바인딩/ 자료 iteration/ DOM 조작이 매우쉬워짐
2. 요소를 실시간으로 반응시켜야할 때
3. Component 기반 개발방식을 적용할 때
4. SPA를 만들 때



The Progressive  
JavaScript Framework





WHY VUE.JS?



GET STARTED





GITHUB



Bangalore 




LOGIN / SIGNUP




Home > Packages


PackagesFurnitureAppliancesElectronicsBikesMen's Apparel

 Filters


Reset

Living Room 


Choose by Room Type




Bedroom  
7 Packages




Living Room  
10 Packages




Office & Study  
1 Packages




Dining  
1 Packages




Barney Living Room


Starts From ₹987/mo 3 Items



Heathcliff Living Room

Starts From ₹1297/mo 3 Items

<https://www.rentomojo.com/bangalore/packages/living-room-furniture-on-rent>

 **rentomojo**


Bangalore ▾


Categories ▾


🔍

🛒

[LOGIN / SIGNUP](#)







Barney Living Room

3 Products | 2 Essentials

Make your RMI (Rental monthly Installments) plan

Choose a plan without worry. You can close or extend it anytime

3 Mo

6 Mo

12 Mo

18 Mo


24 Mo


36 Mo


Retail Price	EMI for 12 mo	RMI for 12 mo
₹21000	₹1894 / mo	₹1514 / mo

Refundable Deposit

₹2986/-

 Book your plan

 View Images



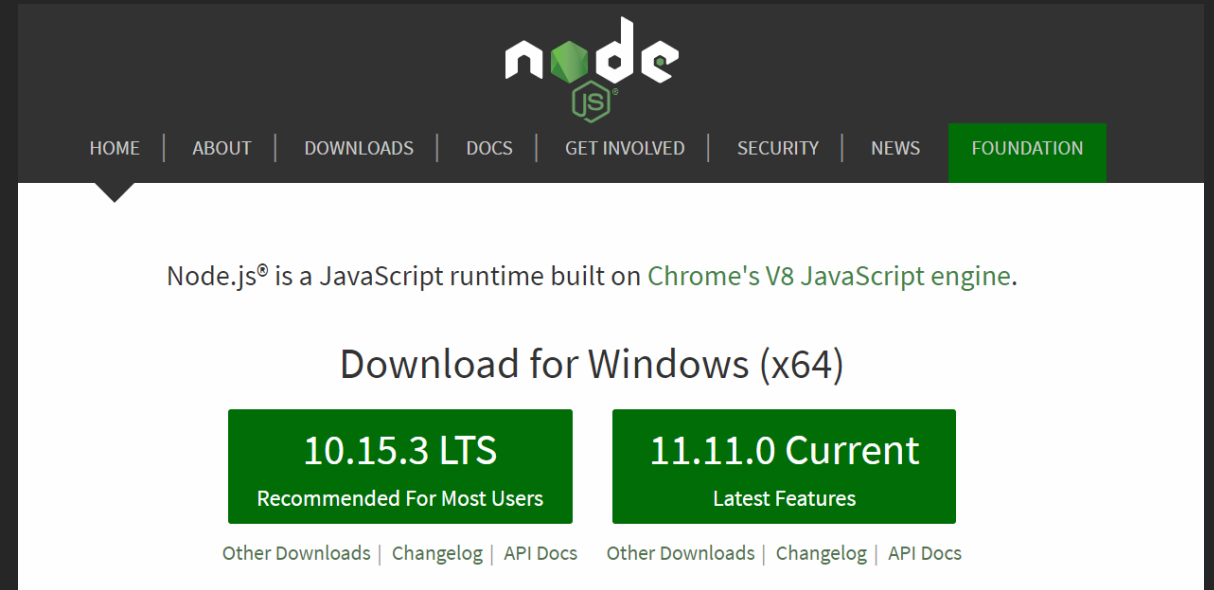
## 만들어볼 내용

1. 상품진열
2. 가격정렬, 필터 버튼
3. 상세페이지 모달창
4. 가격슬라이더 기능 등

개발환경셋팅 1. Node.js 설치

개발환경셋팅 2. VS code 에디터 설치

개발환경셋팅 3. yarn for windows 설치  
(혹은 npm install -g yarn 터미널에 입력)



VS code **오픈**시

- 작업용 폴더 통째로 열기

VS code에서 **터미널** 사용법

- 메뉴에서 Terminal -> New Terminal 선택

Vscode 에디터 설치하면 좋은 **부가기능**

Vetur

(Vue쓰려면 필수)

Vue 2 snippets

HTML CSS support



# Vue 설치 & 프로젝트 생성

vue CLI 설치하는 터미널에서

```
yarn global add @vue/cli
```

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

-g 는 다른 프로젝트 폴더에서도 vue/cli 사용하기 위해  
글로벌하게 설치할 때 사용

프로젝트생성 :

1. 작업용 폴더 만들고 에디터에서 열기
2. 터미널 열기
3. 프로젝트 생성 명령어 입력

```
vue create vuedongsan
```

4. 생성된 vuedongsan 하위 폴더를 다시 에디터로 오픈

서버실행 (live update)

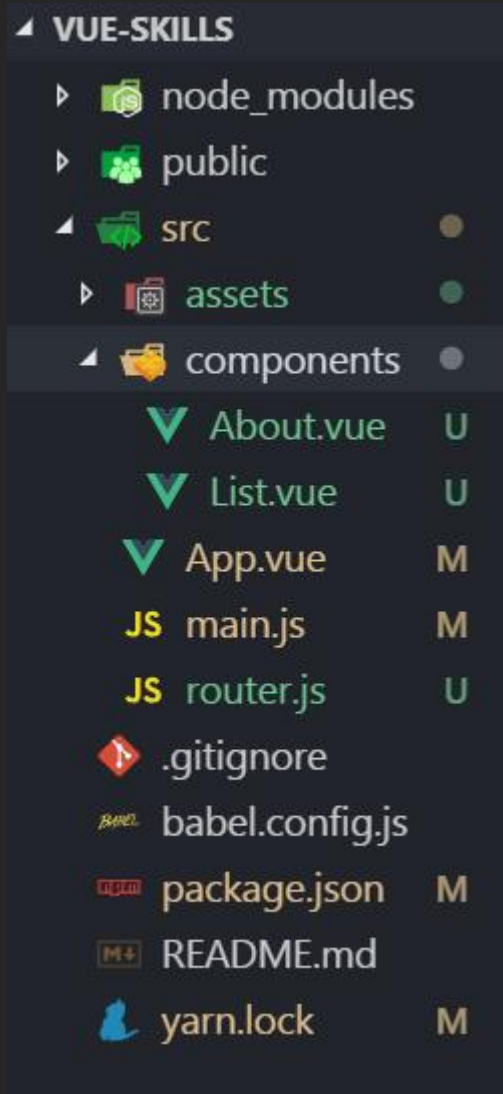
```
yarn serve 또는 npm run serve
```

배포용 파일 생성

```
yarn build 또는 npm run build
```

터미널이 싫거나 명령어 까먹으면

```
vue ui
```



node\_modules : 필요한 npm 패키지담기는 곳

assets폴더 : 이미지 등 필요한 파일들 담는 곳

public폴더 : 웹사이트 발행시에도 남길 파일들 담는 곳

components : 본격적인 vue 컴포넌트파일 생성하는 곳

App.vue : 메인 컴포넌트

index.html : 메인 HTML 페이지 (App.vue를 이곳에 붙여넣음)

## 방금만든 프로젝트 설명 :

App.vue

(우리가 만든 웹앱)



main.js

(렌더링 항목/위치결정)

index.html

(메인페이지에 App.vue 꽂아넣음)

```
<template>
  <div>
    {{ message }}
  </div>
</template>

<script>
export default {
}
</script>

<style>

</style>
```

기본적인 .vue 파일 구성:

1. template에서 html 구조 작성
2. script란에서 각종 데이터 바인딩/ 기능개발
3. style란에서 css 스타일 적용

## 일반 웹개발 방식 :

- 그냥 HTML 파일에 썬코딩 or 서버가 있으면 서버언어로 HTML 생성

## 모던 웹개발 방식 :

- 서버데이터를 수신해와서 자바스크립트로 HTML 생성

## Vue/React 방식 :

- 서버데이터를 수신해와서 Vue 문법으로 '쉽게' HTML 생성

## Vue로 HTML 쉽게 만드는 방법(외울거)

1. HTML내용 데이터바인딩
2. HTML속성 데이터바인딩
3. HTML을 if문 써서 넣기
4. HTML을 for문 돌려서 넣기
5. HTML에 이벤트리스너 달기 (버튼에 기능만들기)
6. 역방향 데이터바인딩 (input 등에 글자를 넣으면 자바스크립트 데이터가 바뀔다능)

```
<div>
  {{ 메시지 }}
</div>
```

```
export default {
  name: 'app', (디버깅용 컴포넌트 이름)
  data() {
    return {
      메시지: '안녕',
    }
  }
}
```

옛 자바스크립트 : document.getElementById('어쩌구').innerHTML = "안녕"

## 1. 내용 데이터바인딩

{{콧수염}} 으로 간단히 **HTML 내부 글자**를 데이터바인딩 할 수 있음

{{콧수염}}을 넣은다음 원하는 데이터 명을 적어주면 HTML 내용이 데이터로 변경됩니다.

data ()

- 본격적으로 데이터가 담기는 공간.
- return문안에 데이터를 object 형식으로 담아야함.



```
<div>
  <p v-bind:style="테스트">{{ 메시지 }} </p>
</div>
```

```
export default {
  data() {
    return {
      테스트: 'font-size: 26px',
      메시지: '안녕'
    }
  }
}
```

## 2. 속성 데이터바인딩

**v-bind:** 라는 수식어로 속성을 데이터바인딩할 수 있음

예제는 data에 들어있는 '테스트'라는 값을 가져와서 style 속성안에 집어넣는 예제

{{콧수염}}과 기능이 동일합니다.

참고: 원래 style 같은건 {object} 형식으로 써넣는게 좋음    { backgroundImage : 'url()' }

```
<div>
  <p v-if="보여주까여"> hello </p>
</div>
```

```
export default {
  data() {
    return {
      보여주까여: true,
    }
  }
}
```

옛 자바스크립트 : if ( ) { }

### 3. HTML if문


조건에 따라서 HTML을  
hide/show 할 수 있음

HTML 내에 `v-if=""` 라는 속성을 사용  
합니다. if문 내의 조건식이 true일 경  
우, HTML 요소를 보여줍니다.

예제는 간단한 true/false값만 입력  
해서 글자보여주는 예제

```
<ul>
  <li v-for="상품 in 상품들">{{ 상품 }} </li>
</ul>
```

```
export default {
  data() {
    return {
      상품들: ['갤럭시', '아이폰', 'LG']
    }
  }
}
```



## 4. HTML for문

조건에 따라서 HTML을 원하는  
개수만큼 반복 & iterate 가능

v-for="아이템 in 데이터" 라는 문법  
을 사용합니다. 데이터 array의 길이  
만큼 for문을 반복해서 HTML을 생성  
합니다.

참고) for 반복문을 쓰면 꼭 v-bind:key="" 도 함께 사용해야합니다.

```
<p>{{ message }}</p>
<button v-on:click="증가함수">버튼임</button>
```

```
export default {
  data() {
    return {
      message: 1
    }
  },
  methods : {
    증가함수(){
      this.message++
    }
  }
}
```

## 5. 이벤트리스너달기

버튼누르면 뭔가 동작하게 기능을 추가하고싶다면..

옛 자바스크립트처럼 `v-on:click`, `v-on:submit` 이런걸 붙인 후 안에 함수를 넣어주시면 됩니다.

- 함수만들기 귀찮으면 생자바스크립트 넣으셔도 좋습니다
- 함수들은 `method {}` object안에 보관하시면 됩니다.

```
<p>{{ 메시지 }}</p>  
<input v-model="메시지">
```

```
export default {  
  data() {  
    return {  
      메시지: " "  
    }  
  }  
}
```

## 6. 역방향 데이터바인딩

HTML데이터가 JavaScript 데이터를 변경할 수도 있습니다.

input같은 곳에 **v-model**이라고 추가하면 사용자가 입력한 데이터로 data()안의 값이 변경됩니다.

## Vue 맛보기 프로젝트 :

### 데이터에 의한 HTML내용 변경과 CSS 디자인

```
<div>  
  <p>임시 텍스트</p>  
</div>
```

return {} 내부에 있는 메시지와 클래스명을 이용해 '임시텍스트' 글자를 바꾸고 스타일도 입혀보세요.

메시지 : '안녕하세요' ,  
클래스명 : 'text-red'

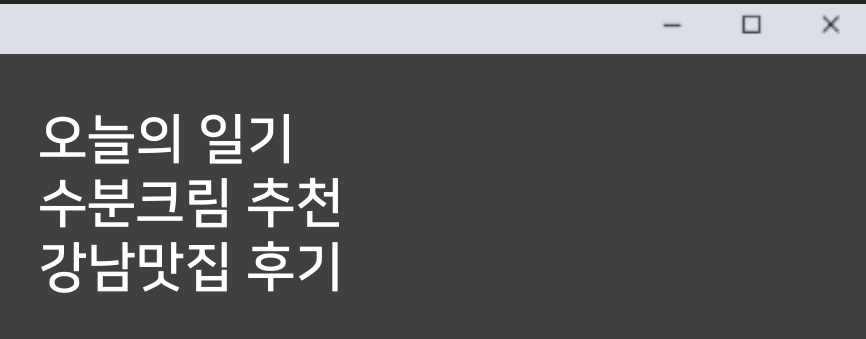
## Vue 맛보기 프로젝트 :

### 블로그 글 목록 레이아웃 만들기

```
<div>  
  <h4>글 제목</h4>  
</div>
```

return {} 내부에 있는 posts 데이터를 가져다가 블로그 글 목록 div박스 3개를 만들어보세요.

```
posts : ['오늘의 일기', '수분크림 추천', '강남맛집 후기']
```

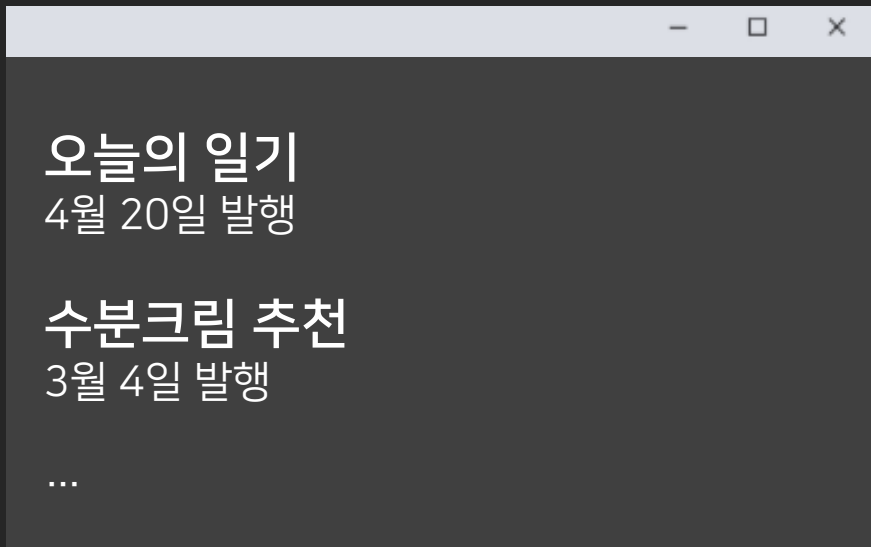


오늘의 일기  
수분크림 추천  
강남맛집 후기

## Vue 맛보기 프로젝트 :

### 블로그 글 목록 레이아웃 만들기2

```
<div>  
  <h4>글 제목</h4>  
  <p>5월 25일 발행</p>  
</div>
```



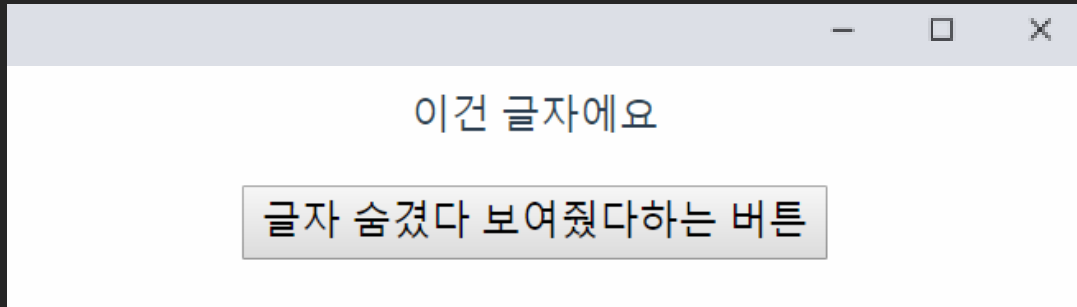
이번엔 Object 자료형을 담은 Array입니다.  
title과 date를 각각 바인딩 해봅시다.

```
posts : [  
  {  
    title : '오늘의 일기',  
    date : '4월 20일'  
  },  
  {  
    title : '수분크림 추천',  
    date : '3월 4일'  
  },  
  {  
    title : '강남 맛집 후기',  
    date : '2월 22일'  
  }  
],
```



## Vue 맛보기 프로젝트 :

### 글자 숨겼다가 보여줬다가 하는 버튼 만들기



썬 자바스크립트로는.. 버튼에다가 이벤트 리스너 개발하지 않았나요?  
이것도 비슷한데, 이벤트리스너의 역할은 HTML 변경이 아닌 data 변경입니다.  
(data가 변경되면 알아서 재-렌더링 해줌)

VueDongsan

원래대로

가격순 정렬

50만원 이하만



Sinrim station 30 meters away  
월 340000 원



Changdong Aurora Bedroom(Queen-size)  
월 450000 원



Geumsan Apartment Flat



Double styled beds Studio Apt

## Project1. 심플 뷰동산앱

- 간단한 컴포넌트만들기
- for 반복문
- Props를 이용한 디스플레이
- 정렬기능
- v-model 바인딩
- 라이프사이클 함수들

프로젝트생성 :

1. 작업용 상위폴더 만들고 에디터에서 열기
2. 에디터 오픈 후 New Terminal 열기
3. 프로젝트 생성 명령어 입력

```
vue create vuedongsan
```

4. 생성된 vuedongsan 폴더를 에디터로 오픈

## 자바스크립트 export / import 기본

```
export default [1, 2, 3, 4, 5]
```

data.js

1. 보통 페이지 마지막 줄에서  
다른 곳에서 쓸 자료/함수를 export

```
import 자료 from './data.js'
```

```
console.log(자료[0])
```

2. 원하는 곳에서 import 해서 사용

- export default는 파일내에서 한번 밖에 못 씀
- export default만 있으면 import시 이름 변경가능

## export를 여러 개 할 경우

```
export let a = 100;  
export function hello() {  
  ~  
}
```

data.js

1. export는 원하는 개수 만큼 자유롭게 쓸 수 있음.

```
import {a, hello} from './data.js'
```

```
console.log(a)
```

2. 그럼 import 할 때 첨부를 원하는 데이터/변수명을 지정해줘야함. (export 할때와 동일한 이름)

# Bootstrap으로 반응형 레이아웃 만들기

## 1. Bootstrap-vue 플러그인 설치

```
yarn add bootstrap-vue
```

```
안되면 npm install bootstrap-vue
```

## 2. main.js에 부트스트랩 css파일 첨부

```
import Vue from 'vue';  
import BootstrapVue from 'bootstrap-vue';  
  
Vue.use(BootstrapVue);  
  
import 'bootstrap/dist/css/bootstrap.css'  
import 'bootstrap-vue/dist/bootstrap-vue.css'
```


# Bootstrap으로 반응형 레이아웃 만들기

## 3. Bootstrap-Vue 또는 Bootstrap 홈페이지에서 원하는 HTML UI 템플릿 복붙


## 4. 목표 : 우측 사진

*VueDongsan*


원래대로
가격순 정렬
50만원 이하만




Sinrim station 30 meters away  
월 340000 원



Changdong Aurora Bedroom(Queen-size)  
월 450000 원



Geumsan Apartment Flat



Double styled beds Studio Apt

# 이미지 첨부방법 (Webpack문법)

1. assets폴더에있다면 src경로로 잘 첨부하세요

```

```

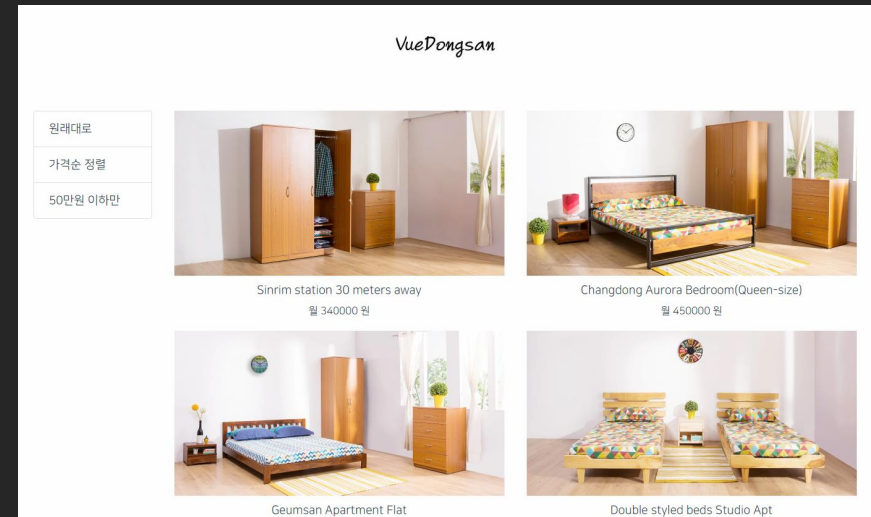
2. require 사용해ダイナ믹하게 첨부하셔도 됩니다.

```

```

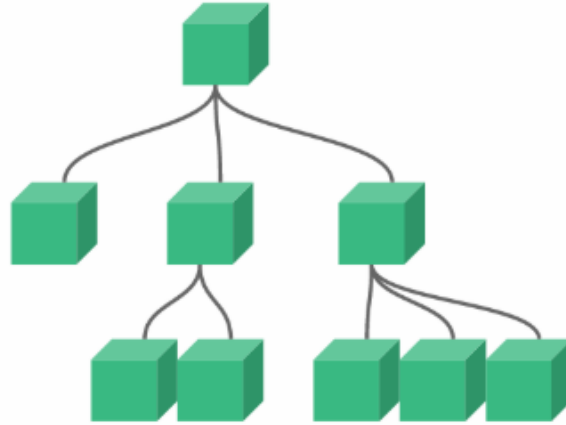
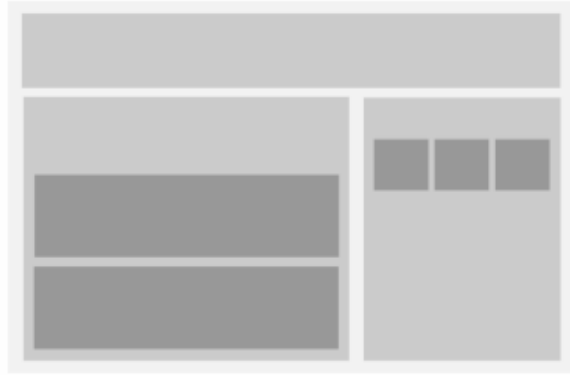
3. import 해서 첨부해도 됩니다

```
import image from './assets/logo.png'
```





# Component 기반 개발



초록색 네모가  
전부 component

웹사이트를 저렇게 Component화 해놓은 뒤,  
페이지마다 component 들을 첨부하는 방식으로 웹개발진행.

# Component 기반 개발

```
<배경>  
  <카드/>  
</배경>
```

Component화 해놓고 원하는 페이지에서 조립하여 웹을 개발합니다.

장점1. 유지보수가 쉬움

장점2. 페이지수가 많은 큰 프로젝트일 때 좋음

장점3. HTML 레이아웃의 쉬운 재사용

HTML 컴포넌트들 만들어놓고 개발하면 좋은 점 :

1. 수정사항 생길 때 수정이 편리합니다.
2. 조립식 개발이 가능해져 큰 프로젝트일 때 좋습니다.
3. 다른곳에서도 쉽게 재사용가능

App.vue

<카드 />

<카드 />

어쩌구.vue

```
<template>
  <div>
    {{ 메시지 }}
  </div>
</template>

<script>
export default {
  name : 어쩌구,
  data() {
  }
}
</script>

<style>

</style>
```

## (중요) 컴포넌트 만드는 법

1. 어쩌구.vue 파일 만드시면 됩니다.

2. (참고) template 내부는 하나의 div로 싸매셔야 합니다.

App.vue

(더 중요) 컴포넌트 만든거 넣는 법

```
<Card />  
<Card />
```

3. 원하는 곳에 쳐넣기

```
import Card from './components/Card.vue'
```

1. 아까만든 vue파일 import해오기

```
export default {  
  data() {  
  },  
  components : {  
    Card  
  }  
}
```

2. 컴포넌트 이름 등록

# 컴포넌트.vue 만들어 쓰기 요약

## Card.vue

### 1. 여기서 UI 컴포넌트 제작

HTML 만들고  
자바스크립트짜고  
CSS로 꾸미고

## App.vue

### 2. 제작한거 import하고 등록하고 쓰기

1. import Card from '~~~'  
2. components : { Card }  
3. 원하는 곳에 <Card /> 쓰기

Vue 맛보기 프로젝트 :

직접 컴포넌트 만들어서 App.vue에 컴포넌트로 첨부하기

이건 글자예요

글자 숨겼다 보여줬다하는 버튼

대충 이런 UI를 나중에 재사용 할 것 같아서  
컴포넌트화 하기로 합니다.

App.vue에 컴포넌트 형식으로 첨부해보십시오

LargeModal.vue

PostList.vue

UI 요소를 만들어놓고 간편하게 재사용하고싶다면  
컴포넌트로 만들자

1. Vue파일 이름지을 땐 CamelCase
2. 이름지을 땐 단어 여러개 쓰는게 좋음 (HTML태그 이름과 중첩되면 에러남)
3. 컴포넌트에 넣어야할 이미지 등이 많이 있을 경우 하위 폴더를 하나 만들고 다 담으셈
4. Vue파일 내의 <template> 내부는 항상 큰 <div>로 싸매야함



상품진열용 **카드** 컴포넌트를 만들어보도록 합시다.  
생 HTML로 안짜고 굳이 컴포넌트를 만드는 이유는..?

<Card />

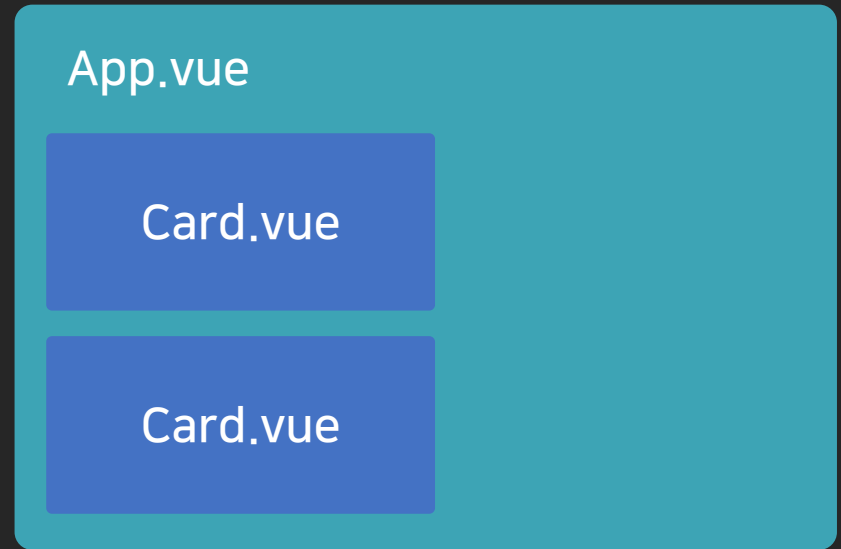
<Card />

<Card />

<Card />

<Card />

<Card />



```
export default [{
  title: "Sinrim station 30 meters away",
  image: "https://p.rmjo.in/moodShot/qqnt2gl5-1024x512.jpg",
  caption: "18년 신축공사한 남향 원룸, 공기청정기 제공 ☀️",
  price: 340000
},
{
  title: "Changdong Aurora Bedroom(Queen-size)",
  image: "https://p.rmjo.in/moodShot/p85wsl40-1024x512.jpg",
  caption: "침실만 따로 있는 공용 세어하우스입니다. 최대 2인 가능",
  price: 450000
},
{
  title: "Geumsan Apartment Flat",
  image: "https://p.rmjo.in/moodShot/h51acr02-1024x512.jpg",
  caption: "금산오거리 역세권 아파트입니다. 애완동물 불가 🐾",
  price: 780000
},
{
  title: "Double styled beds Studio Apt",
  image: "https://p.rmjo.in/moodShot/c3ii5yl7-1024x512.jpg",
  caption: "천호동인근 2인용 원룸입니다. 전세 전환가능",
  price: 550000
},
{
  title: "MyeongIl Apartment flat",
  image: "https://p.rmjo.in/moodShot/6ouap4qn-1024x512.jpg",
  caption: "명일동 아파트 월세, 남향, 역 5분거리",
  price: 680000
},
{
  title: "Banziha One Room",
  image: "https://p.rmjo.in/moodShot/ugt8rk70-512x256.jpg",
  caption: "반지하 원룸입니다. 비올 때 물가끔 새는거 빼면 좋아요",
  price: 370000
}
];
```

원룸 데이터를 첨부해보자  
./data/post.js 같은 파일에 빨리 저장하세요

# Vue 파일 script 섹션 구성

```
export default {  
  name: "app",  
  data() {  
    return {  
      products: ''  
    }  
  },  
  components: { },  
  methods: { },  
  computed: { },  
  props: { },  
}
```

사이트에 필요한 모든 데이터는 여기에 넣기

이 페이지에 첨부할 컴포넌트 이름

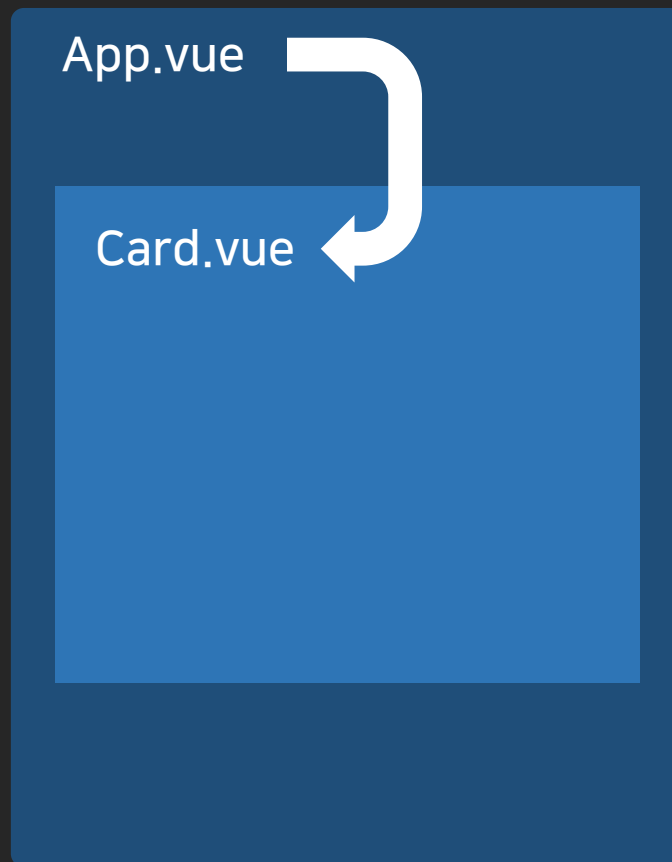
메소드들 (함수들)

제2의 데이터 저장공간

상위 컴포넌트에서 물려받아온 데이터

꼭 알아야 할

## 부모.vue의 데이터를 사용하는 법

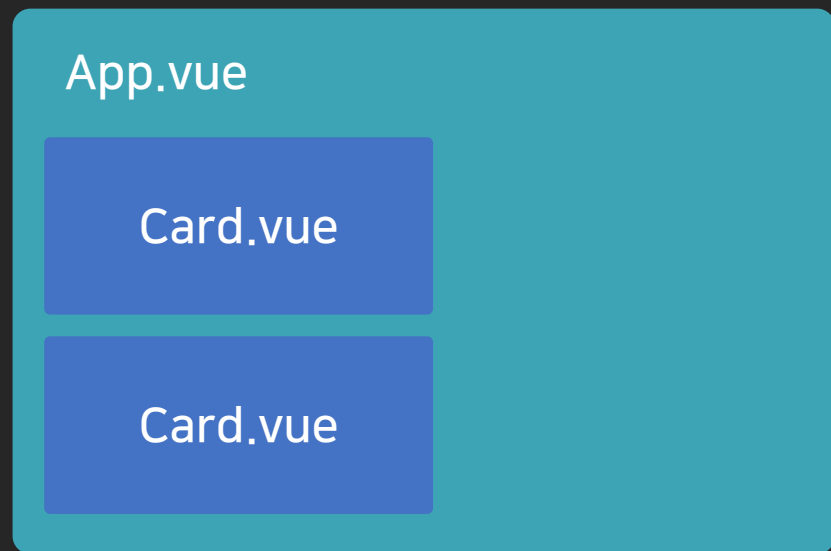


자식에게 데이터 전달은  
Props로 합니다.

그럼 자식은  
부모의 데이터를 사용할 수 있음

# [ 데이터 바인딩 ]

1. 모든 중요데이터는 보통 App.vue에서 보관함
2. 그럼 Card.vue에서 App.vue 데이터를 쓰려면?
  - 데이터를 물려받아야합니다.
  - 물려받은 데이터를 props라고 합니다.



```
<List v-bind:내꺼="아빠꺼" />
```

1. v-bind:props이름="전해줄데이터"

```
import List from './components/List.vue'
```

```
export default {  
  data() {  
    아빠꺼: '정장바지'  
  },  
  components: {  
    List,  
  }  
}
```

List.vue

props 데이터 물려받아와서 사용하기

<p> {{내꺼}} </p>

2. 받아온 props 자유롭게 쓰기

```
export default {  
  props : {  
    내꺼 : String  
  }  
}
```

1. 받아올 props이름과 데이터형 명시해주기

Q. 그냥 부모요소의 data 이름 쓰면 안되나요?

A. 안되니까 이짓거리 하는것

# Props로 데이터 전달해서 쓰는 법 요약

App.vue

<Card /> 첨부할 때  
v-bind:내데이터="부모데이터이름"

Card.vue

props : { 내데이터: String } 등록하고  
내 데이터 자유롭게 사용하기



# [ for 문과 함께하는 props 바인딩 ]

6개의 Card.vue 에는 각각

products[0]

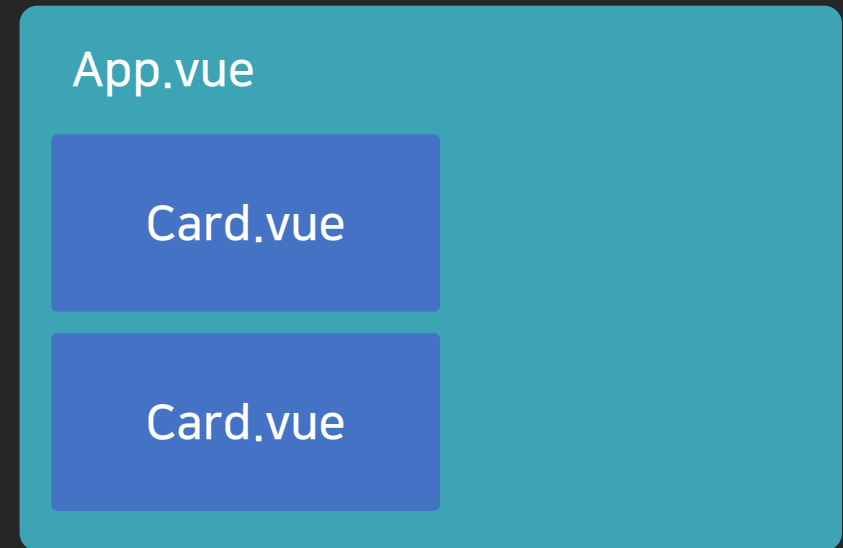
products[1]

products[2]

...

데이터가 props로 전달되어야합니다.

어떻게 전달하죠?



HTML요소가 반복되면 v-for 반복문을 돌리면 됩니다.  
근데 몇번?

```
<card v-for="___ in ___" />
```

상품 데이터 개수 만큼!

App.vue

Card.vue

Card.vue

Vue는 for 반복문을 돌릴 때 key를 강요합니다.

```
<card v-for="___ in ___" v-bind:key="___" />
```

<card>마다 차례로 0,1,2,3,4,5...의 key값을 가지면 되는데  
그러려면 for문을 약간 고치면 됩니다.

App.vue

Card.vue

Card.vue

## <Card />에 반복문으로 Props 전달하는 패턴

App.vue

1. v-for과 v-bind 동시 사용하며  
for 으로 하나씩 뽑은 아이템을 전달

```
v-for="아이템 in 데이터"  
v-bind:전해줄아이템="아이템"
```

App.vue

2. 똑같이 props처럼 사용하기

```
props : { } 에 등록하고  
자유롭게 사용하기
```

# HTML조작이 빨라진 이유 : Virtual DOM

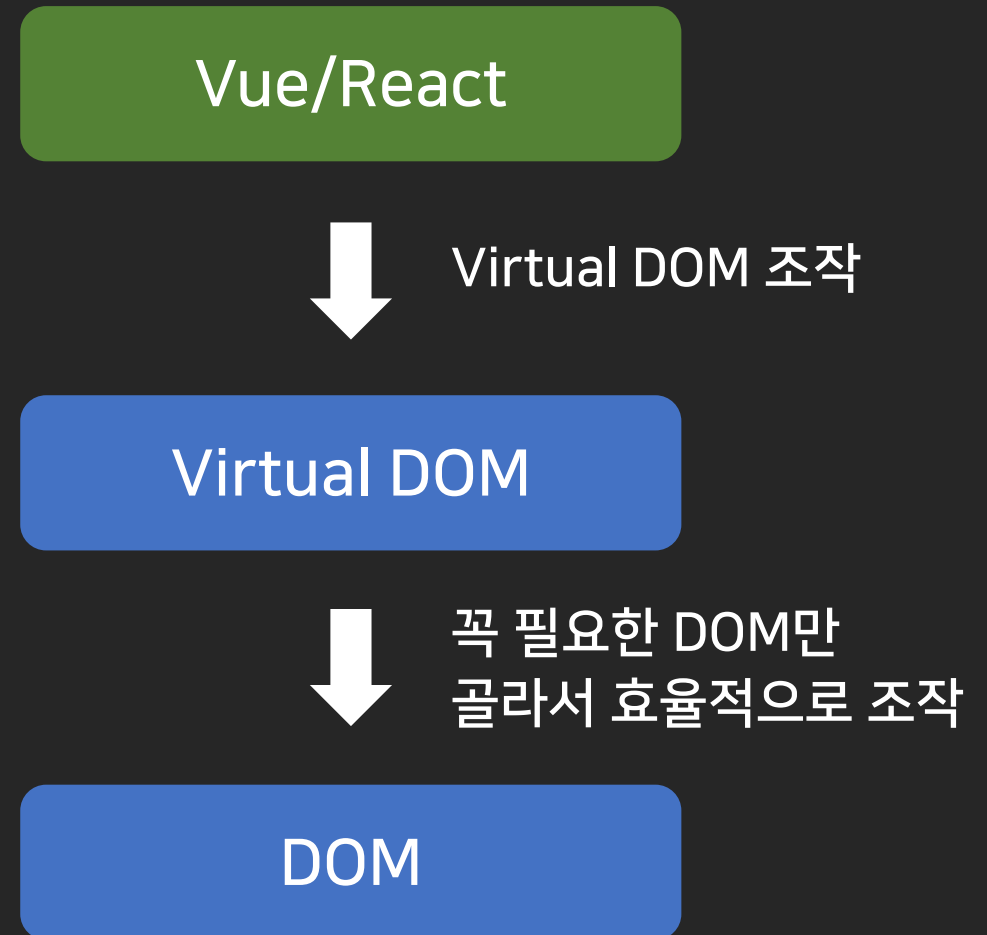
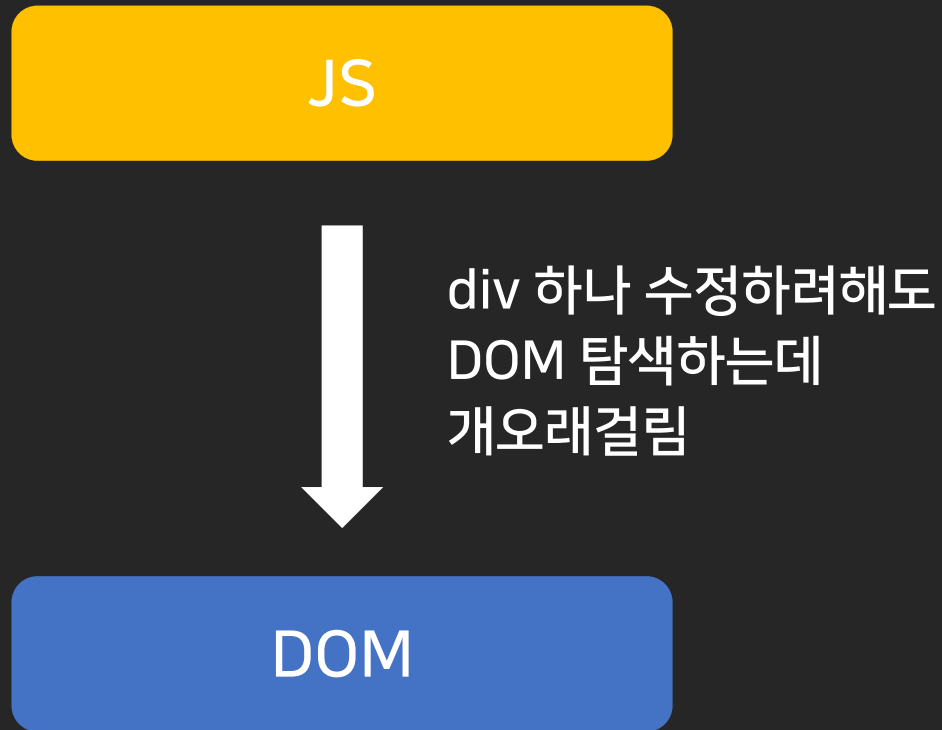
DOM

JS로 HTML을 조작하기위해 object형태로 추상화해놓은 HTML

Virtual DOM

React/Vue언어로 DOM을 카피해놓은 것  
React/Vue 에서 **HTML변경**이 발생하면 여기서 먼저 반영함

# HTML조작이 빨라진 이유 : Virtual DOM



# [ 데이터 정렬 ]

Q. 그림 카드를 가격순 정렬하려면 어떻게 해야할까요?  
(data를 변경하면 HTML도 알아서 re-rendering이 됩니다)

원래대로

가격순 정렬

50만원 이하만



Sinrim station 30 meters away

월 340000 원



## 메소드 만드는 법 (기능만들기)

```
<button v-on:click="가격순정렬">
```

2. 그리고 v-on:click="" 이벤트리스너에 함수첨부

```
export default {  
  methods : {  
    가격순정렬() {  
      ~~  
    }  
  }  
}
```

1. method라는 object 안에다가  
원하는 기능(함수)을 만들자



가격순 정렬 기능을 만들려면 :

- 1.
- 2.
- 3.

원래대로

가격순 정렬

50만원 이하만



Sinrim station 30 meters away

월 340000 원

데이터를 변경하려면 :

<template>안에서는 데이터 이름 (**products**)를 그대로 사용

<script>안에서는 데이터 이름을 this와 함께 사용 (**this.products**)

이유 : object내에서 사용하면 this는 자기 object를 가리킵니다.

# [ 데이터 정렬 ]

Q. sort함수같은건 원본을 변경시키는데, products 데이터 원본을 보호하려면?

원래대로

가격순 정렬

50만원 이하만



Sinrim station 30 meters away

월 340000 원



원래대로 정렬하기 기능을 만들려면 :

- 1.
- 2.

\* Deep copy하려면 [...this.products]

원래대로

가격순 정렬

50만원 이하만



Sinrim station 30 meters away

월 340000 원

페이지로드시 뭔가 실행시키고 싶을 경우  
(onload 이벤트와 유사)

```
<my-list />  
<my-list />
```

```
export default {  
  mounted() {  
    console.log('바보')  
  },  
  
}
```

1. mounted()라는 함수 안에 코드짜시면 됩니다.

기능(함수)을 만들고 싶으면  
`methods` / `computed` / `watch`  
안에 넣어주시면 됩니다.

셋 다 비슷한데 다만..

# computed vs methods vs watchers

어쩌구.vue

```
methods : {  
  정렬하기() { ~ }  
},  
computed : {  
  정렬하기() { ~ }  
},  
watch : {  
  number() { ~ }  
},
```

**methods** 함수들은 부를 때마다 매번 실행함

- render될 때도 실행함

**computed** 는 한번 실행해놓은 값을 저장해놓고 계속 씀

- 일명 '제2의 데이터 저장공간'

- 관련된 데이터가 업데이트 될때만 실행함

- 함수안에 return문을 꼭 가져야함

**watch** 는 한 개의 데이터를 계속 지켜보며 그 데이터가 변경 될 때마다 실행함

- 함수이름 만들 때 꼭 데이터이름으로 해야함

- computed 데이터에도 적용가능

## computed vs methods

```
computed : {  
  정렬하기() {  
    return [1,2,3,4]  
  }  
}
```

1. computed 함수는 함수 쓸 때마다 함수를 실행하지않고 한번 딱 실행하고 남은 값을 가지고 있음
2. computed내의 함수들은 함수가 사용하고있는 데이터가 변경 될 때만 다시 동작함
3. 원래 용도가 계산결과 저장임.. (제2의 데이터저장소)
4. 그래서 computed 함수들은 내부에 return문을 강요함



## computed vs methods 퀴즈 1

Q. 버튼을 누를 때마다 **지금 날짜와 시간**을 가져오는 함수를 만들고 싶은데..  
어디다 만들지?

```
export default {  
  methods : {  
  },  
  
  computed : {  
  }  
}
```

```
현재시간출력() {  
  return new Date()  
}
```

## computed vs methods 퀴즈 2

Q. 이름 200개가 저장된 Array에서 성이 김씨인 사람만 뽑은 Array를 쓰고싶는데  
'김씨만 고르기' 기능은 어디다 만들지?

```
export default {  
  methods: {  
  },  
  
  computed: {  
  }  
}
```

```
김씨고르기() {  
  //이것저것 연산하기  
  return [김XX, 김OO, 김EE]  
}
```

# [ 상세페이지 컴포넌트 만들기 (Modal) ]



Sinrim station 30 meters  
away



애완동물 출입 금지

2개월/ 200만원



계약문의

# [ 상세페이지 컴포넌트 만들기 (Modal) ]

1. 모달 UI 만들기 (이미지/글/버튼)

2. 모달창 열기/닫기 버튼 만들기

- <Card>를 누르면 모달창이 열려야 함 (카드에 이벤트 넣기)
- 닫기버튼을 누르면 모달창이 닫혀야 함



Sinrim station 30 meters  
away

×

애완동물 출입 금지

2개월/ 200만원



계약문의

# <Card> 컴포넌트엔 v-on:click="" 못달아요?

커스텀 event

1. 원래는 커스텀 이벤트 이용해야함 (내가 직접 정하는 이벤트이름!)

.native event

2. 굳이 달고 싶으면 v-on:click.native="" 이렇게 달아야함  
간단하지만 장기적으로 좋은 방법은 아님

참고 : .native는 이벤트 버블링 일어남

참고 : 이벤트시 실행할 함수에서 이벤트관련 기능을 쓰고싶다면 e라는 인자 추가. (e.target 이용가능)

# [ 상세페이지 컴포넌트 만들기 (Modal) ]

1. 모달 UI 만들기 (이미지/글/버튼)

2. 모달창 열기/닫기 버튼 만들기

- <Card>를 누르면 모달창이 열려야 함 (카드에 이벤트 넣기)
- 닫기버튼을 누르면 모달창이 닫혀야 함

3. 모달창을 열면 내용이 바뀌어야함.

- 1번 이미지를 누르면
- 모달창 내용이 1번 데이터의 이미지경로, 제목, 내용으로 바뀌어야함

오늘의 HTML상식 : HTML에다가 data attribute를 넣으면 좋은 점

```
<li data-id = "1" >
```

```
<li data-id = "2" >
```

```
document.getElementById("").dataset.id;
```

- HTML 요소 쉽게 찾기 가능
- HTML에 중요한 정보 삽입 가능

## Vue에서 이벤트함수 사용하기 & 심어놓은 dataset 찾기

```
clickModal(e){  
  console.log(e.target.dataset.id);  
}
```



## 이미지 src 경로 바인딩

```

```

# [ input과 역방향 데이터바인딩 ]

사용자가 입력한 input에 의해 데이터를 변경하려면?  
v-model="바꿀데이터"를 사용합니다.

```
<p>{{ 메시지 }}</p>  
<input v-model="메시지">
```

```
export default {  
  data() {  
    return {  
      메시지: " "  
    }  
  }  
}
```

## 6. 역방향 데이터바인딩

JavaScript 데이터로 HTML을  
변경하는건 많이 했었죠?

HTML데이터가 JavaScript 데이  
터를 변경할 수도 있습니다.

input같은 곳에 **v-model**이라고  
추가하면 사용자가 입력한 데이터  
로 data()안의 값이 변경됩니다.

# [ 월세 금액 계산 슬라이더 만들기 ]



3개월 월세 : 180만원



6개월 월세 : 360만원

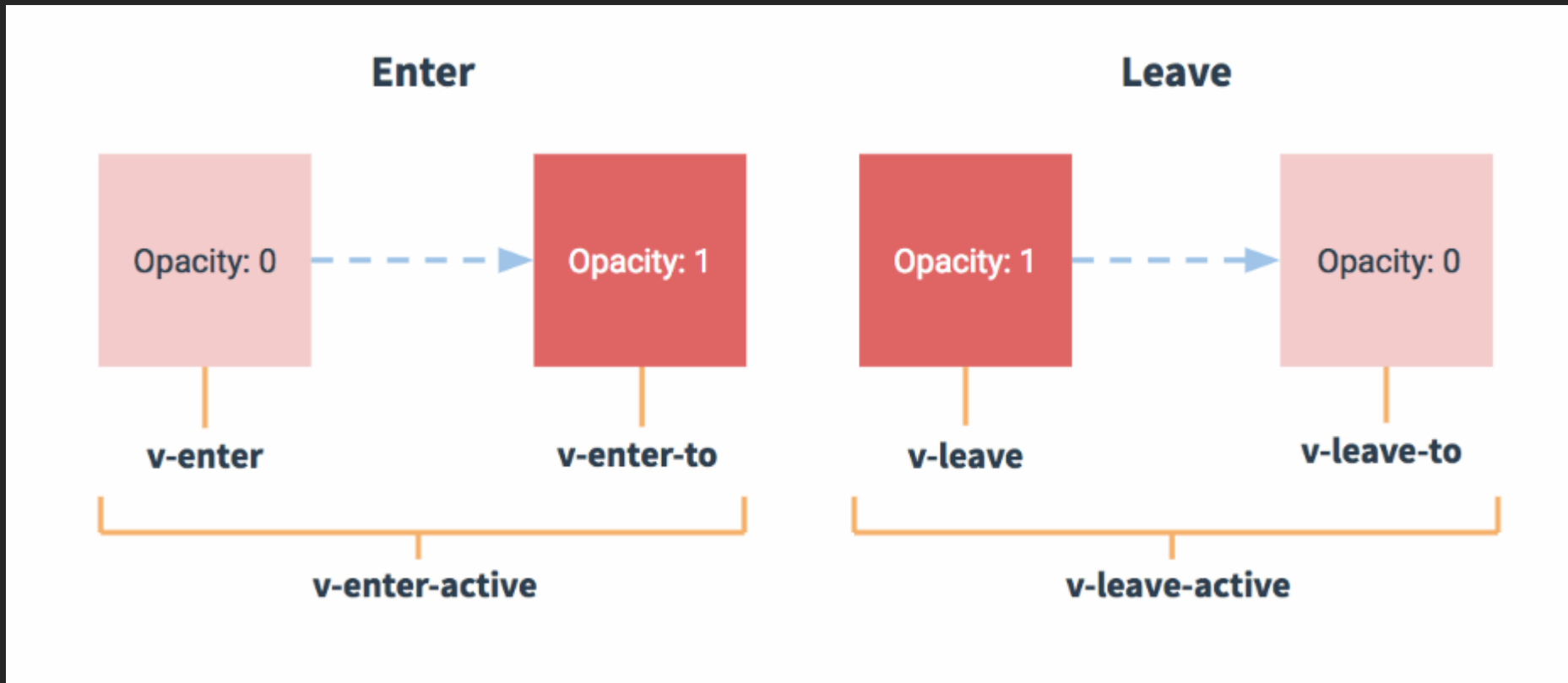
월세 금액계산 슬라이더를 만들어봅시다.  
(슬라이더 HTML은 Bootstrap에서 복붙)

```
<input type="range" class="form-control-range" min="1" max="12" >
```

- 슬라이더도 일종의 폼 input 입니다.
- input에 v-model을 사용하면 input값이 바뀔 때마다 데이터값 변경가능

# [ 간편한 Transition 애니메이션 ]

<transition> 태그 안에 담으면 UI 애니메이션 쉽게 적용가능



# Vue에서 transition 애니메이션 적용하기

```
<transition name="wow">  
  <div v-if="true"> </div>  
</transition>
```

1. transition이라는 태그로 싸매고 name속성으로 이름짓기

```
.wow-enter { 시작 CSS }  
.wow-enter-to { 끝나는 CSS }  
.wow-enter-active { transition : all 0.5s }
```

2. 이름을 넣어서  
세 개의 class를 만들기

# [ 탭기능 1분만에 만들기 ]

옛날 자바스크립트 :  
버튼마다 이벤트 리스너를 달아줍니다.

요즘 Vue :  
(언제나 데이터에 의해 HTML을 만드셔야합니다.)

```
<ul>  
  <button>버튼1</button>  
  <button>버튼2</button>  
  <button>버튼3</button>  
</ul>
```

```
</div>내용1</div>  
</div>내용2</div>  
</div>내용3</div>
```

# [ 무한스크롤 이런건? ]

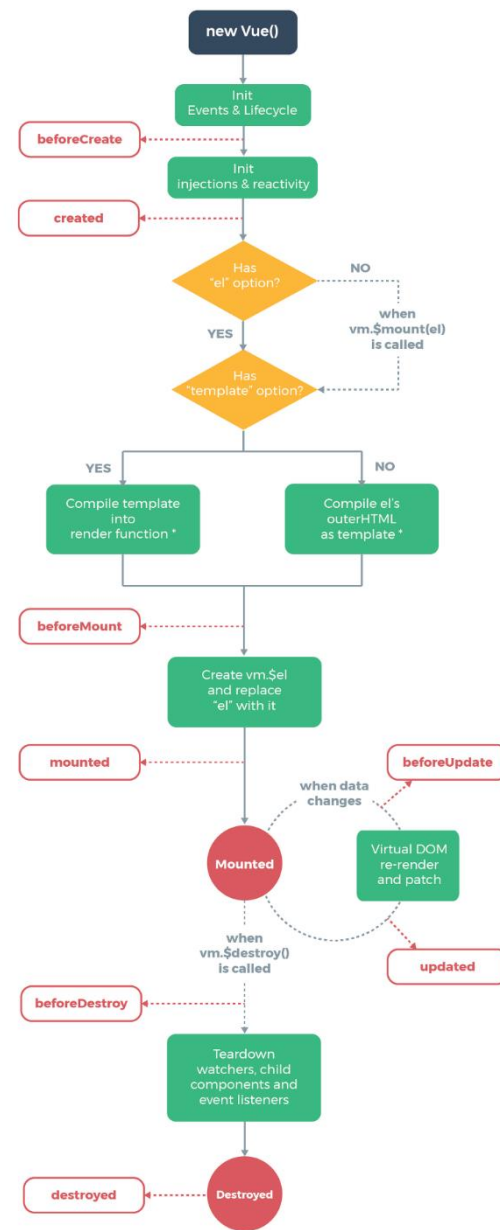
vue infinite scroll 검색 후  
플러그인 설치로 쉽게 이용가능합니다.

npm install 어쩌구라이브러리이름  
yarn add 어쩌구라이브러리이름



# [ .vue 컴포넌트의 생성과정 ]

1. Create    컴포넌트 구성하기 전 데이터만 구성된 시점
2. Mount    컴포넌트를 다 만들고 DOM에 추가됨
3. Update    데이터가 변경되어 컴포넌트도 다시 렌더링 될 때
4. Destroy    컴포넌트가 사라짐



\* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

# 자주 쓰는 라이프사이클 함수들의 용도

`created()` 페이지 첫 로드시 데이터 초기값을 설정해주고 싶을 때

`mounted()` HTML이 다 렌더링된 후 뭔가 변경시키고 싶을 때

`beforeUpdate()` 데이터가 변경되고 DOM 렌더링 전

`updated()` 데이터가 변경되어 재렌더링된 후 (여기선 데이터 변경 금지)

```
export default {  
  created(){  
    //  
  },  
  mounted(){  
    //  
  }  
}
```

Q. 컴포넌트 오픈시 Ajax요청으로 데이터를 가져와야해요.  
서버에 Ajax요청하는 코드는 어디다 작성?

`created()` 페이지 첫 로드시 데이터 초기값을 설정해주고 싶을 때

`mounted()` HTML이 다 렌더링된 후 뭔가 변경시키고 싶을 때

`beforeUpdate()` 데이터가 변경된 직후에 뭔가 동작시킬 때

`updated()` 데이터가 변경되어 재렌더링된 후 뭔가 동작시킬 때

Q. 사용자가 input을 이용해 데이터를 변경할 때마다 값이 적절한지 검사해주고 싶다면 어디에 기능개발?

`created()` 페이지 첫 로드시 데이터 초기값을 설정해주고 싶을 때

`mounted()` HTML이 다 렌더링된 후 뭔가 변경시키고 싶을 때

`beforeUpdate()` 데이터가 변경된 직후에 뭔가 동작시킬 때

`updated()` 데이터가 변경되어 재렌더링된 후 뭔가 동작시킬 때



- Project2. 프로필/블로그
- 컴포넌트만들기
- Router로 페이지 나누기
- Router내의 Props 전달

<https://blackrockdigital.github.io/startbootstrap-clean-blog/about.html>

# [Router : URL 이동을 흉내내는 방법]

일반 웹사이트  
(URL마다 다른페이지)

Vue SPA  
(페이지가 하나 뿐)

라우터를 첨가한 SPA  
(URL로 페이지 구분가능)



새로운 폴더 생성 후 블로그 vue 프로젝트 생성

(터미널 명령어 안알라줌)

vue 라우터 라이브러리 설치 방법 (터미널에 알아서 입력)

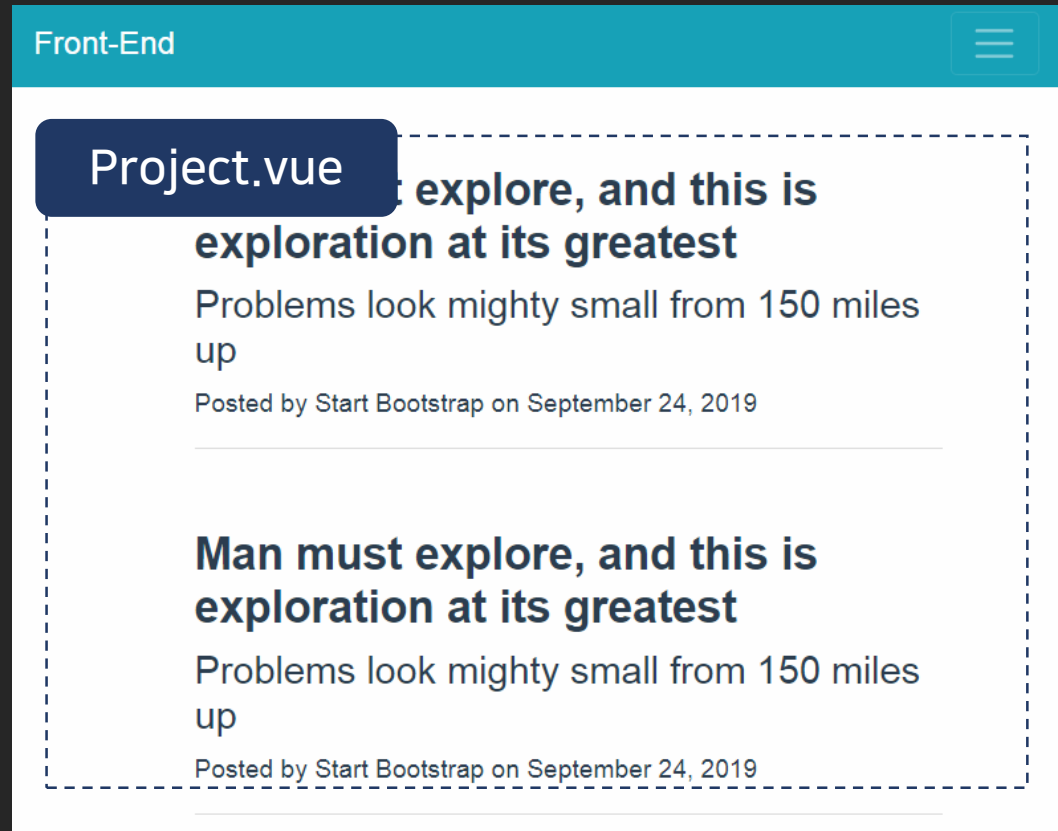
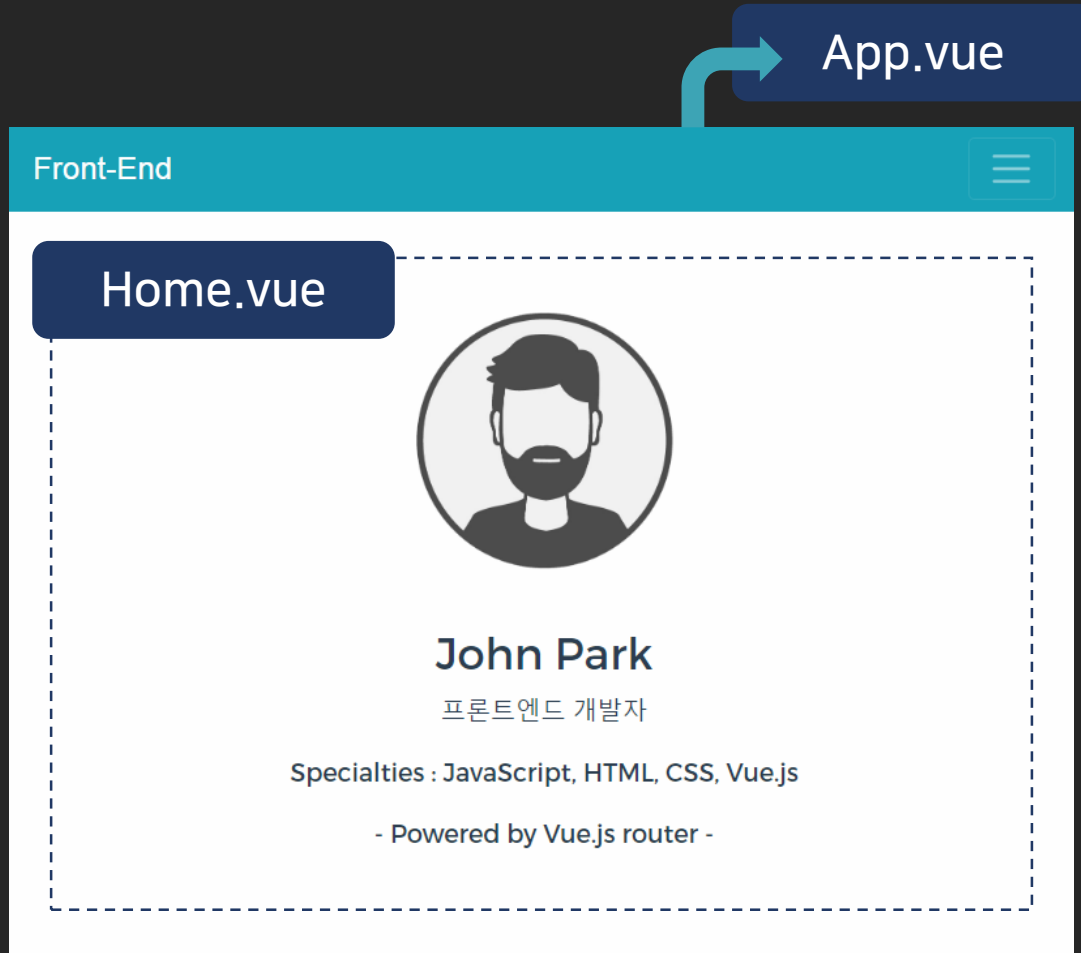
```
yarn add vue-router
```

CSS짜기 귀찮으니 부트스트랩도 설치 (main.js 셋팅까지하셈)

```
yarn add bootstrap-vue
```



# Home / Project 컴포넌트 만들기



## Project.vue에 첨부할 내 블로그 포스팅 데이터 (당연히 App.vue에 보관)

```
포스트들 : [{
  title : '첫 째 프로젝트 : JavaScript 쇼핑몰',
  content : '서버가 아닌 프론트엔드 환경에서 대부분의 기능을 만들어낼 수 있기 때문에 만들어보았습니다.',
  date : 'September 24, 2019',
  number : 0
},{
  title : '둘 째 프로젝트 : 뷰로만든 부동산',
  content : '이제 앱보다는 PWA가 대세입니다. 뷰를 이용하면 매끈한 앱같은 웹을 만들 수 있습니다.',
  date : 'October 20, 2019',
  number : 1
},{
  title : '셋 째 프로젝트 : 현피 앱',
  content : '거리를 설정하면 가장 가까운 파이터를 소개해드려요! 서로 싸워보세요',
  date : 'September 24, 2019',
  number : 2
}],
```

## Router 쓰는 법 (설정편)

main.js

```
import Vue from 'vue'  
import App from './App.vue'  
import router from './router.js';
```

```
new Vue({  
  router,  
  render: h => h(App),  
}).$mount('#app')
```

1. 아까 만든 router.js파일 첨부

2. new Vue 안에 router : router 첨부

# Vue에서 라우터 쓰는 패턴 간단정리

router.js

/project 로 방문하면  
<router-view>에서  
Project.vue를 보여주세요

1. 일단 동작 방법설정

App.vue

<router-link to="/project"> 로 이동버튼 만들고  
<router-view> 어딘가에 박아넣기

2. 메인페이지에서 버튼이랑 내용 설정

App.vue

## Router 쓰는 법 (보여주기)

```
<template>
  <nav>
    <router-link to="/">Home</router-link>
    <router-link to="/project">Project</router-link>
  </nav>

  <router-view />
</template>
```

1. router-link로 이동버튼 넣기

2. 내용보여줄 router-view 원하는 위치에 넣기

## Router 쓰는 법 (경로설정)

router.js

```
import Vue from 'vue'
import Router from 'vue-router'
import Home from './components/Home.vue'
```

```
Vue.use(Router)
```

```
export default new Router({
  routes: [
    {
      path: '/',
      name: 'home',
      component: Home
    }
  ]
});
```

1. 필요한 파일 첨부 (라우터, 라우팅할 컴포넌트들)

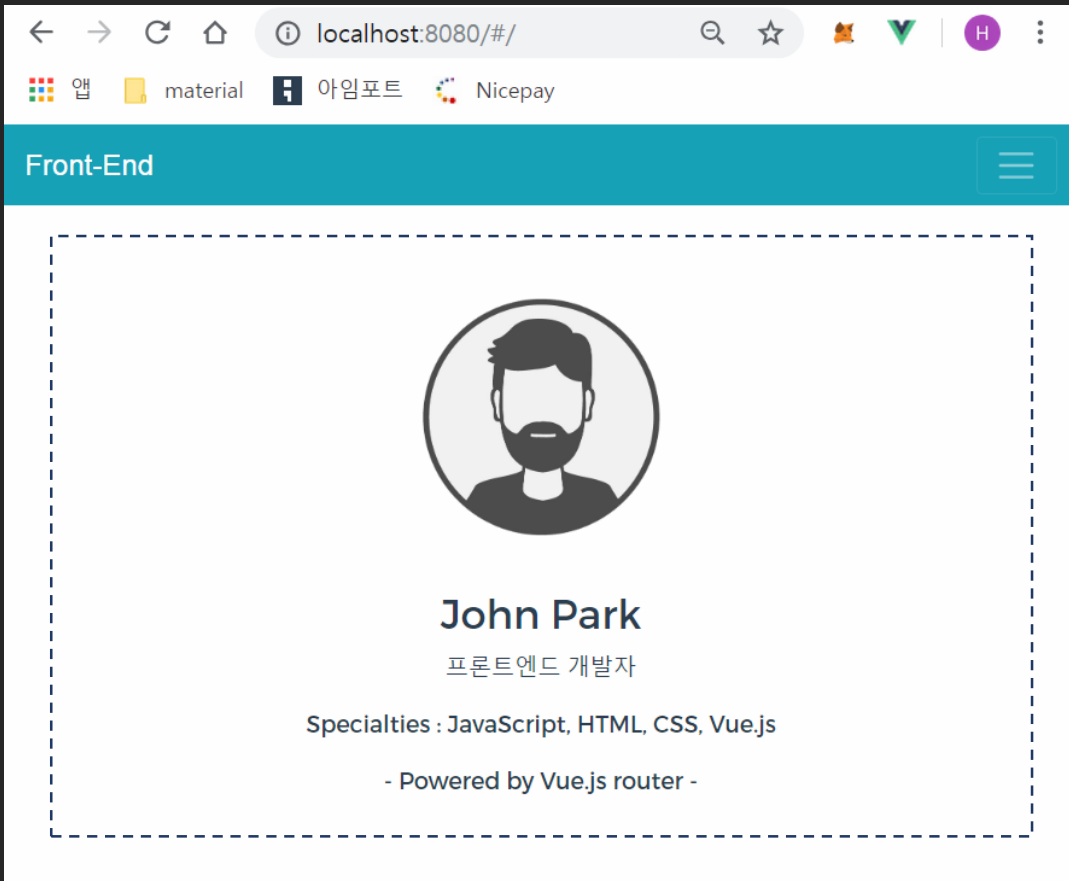
2. path는 원하는 url 경로,  
name은 이름 아무거나

3. component에는 링크방문시 <router-view>에  
서 보여줄 컴포넌트

# Home, Project 페이지 라우팅하기

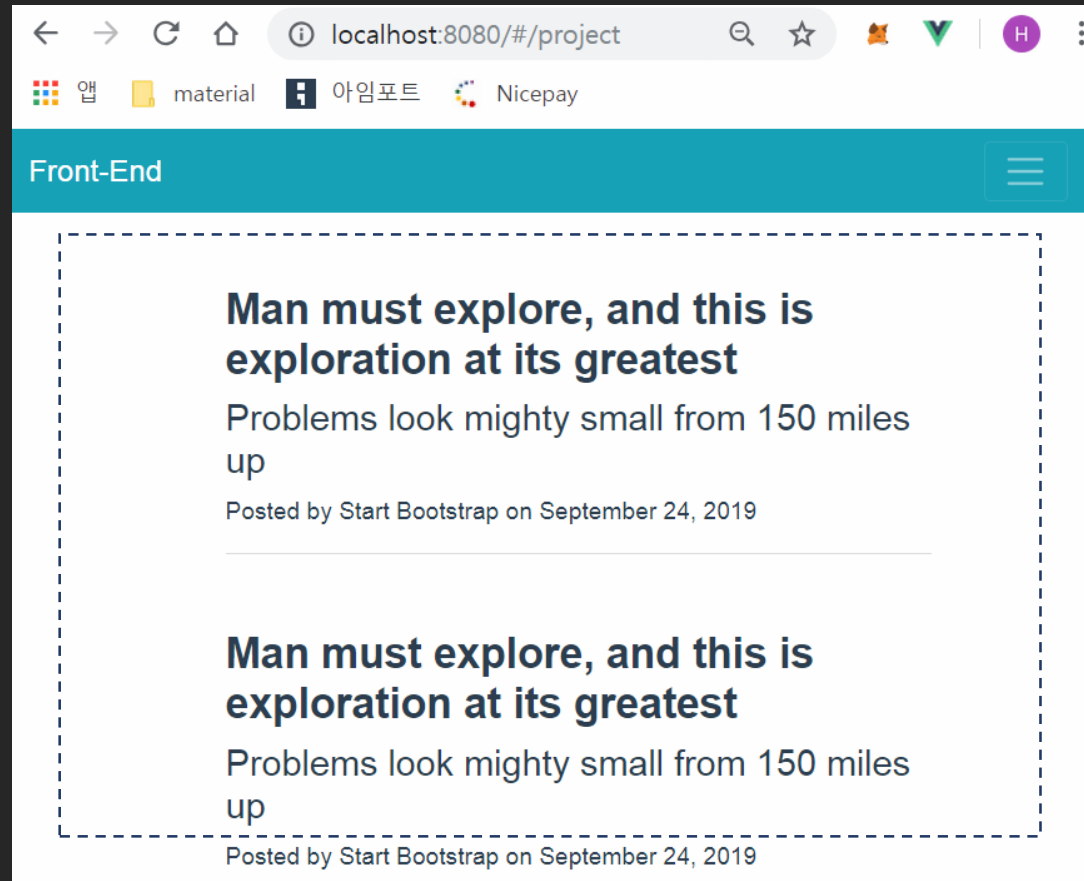
Home

누르면 Home.vue가 보여야 함



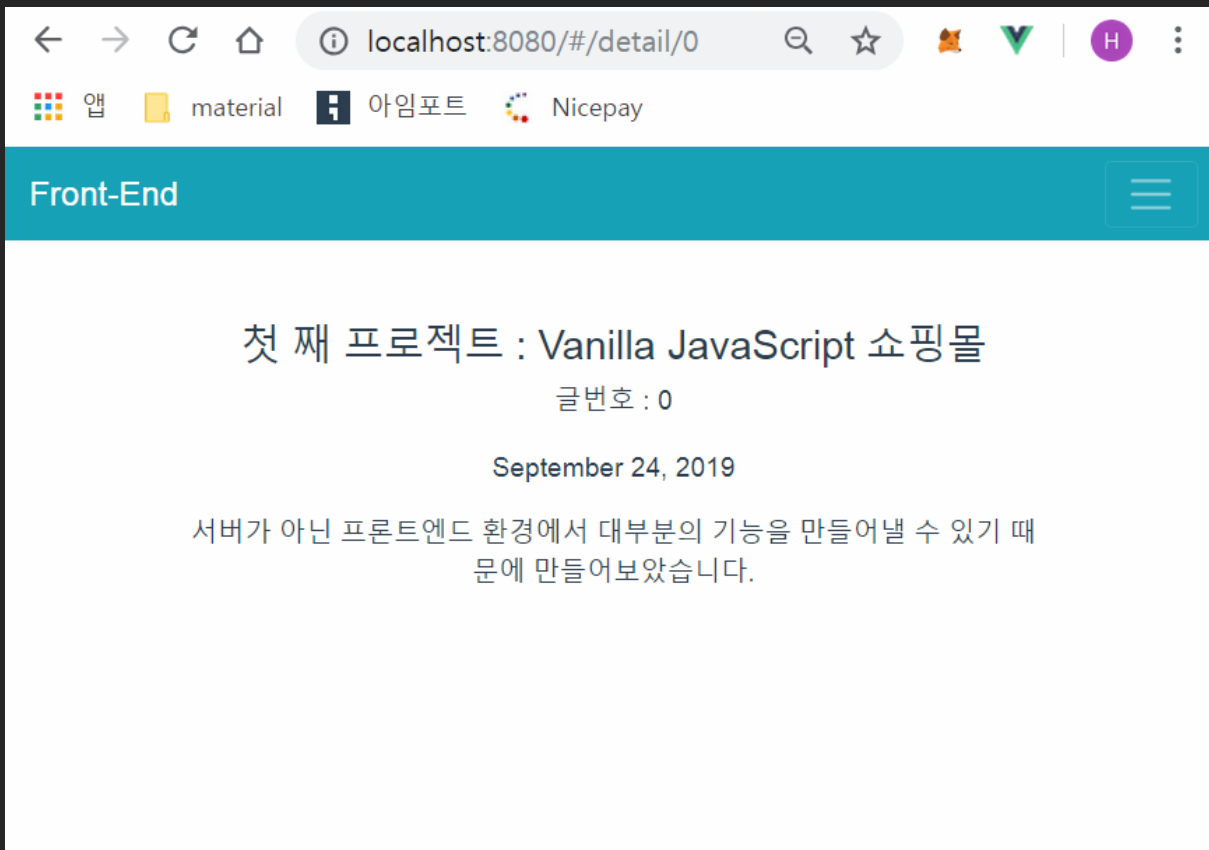
Project

누르면 /project 로 이동하며  
Project.vue가 보여야 함

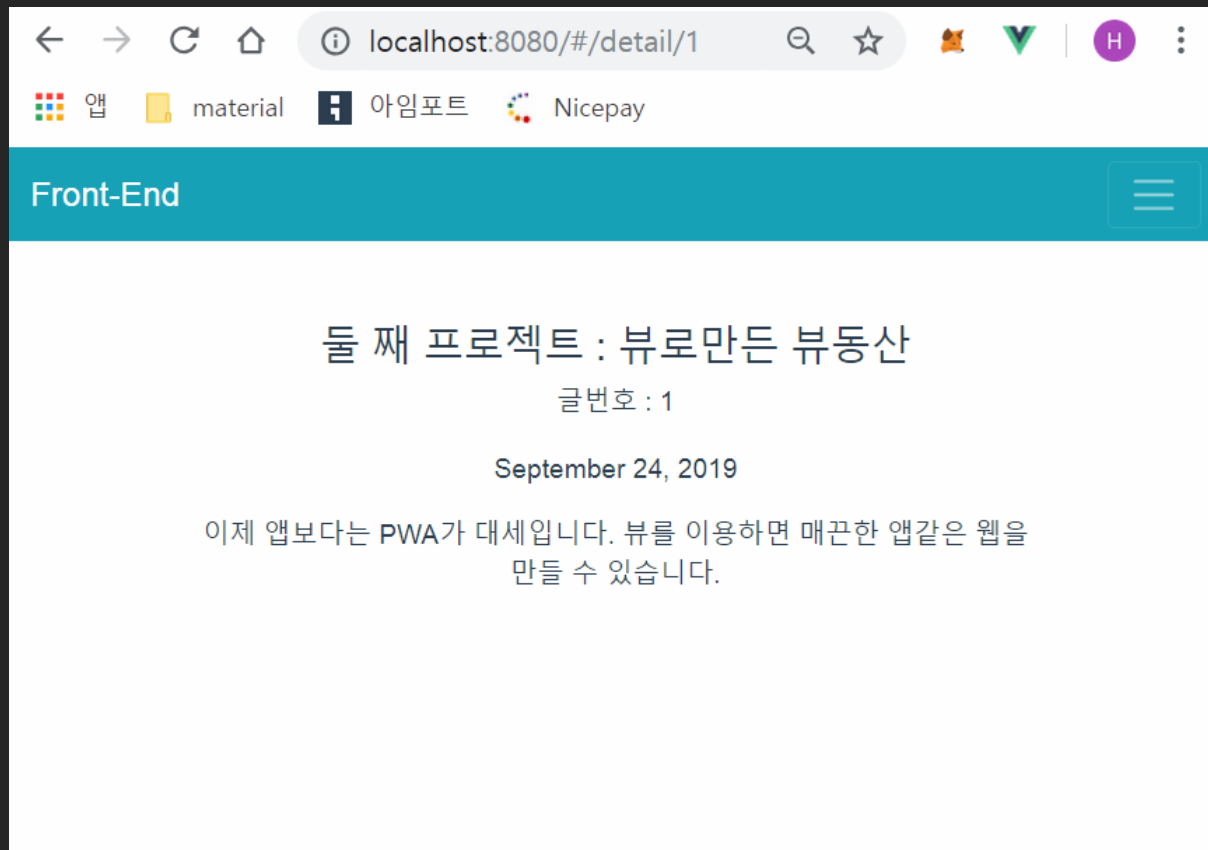


# /detail 상세페이지 라우팅하기

/detail/0



/detail/1





## /detail 상세페이지 라우팅하기

localhost/detail/0

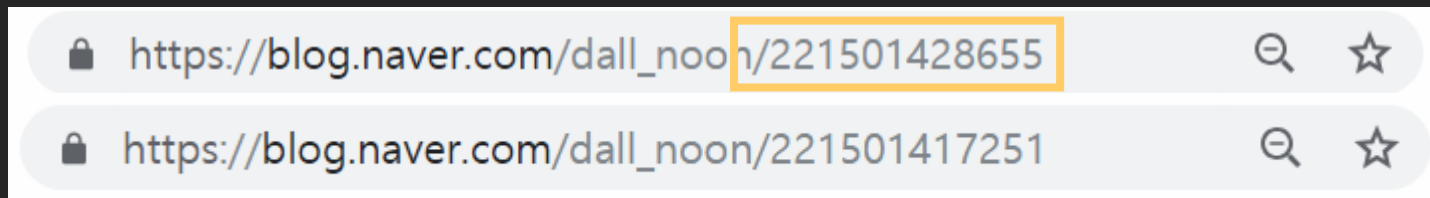
여기서는 첫 째 프로젝트가 보여야함

localhost/detail/1

여기서는 둘 째 프로젝트가 보여야함

첫 째 프로젝트, 둘 째 프로젝트 이렇게 각각의 컴포넌트를 만들어야하나?  
그런데 둘다 비슷한 페이지 아닌가..? (제목만 다르고..)

## ★ URL의 파라미터 (Parameter)



### 파라미터 용도 :

블로그 포스팅을 구분할 때

게시판에서 현재 몇번째 페이지인지 구분할 때

...

- URL에 몰래 **데이터**를 심어놓을 수 있습니다.
- **데이터**에 따라 어떤 내용을 보여줄지 구분할 수 있습니다.

## ★ URL의 파라미터 (Parameter)

router.js 설정 부분

```
path: '/detail/:id',  
name: 'detail',  
component: Detail
```



/detail 페이지로 이동할 때  
:id 자리에 뭔가 글자를 입력 가능

/detail/바보

▲ 바보라는 글자가 **파라미터**!

router.js

## Router 쓰는 법 (파라미터편)

```
import Vue from 'vue'
import Router from 'vue-router'
import Home from './components/Home.vue'
import Detail from './components/Detail.vue'
```

```
Vue.use(Router)
```

```
export default new Router({
  routes: [
    {
      path: '/',
      name: 'home',
      component: Home
    },
    {
      path: '/detail/:id',
      name: 'detail',
      component: Detail
    }
  ]
});
```

1. path에 :어쩌구로 원하는 파라미터 넣기가능

Detail.vue 개발 방법 :

그럼 사용자가

파라미터에 0을 입력하면 0번째 게시물을 보여주면 되고

파라미터에 1을 입력하면 1번째 게시물을 보여주면 되겠군

/detail/1

게시물 1

# Detail.vue 개발 : 근데 파라미터에 어떤 데이터가 들어왔는지 내가 어케알어

localhost:8080/detail/0

localhost:8080/detail/1

`{{ $route.params.id }}` 혹은 `this.$route.params.id`

라고 적으면 지금 파라미터에 적힌 데이터를 알 수 있습니다.

# 그럼 이제 0번째 게시물을 어떻게 보여주는데여?

/detail/0

1. App.vue에 있던 3개의 포스트 데이터를 detail.vue까지 전해줌 (Props)
2. detail.vue가 렌더링되기 전에,
  - 지금 **파라미터**가 0번째인지 1번째인지 검사함
  - 파라미터에 맞는 포스팅을 props에서 딱 하나 골라서 data()에 저장함
3. 저장한 data를 HTML에 {{데이터바인딩}}

# 라우터 컴포넌트들에게 Props 데이터 전달하는법

App.vue

```
<router-view v-bind:포스트들="포스트들"></router-view>
```

(그냥 평소 props전달 처럼.. v-bind 똑같이 쓰면 됩니다)



## 파라미터로 페이지 내용 구분하는 것의 문제점

/detail/0



/detail/1

이렇게 이동하면 1번째 게시물 안보여주는데어?

파라미터를 바꾼다고 HTML을 re-렌더링하지 않습니다.

★ 파라미터가 바뀔 때 데이터를 변경시켜줘야 HTML이 제대로 렌더링 됩니다.

## 파라미터로 페이지 내용 구분하는 것의 문제점

/detail/0



/detail/1

이렇게 이동하면 1번째 게시물 안보여주는데여?

방법1. <router-view>에 :key="\$route.fullPath" 를 추가

방법2. watch() 로 파라미터 변경 체크하기

```
watch: {  
  '$route' () {  
    //페이지 URL 변경시 시킬 작업  
  }  
}
```

## URL 경로 이동시키는 두 가지 방법 (참고)

### 1. 버튼으로 만들기

`<router-link to="/detail/0">` 버튼 `</router-link>`

### 2. push함수 쓰기

```
$router.push('detail')
```

```
$router.push({ name: 'detail', params: { id: '0' } })
```

▲ 파라미터 넣어야할 경우엔 이렇게

참고) `<router-link :to="{ name: 'detail' }">` 이렇게 넣어도 됨

## Nested Routes – 라우터 안에 또 라우터 집어넣기

/detail/1/**author**

게시물 1

Author.vue

/detail/1/**comment**

게시물 1

Comment.vue

## Nested Routes (라우터 설정 편)

```
{  
  path: '/detail/:id',  
  name: 'detail',  
  component: Detail,  
  children : [  
    { path: 'author', component: Author },  
    { path: 'comment', component: Comment },  
  ]  
}
```

1. children이라는 이름의 Array 생성

2. path 와 component 첨부 (+ import 까지)



/detail/1/author

에서 보여줄 컴포넌트



/detail/1/comment

에서 보여줄 컴포넌트

## Nested Routes (HTML 설정 편)

Detail.vue

```
<router-view></router-view>
```

요렇게 쓰면 아까 설정한 컴포넌트를 보여줌