

Vue.js 프레임워크

inky4832@daum.net

1장. Vue.js 개요



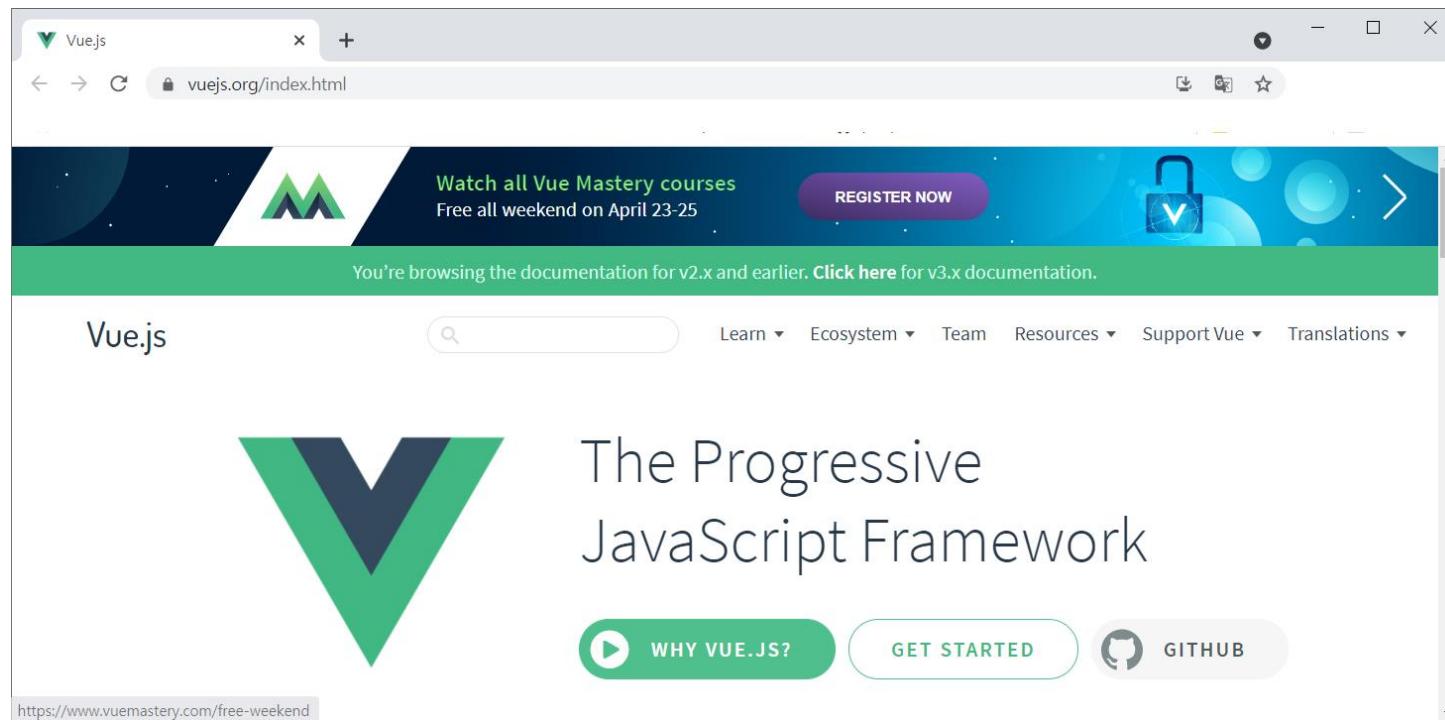
vue.js 소개

MVVM 패턴

Vue.js 환경설정

○ Vue.js 소개 (<https://vuejs.org/v2/guide/>)

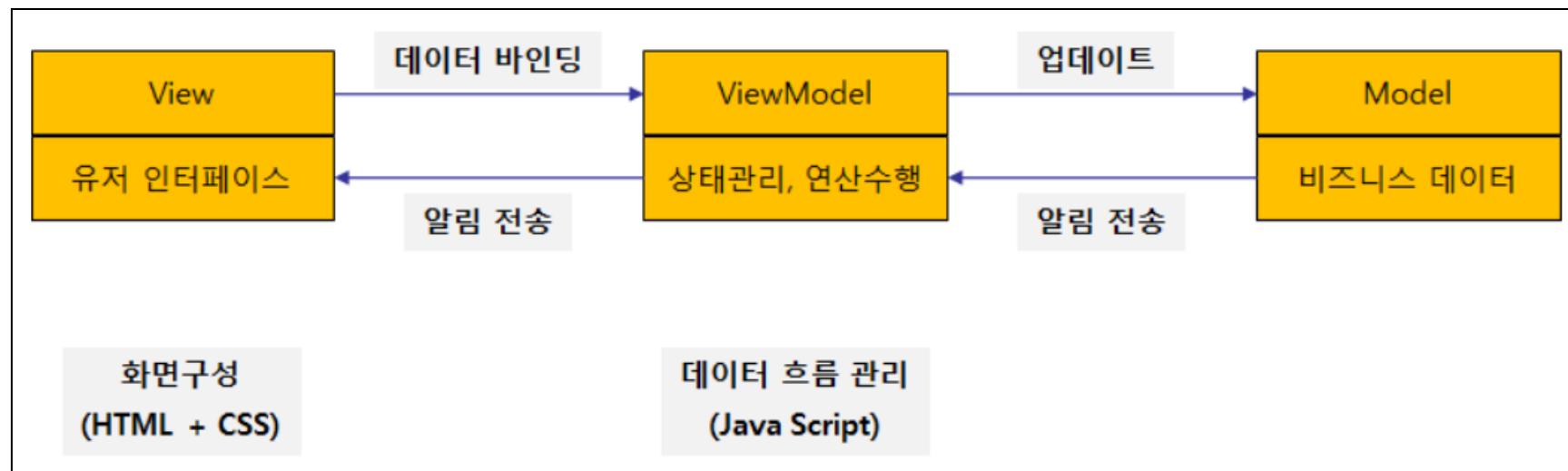
- 가. 예반 유(Evan You) 가 2013년에 발표한 Front-end Javascript framework.
- 나. Angular 와 React의 장점만 가진 프레임워크.



2) MVVM 패턴

Vue.js

Vue나 뷰의 경쟁자들인 React, Angular와 같은 프레임워크들이 지향하는 패턴. MVVM 패턴은 Model, View, ViewModel의 약자로 어플리케이션 로직과 UI의 분리를 위한 패턴이다.



ViewModel은 Model과 View를 연결하는데 Model의 데이터가 변경되면 View에 전달해서 화면을 갱신하게 하고 사용자가 화면에서 값을 변경하면 다시 그 값을 Model에 업데이트 시켜준다.

3) 다른 프레임워크와의 비교

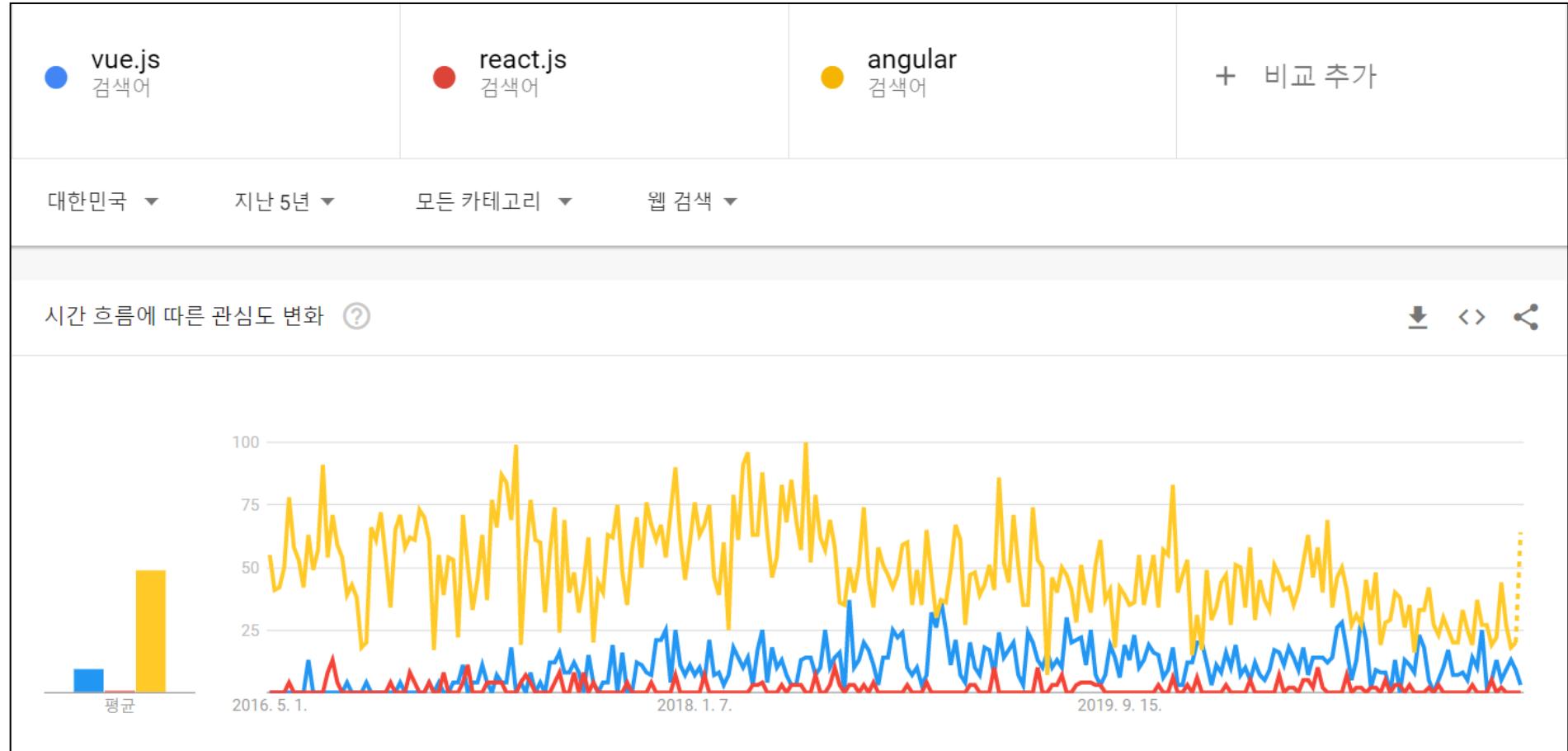
<https://kr.vuejs.org/v2/guide/comparison.html>

항목	Angular	React	Vue
안정성	안정적	안정적	안정적
생태계 지원	거대 커뮤니티와 Google 지원	거대 커뮤니티와 Facebook 지원	상대적으로 작은 커뮤니티와 Alibaba등 지원
문서화	좋음	좋음	좋음(한글)
학습 난이도	TypeScript 등 학습 필요	보통	쉬움
크기	566K	139K	58.8K
코드 재사용성	가능	CSS만 가능	가능
코딩 속도	느림	보통	빠름
반응성	상대적	좋음	좋음

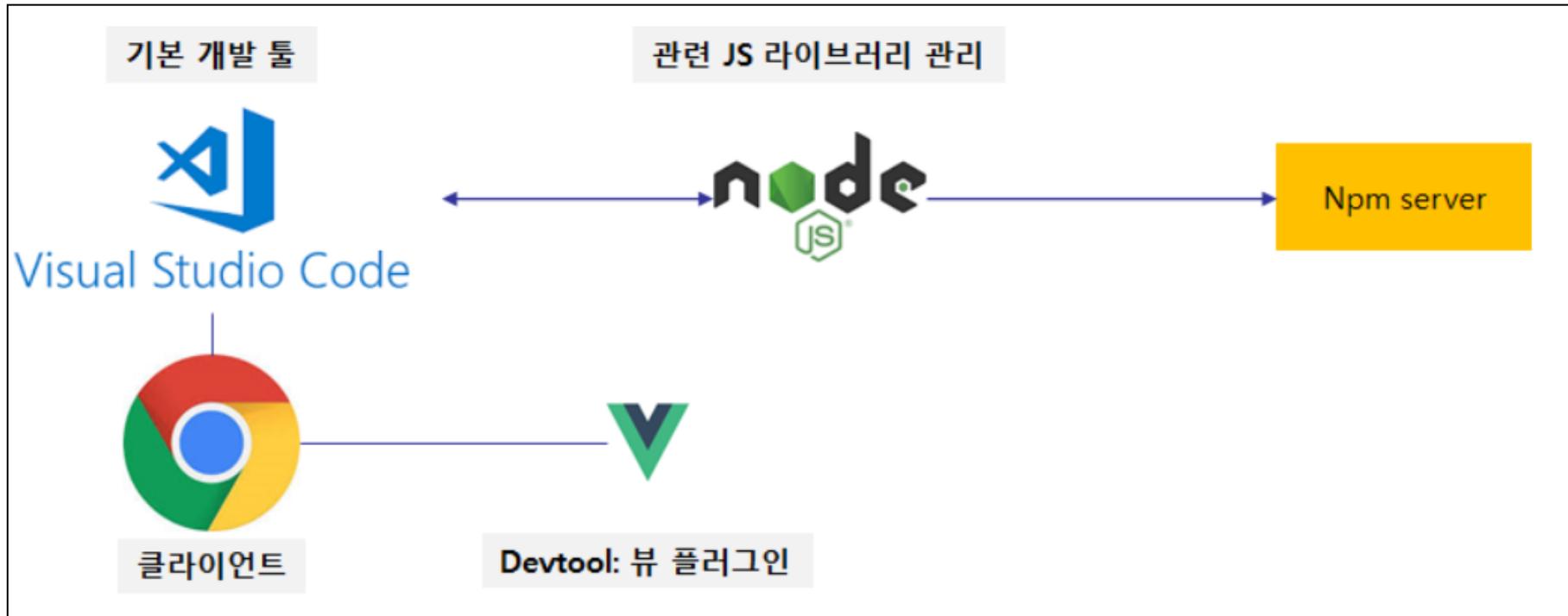
4) Google Trends

Vue.js

<https://trends.google.com/trends/explore?date=today%205-y&q=vuejs,react,angular>



○ Vue.js 개발을 위한 환경 설정



5) Vue.js 환경 설정

Vue.js

가. Node.js 설치

<http://nodejs.org>

The screenshot shows the official Node.js website. At the top, there's a navigation bar with links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. Below the navigation, a message states "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine." A "#BlackLivesMatter" button is present. A green box highlights "New security releases now available for 15.x, 14.x, 12.x and 10.x release lines". Below this, a "Download for Windows (x64)" section features two buttons: "14.16.1 LTS" (Recommended For Most Users) and "16.0.0 Current" (Latest Features). A note at the bottom suggests looking at the Long Term Support (LTS) schedule.

This screenshot shows the "Downloads" section of the Node.js website. It includes instructions for automatic tool installation via Chocolatey, a link to manual dependency installation, and a note about following GitHub instructions for Windows users.

```
C:\WINDOWS\system32>node -v  
v14.16.1
```

나. 개발 툴 설치 (Visual Studio Code)

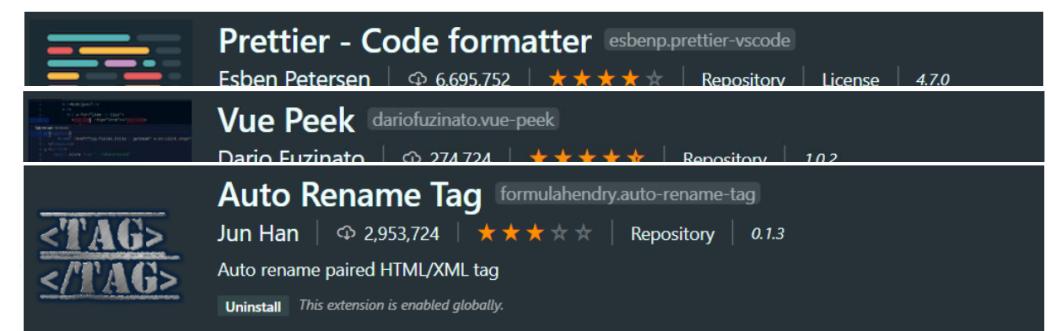
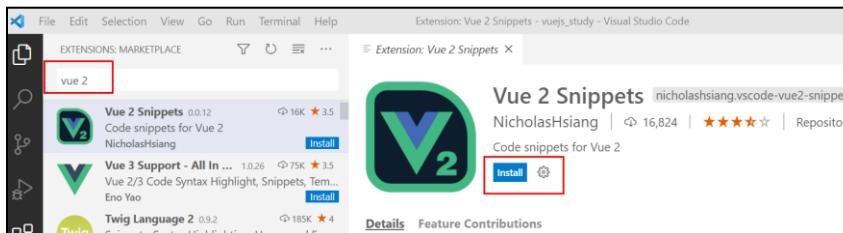
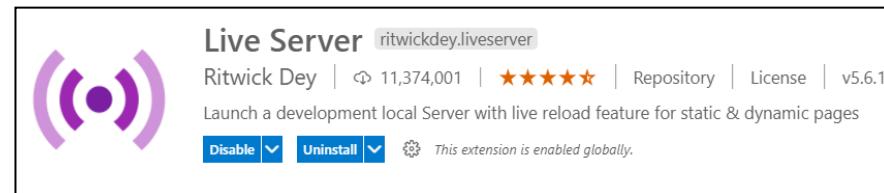
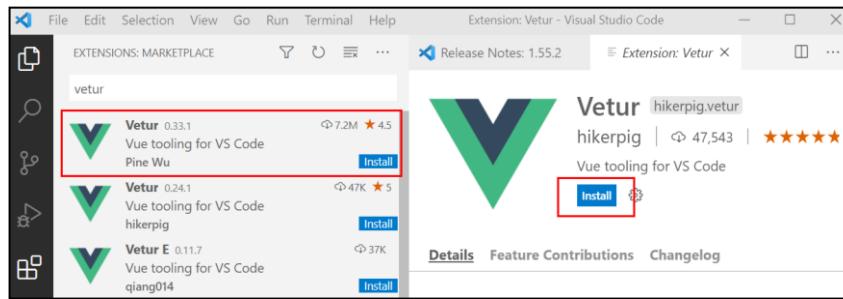
<https://code.visualstudio.com/>

The screenshot shows the Visual Studio Code website. The header includes the logo, "Visual Studio Code", and links for Docs, Updates, and Blog. A banner at the top says "Version 1.55 is now available". The main content area features the tagline "Code editing. Redefined." in large white text on a dark background. Below it, a sub-tagline reads "Free. Built on open source. Runs everywhere." A prominent blue "Download for Windows" button is highlighted with a red box, indicating a stable build. Other download options for other platforms and insiders editions are also shown.

5) Vue.js 환경 설정

Vue.js

다. VSC extension 추가 (vetur 와 vue 2 Snippets, Live Server, ..)



라. Chrome에 Vue.js devtools 확장 프로그램 추가



6) Vue.js 개발 방법

Vue.js

<https://kr.vuejs.org/v2/guide/installation.html>

1) CDN 이용

직접 <script> 에 추가

다운로드 받아 script 태그에 추가하기만 하면 됩니다. `Vue` 는 전역 변수로 등록됩니다.

! 개발 중에는 프로덕션 버전을 사용하지 마십시오. 경고 및 일반적인 실수를 놓칠 수 있습니다!

개발 버전

모든 오류 메시지 및 디버그 모드

프로덕션 버전

오류 메시지 없음, 33.30kb min+gzip

HTML

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

2) CLI 이용

<https://cli.vuejs.org/guide/installation.html>

```
npm install -g @vue/cli  
# OR  
yarn global add @vue/cli
```

sh

2장. CDN 활용한 개발 개요



기본구조

HelloWorld 실습

Vue 객체

Vue 객체의 유효범위

1) 기본 구조

Vue.js

<https://kr.vuejs.org/v2/guide/index.html>

HTML

```
<!-- 개발버전, 도움되는 콘솔 경고를 포함. -->
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

HTML

```
<div id="app">
  {{ message }}
</div>
```

JS

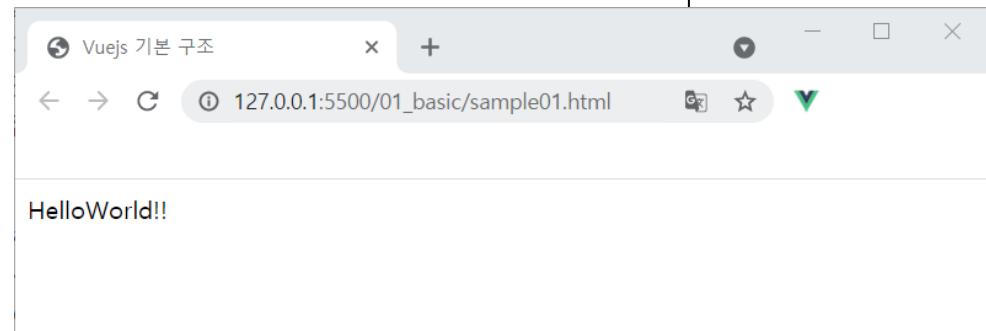
```
var app = new Vue({
  el: '#app',
  data: {
    message: '안녕하세요 Vue!'
  }
})
```

안녕하세요 Vue!

2) HelloWorld 실습

Vue.js

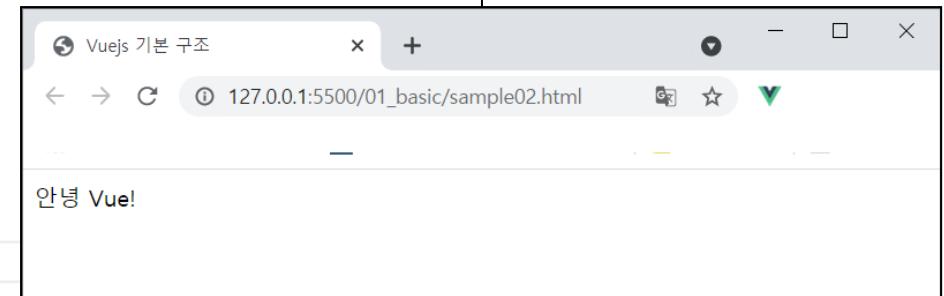
```
sample01.html ×  
vusjs_exam > 01_basic > sample01.html > html > body > div#app  
1  <!DOCTYPE html>  
2  <html lang="en">  
3  <head>  
4      <meta charset="UTF-8">  
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">  
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">  
7      <title>Vuejs 기본 구조</title>  
8      <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>  
9  </head>  
10 <body>  
11     <div id="app">  
12         HelloWorld!!  
13     </div>  
14  
15     <script>  
16         var app = new Vue({  
17             el: '#app'  
18         });  
19     </script>  
20 </body>  
21 </html>
```



3) 안녕 Vue! 실습

Vue.js

```
< sample02.html >
vusjs_exam > 01_basic > < sample02.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Vuejs 기본 구조</title>
8      <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
9  </head>
10 <body>
11   <div id="app">
12     {{message}}
13   </div>
14
15  <script>
16    let model ={
17      message: "안녕 Vue!"
18    };
19    var app = new Vue({
20      el:'#app',
21      data:model
22    });
23  </script>
24 </body>
25 </html>
```



3) 안녕 Vue! 실습의 MVVM 패턴 이해

Vue.js

View

Vue.js 설치

Model

ViewModel

- Vue 객체

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Insert title here</title>
  </head>
  <body>
    <div id="app">
      {{message}}
    </div>
  </body>
```

디렉티브

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

```
<script>
  let model = {
    message: "안녕 Vue!",
  };

```

```
let vueInstance = new Vue({
  el: "#app",
  data: model,
});
</script>
</html>
```

View (Template)

- HTML과 CSS로 이루어진 요소
- Vue.js의 디렉티브 또는 {{ }} 같은 템플릿 표현식으로 HTML DOM에 데이터를 랜더링한다.
- v-xxxx 형식의 다양한 디렉티브를 사용할 수 있다.

Model

- JSON 형태로 데이터를 표현한다.

ViewModel

- Controller 역할을 하며 Vue객체가 담당한다.
- 가장 중요한 역할은 View와 Model의 연결이다.

```
let vi = new Vue({
  el: '#app', // 연결할 View 선택
  data : model // 사용할 Model 설정
});
```

4) Vue 객체

Vue.js

Vue 객체의 생성자는 옵션 객체를 통해서 필요한 속성들을 설정한다.

다음은 Vue 객체에서 사용 가능한 대표적인 속성들이다.

속성 명	설명
el	Vue로 만든 화면이 그려지는 인스턴스의 시작 지점 - CSS로 요소 지정
data	인스턴스의 데이터 속성으로 model을 나타냄
template	화면에 표시할 HTML, CSS등 마크업 요소를 정의하는 속성 - Vue의 데이터 및 기타 속성들도 함께 화면에 렌더링
methods	이벤트 및 화면 동작 메서드로 속성들은 function
created...	라이프 사이클 커스터마이징 로직

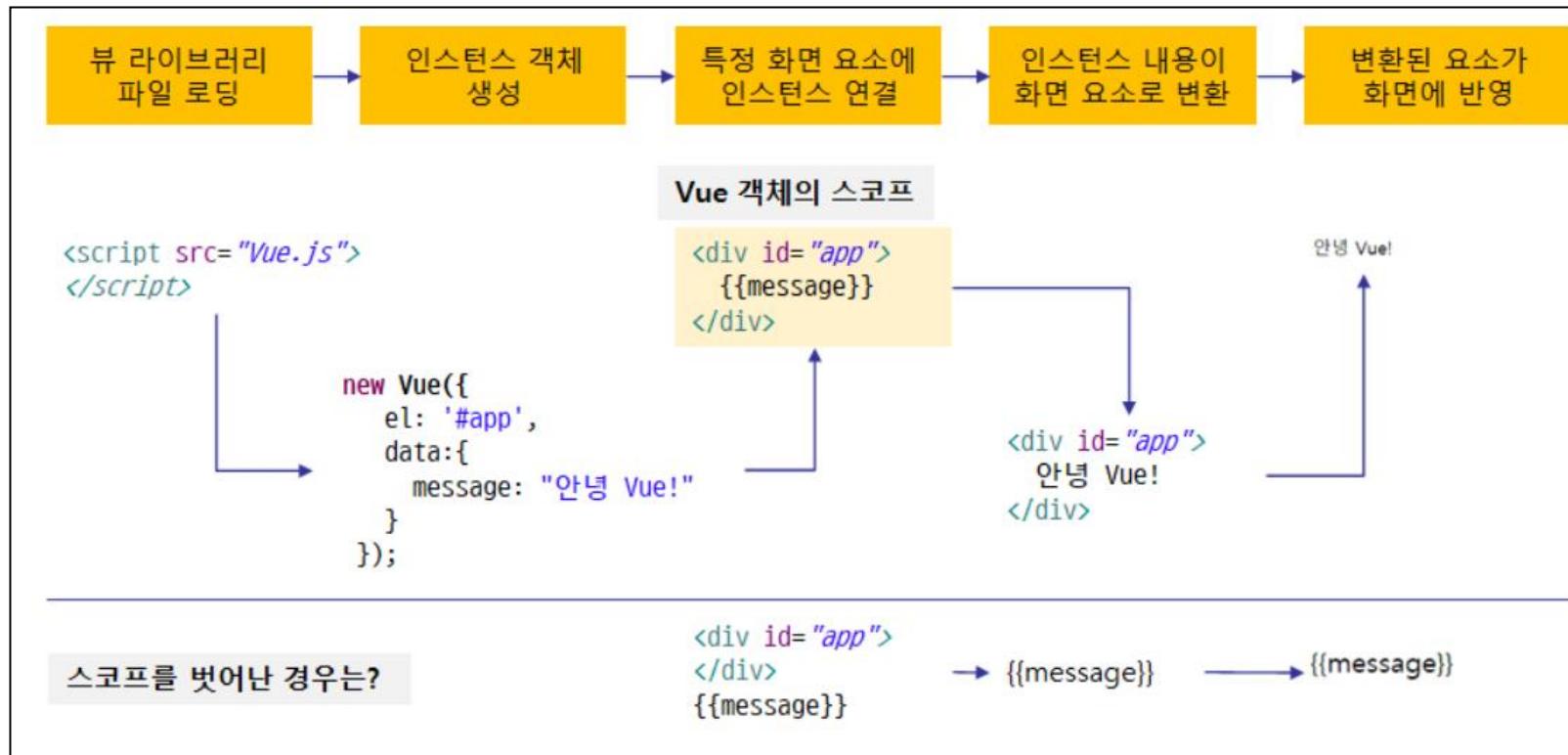
5) Vue 객체의 유효범위 (scope)

Vue.js

Vue 객체의 유효범위는 Vue 객체가 적용되는 HTML의 범위를 의미하며 el 속성에 지정된 영역을 가리킨다.

해당 범위에서만 Vue가 동작하기 때문에 지정된 scope를 벗어나면 {{}} 표현식 등은 단순한 문자열로 처리된다.

다음은 Vue 객체가 동작하는 과정을 보여준다.



3장. 디렉티브 (directive)



디렉티브

v-bind

v-model

v-for

* 지시자 (directive)

- 디렉티브는 지시자로서 Vue에서 template에 데이터를 표현하는 등의 용도로 사용되는 특별한 속성이다.
- Vue 의 디렉티브들은 v-를 접두어로 갖는다.

* 지시자 (directive) 종류

텍스트 표현: v-text, v-html, v-once

html 속성 바인딩: v-bind

양방향 데이터 바인딩: v-model

제어문: v-show, v-if, v-else, v-else-if

반복문: v-for

2) 텍스트 표현

Vue.js

* v-text, v-html, v-once

디렉티브	설명
v-text, {}	innerText 속성에 연결되며 문자열 그대로 화면에 표현. Model과의 반응성이 유지됨
v-html	innerHTML 속성에 연결되며 문자열 중 html 태그를 파싱해서 화면에 나타냄. Model과의 반응성이 유지됨
v-once	v-text,v-html에 부가적으로 사용되며 v-once는 한번 연동 후 반응성이 제거됨

```
<body>
  <ul id="app">
    <li>{{message}}</li>
    <li v-text="message"></li>
    <li v-html="message"></li>
    <li v-once>{{message}}</li>
  </ul>

  <script>
    var app = new Vue({
      el: '#app',
      data: {
        message: `<li>Hi Vue</li>`
      }
    });
  </script>
</body>
```

The screenshot shows the Vue DevTools Inspector interface. At the top, it says "Ready. Detected Vue 2.6.12." Below that, the component tree shows a single root node: <Root> - \$vm0. To the right, under "data", there is a message object with a value of "Hi Vue". A red box highlights this value. An arrow points down from this state to the resulting list of items.

- Hi Vue
- Hi Vue
- Hi Vue
- Hi Vue
- 안녕 Vue!
- 안녕 Vue!
- 안녕 Vue!
- Hi Vue

3) v-bind (단방향 바인딩)

Vue.js

* v-bind

html의 속성인 id, class, style 등에 model값을 연결할 때 사용하는 단방향 바인딩이다.
연동하려는 대상에 따라 v-bind:id, v-bind:class, v-bind:style 등을 사용할 수 있으며
축약형으로 v-bind를 생략할 수 있다.

The diagram illustrates the binding process between Vue.js code and generated HTML. A dashed blue box encloses the Vue.js code on the left, which includes a CSS block, a script block with a 'model' object, and a 'script' block with a 'new Vue' instance. A dashed red box encloses the generated HTML on the right, which shows the final rendered state of the components. A callout points from the 'list' key in the 'model' object to its corresponding 'id' attribute in the generated HTML's

 element. Another callout points from the 'red-accent' class in the CSS to its corresponding 'class' attribute in the generated HTML's - element. A third callout points from the 'color:blue' style in the CSS to its corresponding 'style' attribute in the generated HTML's - element. A fourth callout points from the 'link' key in the 'model' object to its corresponding element in the generated HTML.

```
<style>
  .red-accent {
    color: red;
  }
</style>

<script>
let model = {
  [mid: "list",]
  mclass: "red-accent",
  mstyle: "color:blue",
  link: {
    to: "http://google.com",
    title: "구글로",
  },
};
let vue = new Vue({
  el: "#app",
  data: model,
});
</script>
```

```
<div id="app">
  <ul v-bind:id="mid">
    <li v-bind:class="mclass">클래스 바인딩</li>
    <li :style="mstyle">스타일 바인딩</li>
    <li><a :href="link.to" :title="link.title">가자</a></li>
  </ul>
</div>
```

생성된 html

```
▼<div id="app">
  ▼<ul id="list">
    <li class="red-accent">클래스 바인딩</li>
    <li style="color: blue;">스타일 바인딩</li>
    <li>
      <a href="http://google.com" title="구글로">가자</a>
    </li>
  </ul>
</div>
```

3) v-bind (단방향 바인딩)

Vue.js

html의 class 속성의 경우 여러 값이 설정될 수 있는데, Vue에서 v-bind를 통해서도 당연히 여러 값을 설정할 수 있다.

이 경우 객체형태로 값을 전달하는데 boolean 타입의 model값을 이용해서 해당 class를 on/off 처리할 수 있다.

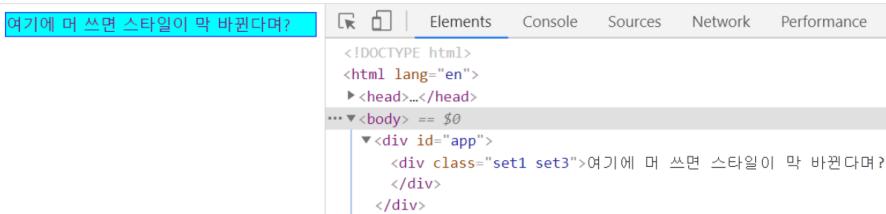
```
<div id="app">
  <div :class="{set1: s1, set2: s2, set3: s3}">
    여기에 머 쓰면 스타일이 막 바뀐다며?
  </div>
</div>
```

```
<script>
let vm = new Vue({
  el : "#app",
  data : {
    s1 : true,
    s2 : false,
    s3 : true,
  },
});
</script>
```

```
<style>
.set1 {
  background: aqua;
  color: purple;
}

.set2 {
  text-decoration: red;
  font-style: italic;
}

.set3 {
  border: 1px solid blue;
}
</style>
```



* v-model

View와 Model 사이의 양방향 데이터 바인딩에 사용되는 디렉티브이다.

일반적으로 사용자로부터 값을 입력 받는 <input>, <select>, <textarea> 등에서 사용되며 사용자가 입력한 값을 Model에 반영시켜준다.

당연히 Model이 변경되면 입력 요소의 값도 변경된다.

다중 선택이 지원되는 <select>나 checkbox 경우의 값은 배열 객체와 연결되고 나머지 요소들은 단일 값과 연결된다.

V-model에서는 특별한 기능을 추가하기 위한 수식어가 제공된다.

수식어	설명
lazy	입력 폼에서 change 이벤트가 발생하면 데이터 동기화
number	입력 값을 parseInt 또는 parseDouble를 이용해 number 타입으로 저장
trim	입력 문자열의 앞 뒤 공백을 자동으로 제거

이러한 수식어는 ‘.’를 이용해서 설정하며 여러 속성을 chaining 할 수도 있다.

4) v-model (양방향 바인딩)

Vue.js

* v-model 수식어

```
<input v-model.lazy="lazy" type="text" />
```

입력 후에 Enter 치면
반영.

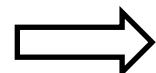
Lazy

AAAB

Lazy

AAAB

Lazy:



Lazy: AAAB

```
<input v-model.number="number" type="text" />
```

Number

1111asdfasf

숫자 입력 후에 문자 입력하면 반영 안됨

Number: 1111

Type of Number: number

```
<input v-model.trim="trim" type="text" />
```

자동으로 공백 제거

Trim

aaaaaa

Trim: "aaaaaa"

4) v-model (양방향 바인딩)

Vue.js

```
<div id="select">
  <label>가장 좋아하는 캐릭터는?</label>
  <input type="text" v-model.trim="favorite"><br>
  <label>팀 멤버를 선택하세요.</label><br>
  <input type="checkbox" value="1" v-model="team"><label>아이언맨</label>
  <input type="checkbox" value="2" v-model="team"><label>토르</label>
  <input type="checkbox" value="3" v-model="team"><label>헐크</label><br>
  <label>출생 연도는?</label>
  <input type="text" v-model.lazy.number="year">
</div>

<div id="result">
  <p>당신의선택은
    <ul>
      <li>캐릭터: {{favorite}}</li>
      <li>팀: {{team}}</li>
      <li>출생년도: {{year}}</li>
    </ul>
  </p>
</div>

<script>
  let model = { favorite : "", team : [], year: "" }

  let select = new Vue({el : "#select", data : model });
  let result = new Vue({el : "#result", data : model });
</script>
```

가장 좋아하는 캐릭터는? 아이언맨

팀 멤버를 선택하세요.

아이언맨 토르 헐크

출생 연도는? 2018

당신의선택은

- 캐릭터: 아이언맨
- 팀: ["2"]
- 출생년도: 2018

* v-show , v-if

조건에 따라서 랜더링 여부를 결정하는 기능이다.

	v-if	v-show
조건 불일치 시 렌더링 방식	렌더링하지 않음	display=none으로 처리
이에 따른 비용	토글 비용이 큼	초기 렌더링 비용이 큼
유용한 경우	자주 바뀌지 않을 때	자주 바뀔 때

* v-if , v-else, v-else-if

```
<div id="info">
  <label>나이를 입력하세요.</label>
  <input type="text" v-model="age" />
  
  
  당신은
  <span v-if="age < 0">입력 오류</span>
  <span v-else-if="age < 8">미취학</span>
  <span v-else-if="age < 19">미성년</span>
  <span v-else>성년</span>
</div>
```

```
<script>
  let model = {age: ""}

  let info = new Vue({el : "#info", data : model });
</script>
```

나이를 입력하세요. 15

당신은 미성년

나이를 입력하세요. 5

```
<input type="text">
<!-->

```

나이를 입력하세요. -5 ▲▲

```
<input type="text">


```

* 실습

다음과 같은 유효성 체크 코드를 구현하는 어플리케이션을 작성 하시오.

- 이름은 '3글자 이상'
- 나이는 '10세 이상이고 120세 이하'

처음 실행시 화면

이름을 입력하세요	<input type="text"/>	이름은 3글자 이상입니다.
나이를 입력하세요	<input type="text"/>	나이는 10세 이상, 120 이하만 가능합니다.

유효성 검증 통과시 화면

이름을 입력하세요	<input type="text" value="홍길동"/>
나이를 입력하세요	<input type="text" value="20"/>

유효성 검증 미통과시 화면

이름을 입력하세요	<input type="text" value="홍길"/>	이름은 3글자 이상입니다.
나이를 입력하세요	<input type="text" value="20"/>	

* v-for

문법:

```
v-for="항목 in 배열 | 객체" :key="값"
```

배열인 경우:

```
< v-for="contact in contacts" :key="유일값" >
```

```
< v-for="(contact , index) in contacts" :key="유일값" >
```

객체인 경우:

```
v-for="(val, key) in regions"
```

```
v-for="(val, key, index) in regions"
```

6) 반복문을 위한 디렉티브

Vue.js

```
<div id="app">
  <h3>객체 데이터 표현</h3>
  <ul>
    <li v-for="(value, key, index) in ironMan" :key="key">{{index}}, {{key}}, {{value}}</li>
  </ul>
<h3>배열 데이터 표현</h3>
<ul>
  <li v-for="(value, key) in heroes" v-if="key %2==0">{{key}}, {{value}}</li>
</ul>
</div>
```

유니크한 속성명을 이용해서 :key 설정

당연히 조건문과의 연계도 가능

객체 데이터 표현

- 0, name, 토니스타크
- 1, nickName, 깡통

배열 데이터 표현

- 0, 아이언맨
- 2, 토르
- 4, 비전

```
<script>
let model = {
  heroes : [ "아이언맨", "헐크", "토르", "캡틴아메리카", "비전" ],
  ironMan : {
    name : "토니스타크",
    nickName : "깡통"
  }
}

let app = new Vue({
  el : "#app",
  data : model
});
</script>
```

* template 태그

<template>는 여러 요소를 묶어서 반복 랜더링할 때 요소들을 묶어주는 태그이다.
실제 랜더링 내용에는 포함되지 않으며 단지 요소들을 그룹핑하는 용도로만 사용된다.

```
<div>
  <template v-for="(value, key) in heroes">
    <div>
      <p>등장 순서: {{key}}, 이름:{{value}}</p>
    </div>
    <hr v-if="key %2==0">
  </template>
</div>
```

등장 순서: 0, 이름:아이언맨

등장 순서: 1, 이름:헐크

등장 순서: 2, 이름:토르

등장 순서: 3, 이름:캡틴아메리카

등장 순서: 4, 이름:비전

* 실습 2 (v-model + 배열) 활용

다음과 같은 데이터를 활용하여 선택된 교재명을 출력하는 코드를 작성 하시오.

```
bookList:[  
  {name:'자바의 정석',price:2000},  
  {name:'JSP의 정석',price:3000},  
  {name:'Spring의 정석',price:1500},  
  {name:'jQuery의 정석',price:1000},  
  {name:'Angular의 정석',price:5000}  
],
```

교재 정보

- 자바의 정석 2000
- JSP의 정석 3000
- Spring의 정석 1500
- jQuery의 정석 1000
- Angular의 정석 5000

선택교재:

교재 정보

- 자바의 정석 2000
- JSP의 정석 3000
- Spring의 정석 1500
- jQuery의 정석 1000
- Angular의 정석 5000

선택교재:
JSP의 정석
Angular의 정석



4장. Vue 객체



Vue 객체

옵션 객체 속성

Methods, computed, watch 함수

라이프 사이클 메서드

1) Vue 객체

Vue.js

Vue 객체는 앞서 이야기 했듯이 MVVM 패턴에서 ViewModel을 담당하는 Vue.js의 핵심 객체이다. 일반적으로 Vue 생성자 함수에 옵션 객체를 전달해서 생성한다.

```
let vi = new Vue(  
  {  
    el : '#app',  
    data : model  
  }  
)
```

속성 명	설명
el	Vue로 만든 화면이 그려지는 인스턴스의 시작 지점 - CSS 선택자로 요소 지정(여러 개가 선택되더라도 맨 처음 요소만 사용됨 → ID 기반 접근 필요)
data	인스턴스의 데이터 속성으로 객체를 이용해 여러 값 저장
template	화면에 표시할 HTML, CSS등 마크업 요소를 정의하는 속성 - Vue의 데이터 및 기타 속성들도 함께 화면에 렌더링
methods	이벤트 및 화면 동작 메서드로 속성들은 function
created...	라이프 사이클 커스터마이징 로직

이러한 속성들은 Vue객체 외부에서 접근하기 위해서는 \$options 속성을 이용한다.

2) Vue 객체의 옵션객체 속성

Vue.js

다음은 Vue객체의 \$options 값을 콘솔에 출력한 예이다.

```
<script>
  var app = new Vue({
    el:'#app'
  });
  console.log("Vue 객체:", app)
</script>
```



```
Vue 객체:
  ▶ Vue {_uid: 0, _isVue: true, $options: {...}, _renderProxy: Proxy, _self: Vue, ...} ⓘ
    ▷ $attrs: ...
    ▷ $children: []
    ▷ $createElement: f (a, b, c, d)
    ▷ $el: div#app
    ▷ $listeners: ...
  ▷ $options:
    ▷ components: {}
    ▷ directives: {}
    ▷ el: "#app"
    ▷ filters: {}
    ▷ render: f anonymous( )
    ▷ staticRenderFns: []
    ▷ _base: f Vue(options)
    ▷ __proto__: Object
  ▷ $parent: undefined
  ▷ $refs: {}
  ▷ $root: Vue {_uid: 0, _isVue: true, $options: {...}, _renderProxy: Proxy, _self: Vue, ...}
  ▷ $scopedSlots: {}
  ▷ $slots: {}
  ▷ $vnode: undefined
```

el 속성은 element를 나타내며 View를 연결하는 속성이다.

el을 지정할 때는 CSS 선택자를 이용해서 HTML의 DOM속성을 지정한다.

이때 반드시 하나만 지정되어야 한다. 따라서 일반적으로 ID속성을 지정하는 #이 사용된다.

만약 Vue객체 외부에서 el 요소에 접근하기 위해서는 \$el 속성을 이용한다.

```
<ul id="app">
  <li>{{message}}</li>
</ul>

<script>
  let model={
    message:"Hi Vue"
  }

  let vi = new Vue({
    el:"#app",
    data:model
  });

  console.log(vi.$el);
</script>
```

\$el을 통해 el에 접근 가능

➢ vi.\$el
↳ ➢ <ul id="app">...

data는 Model을 연결하는 속성으로 JSON 객체이다.

Vue 객체 외부에서 data에 접근하기 위한 내장옵션으로 \$data를 사용할 수 있는데, 좀 더 편리하게 사용하기 위해서 data의 속성들은 모두 Vue 객체의 속성으로 관리된다.

```
<ul id="app">
  <li>{{message}}</li>
</ul>

<script>
let model={
  message:"Hi Vue"
}

let vi = new Vue({
  el:"#app",
  data:model
});

console.log(model.message, vi.message, vi.$data.message);
</script>
```

Hi Vue Hi Vue Hi Vue
모두 같은 내용

5) methods

Vue.js

methods는 Vue 객체에서 사용할 메서드들을 객체로 등록하는 속성이다.

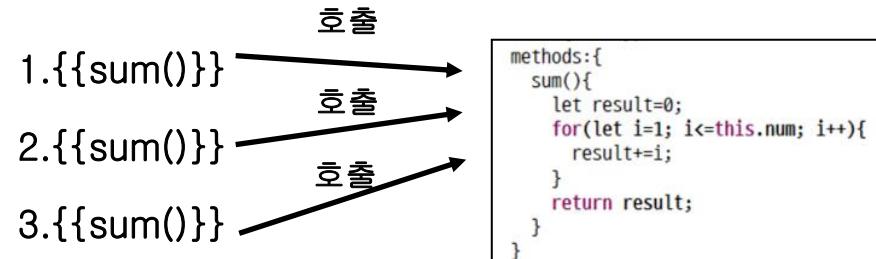
등록된 메서드는 Vue 객체에서 직접 호출하거나 디렉티브 표현식 등에서 data처럼 사용이 가능하다.

주의할 점은 호출할 때 반드시 ()를 이용해야 되며, 화면이 다시 랜더링 될 때마다 화면에서 사용되고 있는 method들은 모두 다시 실행된다.

즉, 호출 시점에 메서드가 실행되어 값을 반환하기 때문에 캐시 기능이 없다.

```
<div id="app">  
  1부터 <input type="number" v-model="num"/> 의 합은 {{sum()}}  
</div>  
  
<script>  
  let vi = new Vue({  
    el:"#app",  
    data:{num:0},  
    methods:{  
      sum(){  
        let result=0;  
        for(let i=1; i<=this.num; i++){  
          result+=i;  
        }  
        return result;  
      }  
    }  
  });  
</script>
```

1부터 [5]의 합은 10



6) computed

“계산된” 속성인 computed는 함수를 속성으로 갖는 객체이다.

methods 속성과는 다르게 생성시점에 계산된 값을 반환하고 이후부터는 캐시값을 사용한다.

주의할 점은 호출시 반드시 ‘함수명’만 사용한다.

```
<ul id="app">
  <li>{{message}}</li>
  <li>{{sum}}</li>
</ul>
<script>
  let model={ message:"Hi Vue"}
  let vi = new Vue({
    el:"#app",
    data:model,
    computed:{
      sum(){
        let result = 0;
        for(let i=1; i<=10; i++){
          result+=i;
        }
        return result;
      }
    }
  });
</script>
```

sum() 처럼 실행하지 않아도 동작
data의 message 처럼 사용

1. {{sum}} → 호출
2. {{sum}} 캐시사용
3. {{sum}} 캐시사용

```
computed:{
  sum(){
    let result = 0;
    for(let i=1; i<=10; i++){
      result+=i;
    }
    return result;
  }
}
```

7) watch

watch도 computed, methods와 유사하게 함수들을 관리하는 객체이다.

주요 기능은 데이터 변경시 특정 동작 처리가 가능한 함수들을 관리한다.

주의할 점은 watch함수를 등록할 때 함수의 이름은 반드시 변경을 모니터링할 객체의 속성명과 같아야 된다.

함수 파라미터값으로 old값과 new값을 전달 받을 수 있으며 주로 긴 시간이 필요로 하는 비동기 처리에 적합하다.

```
<div id="app">
  <input type="number" v-model="x">+<input type="number" v-model="y">={{sum}}
</div>
<script>
  let vi = new Vue({
    el:"#app",
    data:{x:0, y:0, sum:0},
    watch:{
      x(nv){
        console.log("x 변경됨!!!");
        this.sum = parseInt(nv)+parseInt(this.y);
      },
      y(nv){
        console.log("y 변경됨!!!");
        this.sum = parseInt(this.x)+ parseInt(nv);
      }
    });
</script>
```

8) methods vs computed vs watch

methods, computed, watch는 함수를 사용한다는 측면에서는 모두 비슷하다. 하지만 실제 동작은 다음과 같이 다르기 때문에 차이점에 주의해서 적절한 속성을 사용해야 된다.

	computed	methods	watch
처리방식	동기	동기	비동기
결과 리턴	O	O / X	X
함수 호출	화면에서 속성 선언 시 호출됨	화면에서 함수 사용 시 호출됨	참조 모델 변경 시 자동 호출됨
결과 캐싱	O	X	X
일반적으로	모델이 자주 바뀌지 않는 짧은 작업	모델이 자주 바뀌는 짧은 작업	시간이 걸리는 비동기 작업에 유리

filters 속성은 {{}} 표현식 또는 v-bind에서 텍스트의 형식화 하는 기능을 제공한다.

```
new Vue({  
  el: "#app",  
  data: {  
    name: 'hong',  
    price: 10000000,  
  },  
  filters: {  
    toCap(name) {  
      return name.toUpperCase();  
    }  
  }  
})
```

```
<span :id="name | toCap">대문자로</span>
```

전역적으로 사용

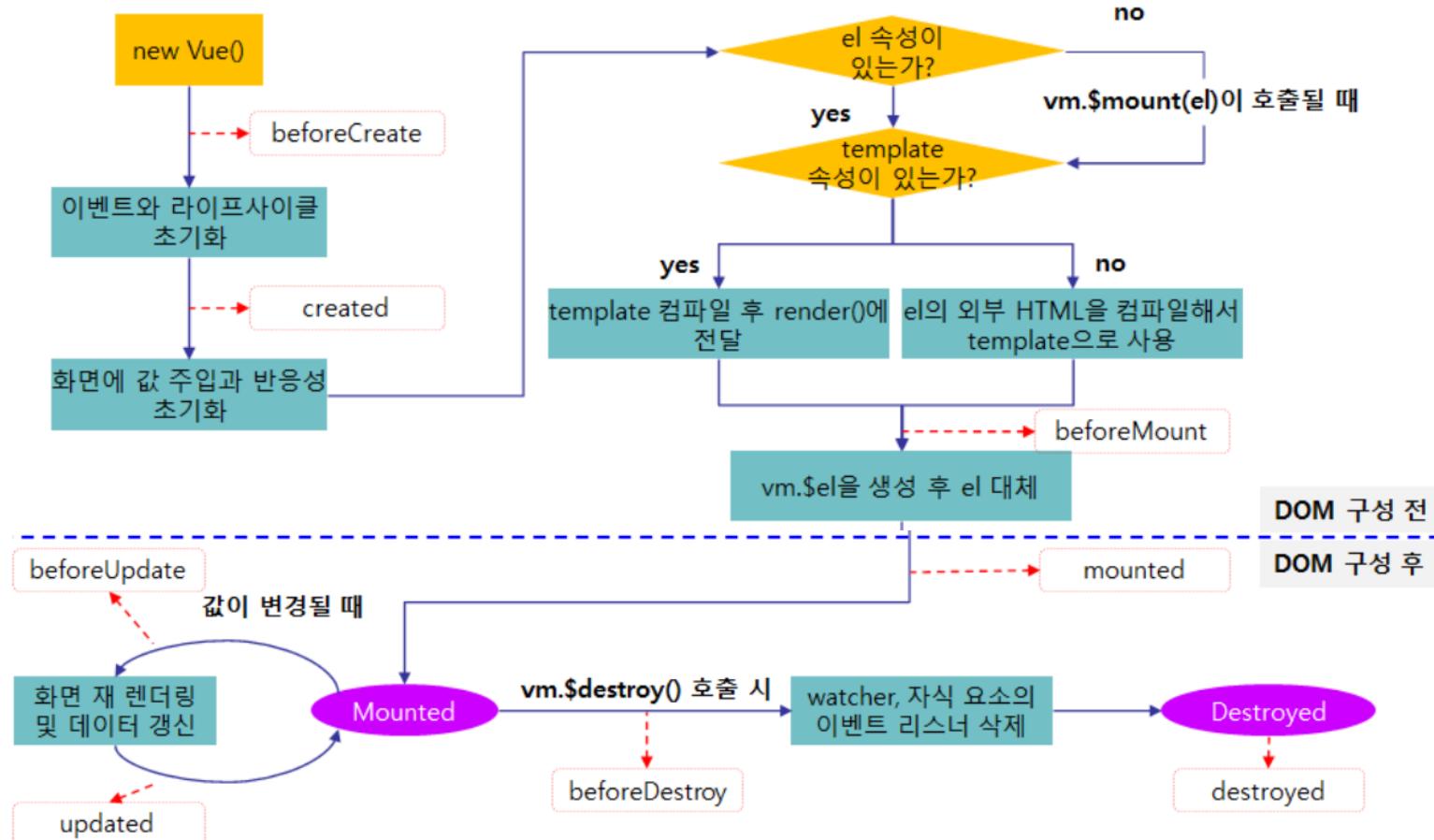
```
Vue.filter("filter_name", callback_function);
```

```
Vue.filter("toCurrency", (val) => {  
  return new Intl.NumberFormat("ko-KR", { style: "currency", currency: "KRW" }).format(val);  
});
```

10) Vue 라이프 사이클 혹은 메서드 (hook method)

Vue.js

Vue 객체는 생성에서부터 화면연결, 값 변경에 대한 처리등 다양한 일들에 관여하다가 소멸된다. 특정 시점에 콜백되는 라이프 사이클 혹은 메서드는 다음과 같다.



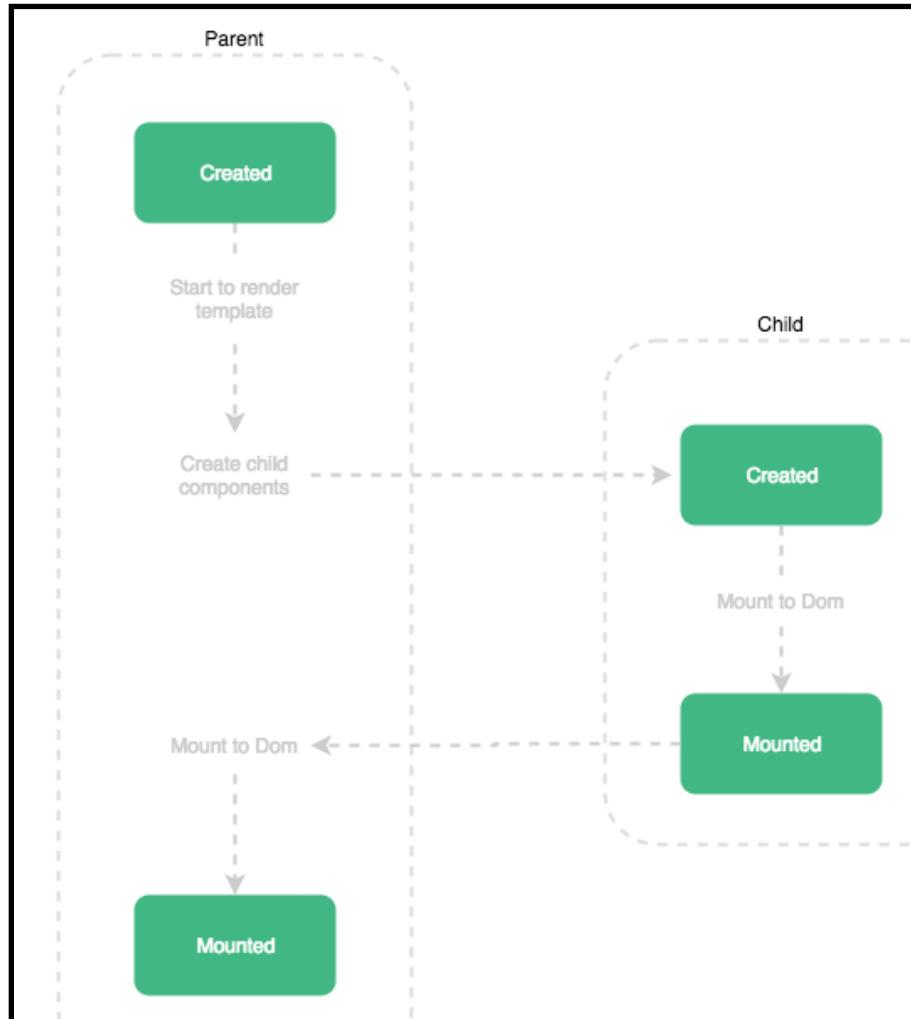
<https://vuejs.org/v2/guide-instance.html#Instance-Lifecycle-Hooks>

- `beforeCreate`
 - Vue 객체가 생성되고 데이터에 대한 관찰 기능 및 이벤트 감시자 설정 전에 호출된다.
 - 사실상 거의 볼 일이 없다. ㅜㅜ
- `created`
 - Vue 객체가 생성된 후 **데이터에 대한 관찰이 가능하다.**
 - `computed`, `methods`, `watch` 설정이 완료된 후이다.
- `beforeMount`
 - 아직 마운트가 시작되기 전에 호출된다.
 - 따라서 DOM 구성 이전이므로 **화면을 건드리기는 부적절하다고 볼 수 있다.**
- `mounted`
 - DOM 요소가 `el`에 의한 가상 DOM으로 대체된 후 호출된다.
 - 드디어 **화면에 대한 완벽한 제어가 가능해지는 시점이다.**

- `beforeUpdate`
 - 데이터가 변경되서 가상 DOM이 다시 렌더링 되기 전에 호출된다.
- `updated`
 - 데이터 변경으로 가상 DOM이 다시 렌더링 되고 패치된 후에 호출된다.
 - 이 흙이 호출된 시점은 이미 가상 DOM이 렌더링 된 후이므로 DOM에 대한 추가 작업이 가능하다.
- `beforeDestroy`
 - Vue 객체가 제거되기 전에 호출된다.
- `destroyed`
 - Vue 객체가 제거된 후에 호출된다.
 - 이 흙이 호출될 때 Vue 객체의 모든 디렉티브 바인딩이 해제되고 이벤트 연결도 제거된다.

10) Vue 라이프 사이클 혹은 메서드 (hook method)

Vue.js



-부모의 mounted hook이 자식의 mounted hook 보다 늦게 실행된다.
즉 부모는 자식의 mounted hook이 끝날 때까지 기다린다.

<https://medium.com/@brockreece/vue-parent-and-child-lifecycle-hooks-5d6236bd561f>

5장. 이벤트 처리



이벤트 처리

이벤트 객체

이벤트 수식어

1) 이벤트 처리

Vue.js

Vue에서 DOM 요소에 이벤트를 등록하기 위해서는 v-on:event_name=이벤트_핸들러 또는 축약형으로 @event_name=이벤트_핸들러를 사용한다.

Vue에서 처리할 수 있는 이벤트의 종류는 일반적인 DOM 객체의 이벤트 처리와 동일하다.

```
<div id="app">
    현재 잔고: <span>{{balance}}</span>
    <label>거래 금액</label><input type="number" v-model="amount">
    <button v-on:click="balance+=parseInt(amount)">입금</button>
    <button @click="withdraw">출금</button>
</div>

<script>
    let vi = new Vue({
        el: "#app",
        data: {
            amount: 0, balance: 10000000
        },
        methods: {
            withdraw() {
                if(this.balance >= this.amount) {
                    this.balance -= this.amount;
                } else {
                    alert("잔액 부족");
                }
            }
        });
    </script>
```

인라인 스타일: 복잡한 스크립트 작성의 한계

이벤트 핸들러 메서드 등록(또는 실행)

2) 이벤트 객체

DOM에서 발생하는 이벤트의 상세 정보(어떤 요소에서 발생했는지, 눌린 키값은 무엇인지등)는 이벤트 객체를 통해서 전달된다.

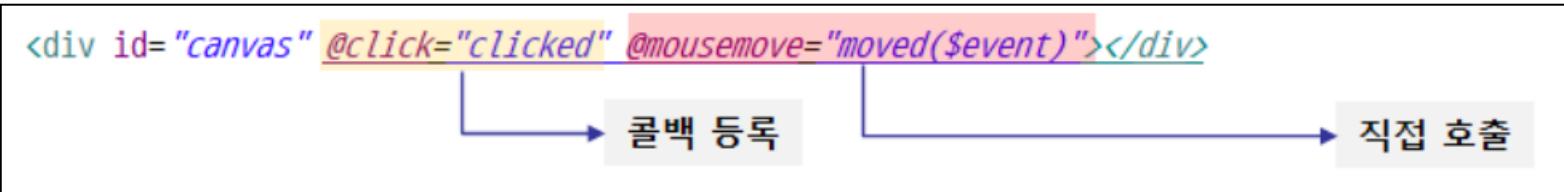
다음은 일반적으로 이벤트 객체에서 뽑아서 사용할 수 있는 속성/함수들이다.

구분	속성명	설명
공통	target	이벤트 소스(이벤트가 발생한 DOM 객체)
	path	배열로 target 부터 window까지 조상을 찾아가는 경로
키보드 이벤트	altKey, shiftKey, ctrlKey	각각 alt, shift, ctrl 키가 눌렸는지 여부(true/false)
	keyCode	이벤트를 발생시킨 키보드의 고유 키 코드(enter: 13 등)
	charCode	keyPress 이벤트에서 눌린 unicode 캐릭터 코드
마우스 이벤트	altKey, shiftKey, ctrlKey	키보드 이벤트 참조
	clientX, clientY	viewport 영역에서 이벤트가 발생한 좌표로 스크롤된 길이에 영향을 받지 않음
	pageX, pageY	document 영역에서 이벤트가 발생한 좌표로 스크롤된 길이에 영향을 받음
	screenX, screenY	screen 영역에서 이벤트가 발생한 좌표
공통함수	preventDefault	기본 이벤트 동작을 중지시킴
	stopPropagation	이벤트 전파를 막음

3) 이벤트 핸들러 호출 방식

Vue.js

Vue에서 이벤트 핸들러를 등록할 때는 단지 핸들러를 등록하거나 직접 호출하는 방식 모두 가능하다.



이때 차이점은 실행하는 형태는 직접 함수를 호출하는 것이고 시스템이 콜백하지 않는다. 따라서 이벤트 객체를 직접 활용할 수 없기 때문에 `$event`를 이용해서 명시적으로 전달해야 된다. 반대로 이벤트 핸들러를 등록만 하는 형태는 시스템이 콜백하는 형태이기 때문에 파라미터로 이벤트 객체가 자동으로 전달된다.

```
<div id="app">
  <div id="canvas" @click="clicked" @mousemove="moved($event)"></div>
  <input type="text" name="event" id="name" v-model="status" />
  <input type="text" name="x" id="x" v-model="x" />
  <input type="text" name="y" id="y" v-model="y" />
</div>
```

```
<script>
let vi = new Vue({
  el: "#app",
  data: {
    x: 0,
    y: 0,
    status: "이벤트",
  },
  methods: {
    clicked(e) {
      this.x = e.clientX;
      this.y = e.clientY;
      this.status = "클릭";
    },
    moved(e) {
      this.x = e.clientX;
      this.y = e.clientY;
      this.status = "이동";
    },
  },
});
</script>
```

4) 이벤트 수식어

Vue.js

Vue에서는 앞서 살펴본 이벤트 속성을 이용할 때 직접 이벤트 객체를 사용하기 보다는 간단한 이벤트 수식어(Event Modifier)를 이용해서 처리할 수 있다.

다음은 Vue에서 자주 사용되는 공통 이벤트 수식어들이다.

구분	수식어 명	설명
공통	prevent	이벤트의 기본 동작(<form>의 submit → 전송, <a>의 클릭 → 페이지 전환 등) 방지 preventDefault() 대체
	stop	하나의 이벤트가 여러 요소에 걸쳐 발생할 때 전파 중단 stopPropagation() 대체
	capture	캡처링 상태에서 이벤트 처리
	self	버블링(raising) 상태에서 이벤트 처리
	once	한번만 이벤트를 발생시키고 이후는 동작하지 않음
	Passive	혹시나 있을지 모를 preventDefault()를 실행 안되게 함(동작 보장)

구분	수식어 명	설명
키보드 이벤트	숫자	입력되는 키에 대한 키 코드 입력 값(대표적인 키들은 상수로 등록됨) 제한 enter, tab, delete, esc, space, up, down, left, right, ctrl, alt, shift 조합 키의 경우 수식어 연결 사용 @keyup.ctrl.67="event_name"
마우스 이벤트	left, right, middle	각각 마우스의 어떤 버튼이 클릭 됐는지로 제한

4) 이벤트 수식어

Vue.js

```
<div id="app">
  <form action="#" @submit.prevent="login">
    ID:<input type="text" placeholder="ID는 4글자 이상입니다." v-model="id">
    <input type="submit">
  </form>
</div>

<script>
let vi = new Vue({ el:"#app", data:{ id:""}, methods:{ login(e){ if(this.id.length>4){ e.target.submit(); }else{ alert("ID를 4글자 이상으로 넣어주세요."); } } });
</script>
```

validation이 필요한 상황에서 기본 전송 동작 중지 필요

5) 화면에서 다른 요소 참조

DOM 이벤트 처리시 화면의 다른 DOM 요소를 참조하는 경우가 있는데, 일반적으로 `document.querySelector("#target")`와 같이 DOM 요소를 지정할 수 있다. 하지만 Vue에서는 DOM에 직접 접근하는 일이 거의 없으며 `ref` 속성으로 DOM 요소에 표시하고 Vue 객체 내부에서는 `$refs` 속성으로 접근한다.

```
<div id="app">
  <form @submit.prevent="login">
    <label for="id">아이디</label>
    <input type="text" name="id" id="id" @keyup.prevent.enter="next" v-model.lazy="id" />
    <input type="password" ref="pass" @keyup.enter="search" />
    <input type="button" value="제출" />
  </form>
</div>

<script>
let vi = new Vue({
  el: "#app",
  data: {
    id: "",
  },
  methods: {
    next() {
      let pass = this.$refs.pass;
      console.log(pass);
      pass.focus();
    },
    login() {
      console.log("로그인 처리");
    },
    search() {
      console.log("id: " + this.id);
    },
  },
});
```

* 실습 3

실습문제 2의 데이터를 사용하여 개별삭제 기능과 전체 삭제 기능을 구현하는 프로그램을 작성 하시오.

교재 정보

- 자바의 정석2000 삭제
- JSP의 정석3000 삭제
- Spring의 정석1500 삭제
- jQuery의 정석1000 삭제
- Angular의 정석5000 삭제

전체 삭제

‘자바의 정석’ 삭제시



교재 정보

- JSP의 정석3000 삭제
- Spring의 정석1500 삭제
- jQuery의 정석1000 삭제
- Angular의 정석5000 삭제

전체 삭제

교재 정보

- JSP의 정석3000 삭제
- Spring의 정석1500 삭제
- jQuery의 정석1000 삭제
- Angular의 정석5000 삭제

전체 삭제

‘Spring 정석’ 과
‘Angular의 정석’
선택후에 [전체삭제]
클릭시

교재 정보

- JSP의 정석3000 삭제
- jQuery의 정석1000 삭제

전체 삭제

6장. 컴포넌트



컴포넌트 개요

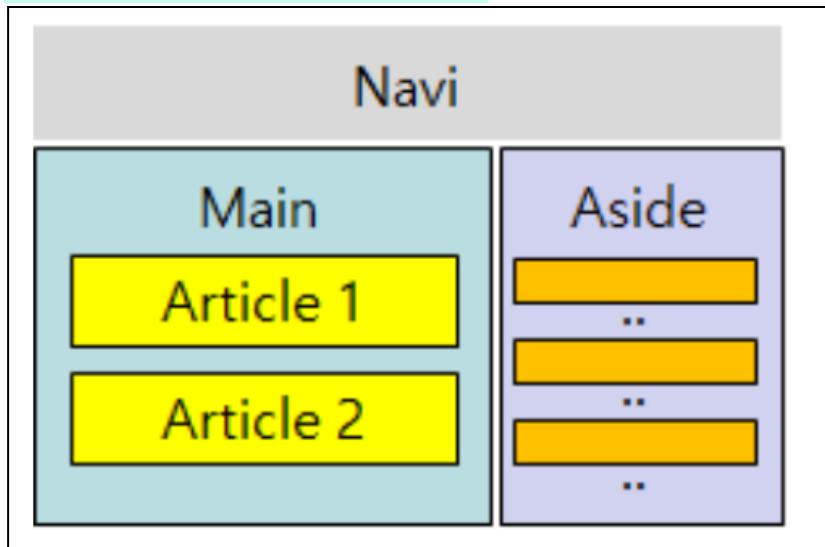
전역 컴포넌트/지역 컴포넌트

1) 컴포넌트 (component)

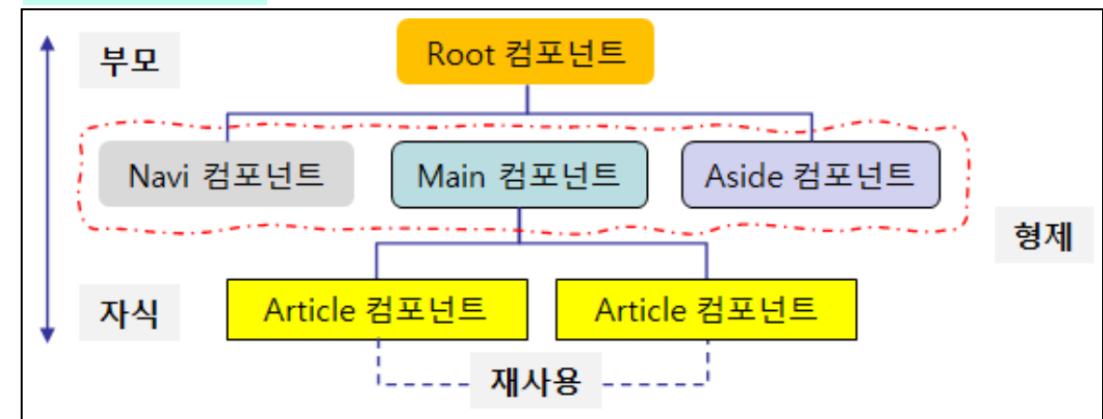
Vue.js

컴포넌트는 일반적으로 재사용 가능한 블록 형태의 요소를 의미한다.
Vue에서의 컴포넌트도 조합해서 화면을 구성할 수 있는 블록을 의미하며
재사용성의 강화가 목적이다.
SPA(Single Page Application) 적용시 컴포넌트 사용은 필수이다.

전통적인 화면 구조



컴포넌트



2) 등록 방식에 따른 컴포넌트의 종류

Vue.js

컴포넌트는 등록 방식에 따라 지역 컴포넌트와 전역 컴포넌트로 구분된다.

지역 컴포넌트는 특정 Vue 객체에서만 유효한 범위를 가지고, 전역 컴포넌트는 여러 Vue 객체에서 공통으로 사용 가능하다.

전역 컴포넌트 등록

```
Vue.component(tagName, options)
```

```
<body>
<div id="my">
  <my-component></my-component>
  <your-component></your-component>
</div>

<template id="body-template">
  <div>body 영역에 정의한 template</div>
</template>
</body>

<script>
Vue.component("my-component", {
  template: "<div>Hello Component</div>",
});

Vue.component("your-component", {
  template: "#body-template",
});

let vi = new Vue({
  el: "#my",
});
</script>
```

지역 컴포넌트 등록

```
new Vue(
  components: {
    'component-name': 'component 내용'
  }
);
```

```
<div id="app">
  <button>컴포넌트 등록</button>
  <my-local-component></my-local-component>!-- 컴포넌트 --
</div>

<script>
new Vue({
  el: '#app',
  components: {
    'my-local-component': {
      template: '<div>local component 등록 방식</div>'
    }
  }
});
</script>
```

전역 컴포넌트는 Vue 객체의 component 속성을 이용해서 등록한다.

```
Vue.component(tagName, options)
```

tagName

- 컴포넌트의 이름으로 사용자 정의 HTML 태그 이름이 된다.
- tagName은 모두 소문자이며 캐밥 표현식(단어의 연결은 하이픈 사용:my-component)을 적용한다.
- 일반적인 HTML태그와 다르게 빈 태그라도 <my-component /> 식으로 사용하지 않는다.

options

- 컴포넌트의 속성을 설정하는 옵션 객체로 template, data 등이 선언된다.
- template는 화면에 보여줄 모습인 html태그를 작성한다.
- data는 Vue객체에서 사용될 data를 선언한다.

3) 전역 컴포넌트

Vue.js

```
<body>
  <div id="my">
    <my-component></my-component>
    <your-component></your-component>
  </div>

  <template id="body-template">
    <div>body 영역에 정의한 template</div>
  </template>
</body>

<script>
  Vue.component("my-component", {
    template: "<div>Hello Component</div>",
  });

  Vue.component("your-component", {
    template: "#body-template",
  });

  let vi = new Vue({
    el: "#my",
  });
</script>
```

```
<div id="my">
  <my-component></my-component>
  <your-component></your-component>
</div>
```



```
<div id="my">
  <div>Hello Component</div>
  <div>body 영역에 정의한 template</div>
</div>
```

1) template

반드시 root element가 존재하고 그 안에 다른 element가 나와야 된다.

```
<!-- 아래의 경우는 가능 -->
<template id="body-template">
  <div>
    <div></div>
    <div></div>
  </div>
</template>

<!-- 하지만 이런 경우는 불가능 -->
<template id="body-template">
  <div></div>
  <div></div>
</template>
```

2) data

컴포넌트의 options에 정의하는 data는 반드시 함수로 작성하고 함수 내부에서 리턴 되는 객체를 사용해야 된다.

```
/*data: {
  current: 0,
},*/
data: function () {
  return {
    current: 0,
  };
},
```

지역 컴포넌트는 Vue 객체를 구성하면서 components 속성을 추가해서 등록한다.

```
new Vue(  
  components: {  
    'component-name': 'component 내용'  
  }  
)
```

나머지 작성 방법은 전역 컴포넌트와 동일하다.

```
<div id="app">  
  <button>컴포넌트 등록</button>  
  <my-local-component></my-local-component><!-- 컴포넌트 -->  
</div>  
  
<script>  
  new Vue({  
    el:'#app',  
    components:{  
      'my-local-component': {  
        template: '<div>local component 등록 방식</div>'  
      }  
    }  
  });  
</script>
```

다음과 같은 화면UI를 위한 컴포넌트를 활용하여 작성 하시오.



App 컴포넌트

```
data:function(){
    return {
        bookList:[
            {id:'p01',name:'위험한 식탁',price:2000,date:'20170401',img:'a'},
            {id:'p02',name:'공부의 비결',price:3000,date:'20170402',img:'b'},
            {id:'p03',name:'오메르타', price:2500,date:'20170401',img:'c'},
            {id:'p04',name:'행복한 여행',price:4000,date:'20170401',img:'d'},
            {id:'p05',name:'해커스 토익',price:2000,date:'20170401',img:'e'},
            {id:'p06',name:'도로 안내서',price:2000,date:'20170401',img:'f'}
        ]
    }
}
```

7장. 컴포넌트 통신



컴포넌트 통신 개요

Props 속성

\$emit 사용자정의 이벤트

이벤트 버스(event bus)

1) 컴포넌트 (component) 통신

Vue.js

각각의 컴포넌트는 개별적으로 고유한 유효 범위를 가지기 때문에 다른 컴포넌트의 데이터를 직접 참조할 수 없다.

컴포넌트간의 자료 전달 방식은 그들간의 관계에 따라 다른 방식을 사용한다.

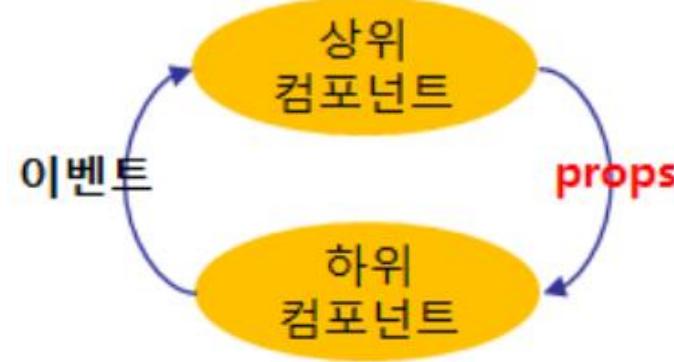
- 부모/자식 즉 상하위 컴포넌트의 관계
 - 상위 → 하위로의 전달은 `props`라는 속성 이용
 - 하위 → 상위로의 전달은 이벤트를 통해서 데이터 전달
- 상/하위 관계가 아닌 경우
 - 이벤트 버스 이용



2) 상위에서 하위로 데이터 전달 (props)

Vue.js

상위 컴포넌트의 데이터를 하위 컴포넌트에서 사용할 때는 하위 컴포넌트의 props라는 속성에 선언한 변수(속성명)에 상위 컴포넌트가 태그의 ‘속성명=값’ 형태로 전달한다.



```
<script>
  Vue.component('child-component', {
    // child-component에서 사용할 속성 정의
    props: ['propsdata'],
    template: '<p>이거 받았나: {{propsdata}}</p>'
  });

  new Vue({
    el: "#app",
    data: {
      message: '이거 가져다가 써보세요'
    }
  });
</script>
```

부모

자식

```
<div id="app">
  <!-- component의 propsdata 속성에 단순 값 할당 -->
  <child-component propsdata="직접 값을 줄 수도 있다."></child-component>

  <!-- component의 propsdata 속성을 바인딩 처리 -->
  <child-component v-bind:propsdata="message"></child-component>
</div>
```

2) 상위에서 하위로 데이터 전달 (props)

```
<div id="app">  
  <!-- component의 propsdata 속성에 단순 값 할당 →<br/>  <child-component propsdata="message"></child-component>  
  ↓  
  <!-- component의 propsdata 속성을 바인딩 처리 --&gt;<br/>  <child-component v-bind:propsdata="message"></child-component>  
</div>
```

하위 컴포넌트의 **propsdata**
라는 그릇에 값을 할당한다.
부모의 데이터 참조시 **v-bind** 사용

```
<script>  
  Vue.component('child-component', {  
    // child-component에서 사용할 속성 정의  
    props: ['propsdata'],  
    template: '<p>이거 받았나: {{propsdata}}</p>'  
  });  
  
  new Vue({  
    el: "#app",  
    data:{  
      message: '이거 가져다가 써보세요'  
    }  
  });  
</script>
```

자식 컴포넌트는 **props**를 통해
데이터를 받을 그릇을 준비한다.

실습 문제 5

Vue.js

다음과 같은 화면UI를 위한 컴포넌트를 활용하여 작성 하시오.
(부모인 App컴포넌트의 data를 자식인 book 컴포넌트로 전달하여 데이터 랜더링)



App 컴포넌트

```
//부모 컴포넌트
var app = new Vue({
  el:'#app',
  data:function(){
    return {
      bookList:[
        {id:'p01',name:'위험한 식탁',price:2000,date:'20170401',img:'a'},
        {id:'p02',name:'공부의 비결',price:3000,date:'20170402',img:'b'},
        {id:'p03',name:'오메르타', price:2500,date:'20170401',img:'c'},
        {id:'p04',name:'행복한 여행',price:4000,date:'20170401',img:'d'},
        {id:'p05',name:'해커스 토익',price:2000,date:'20170401',img:'e'},
        {id:'p06',name:'도로 안내서',price:2000,date:'20170401',img:'f'}
      ]
    }
});
```

<book-component> 컴포넌트

```
//자식 컴포넌트
const Feature = Vue.component('book-component',{
  template:`
    <div>
      
      <p>{{name}}</p>
      <p>{{price}} 원</p>
      <p>{{date}}</p>
    </div>
  `,
  props: [
    'book'
  ],
  data() {
    return {
      img: ''
    }
  },
  created() {
    this.$http.get(`./bookList/${this.book.id}`).then(response => {
      this.img = response.data
    })
  }
});
```

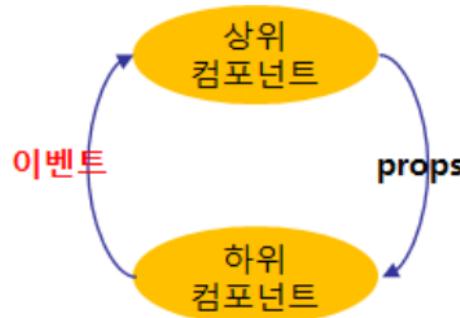
3) 하위에서 상위로 데이터 전달 (이벤트)

Vue.js

하위 컴포넌트의 데이터를 상위 컴포넌트로 전달하기 위해서는 사용자 정의 이벤트를 발생시켜서 데이터를 전달해야 된다.

하위 컴포넌트가 이벤트를 발생시키면 이 이벤트를 통해서 결국은 상위 컴포넌트의 함수를 동작시키는 형태이다.

이벤트를 발생시킬 때에는 Vue 객체의 \$emit 함수를 사용한다.



```
$emit("custom_event_name" [, param1, param2, ...])
```

하위 컴포넌트

```
Vue.component("child-comp", {  
  // 클릭하면 sendLog 호출  
  template: "<button @click='sendLog'>click</button>",  
  methods: {  
    sendLog() {  
      // 부모에게 receive_sendLog 발생  
      this.$emit("receive_send_log", "이거좀 보세요");  
    }  
  }  
})
```

상위 컴포넌트

```
let vi = new Vue({  
  el: "#app",  
  methods: {  
    printLog(msg) {  
      console.log("who call me: " + msg);  
    }  
  }  
}).  
<div id="app">  
  <!--이벤트 발생 시 printLog 호출-->  
  <child-comp @receive_send_log="printLog"></child-comp>  
</div>
```

3) 하위에서 상위로 데이터 전달 (이벤트)

Vue.js

```
<div id="app">  
  <!--이벤트 발생 시 printLog 호출-->  
  <child-comp @receive_send_log="printLog"></child-comp>  
</div>
```

3. log-event에 대한 리스너 print 실행

```
<script>  
  Vue.component("child-comp", {  
    template: "<button @click='sendLog'>click</button>", // 클릭하면 sendLog 호출  
    methods: {  
      sendLog() {  
        this.$emit("receive_send_log", "이거좀 보세요"); // 부 2. child 요소가 가지고 있는 log-evt 발생  
      }  
    }  
  })  
  let vi = new Vue({  
    el: "#app",  
    methods: {  
      printLog(msg) {  
        console.log("who call me" + msg);  
      }  
    }  
  });  
</script>
```

1. click 이벤트 리스너로 showLog 등록

2. child 요소가 가지고 있는 log-evt 발생

* 실습 6

실습문제 5의 코드에 기능을 추가하여
자식 컴포넌트의 도서 목록에서 특정
도서를 클릭할 때,
선택된 도서명을 부모 컴포넌트의
input 태그에 출력하는 코드를
작성 하시오.

도서 목록 6권

선택된 도서:

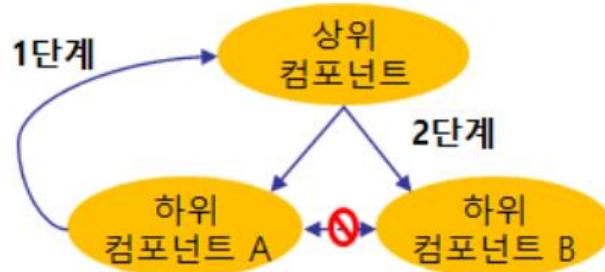
- 위험한 식탁
- 공부의 비결
- 오메르타
- 행복한 여행
- 해커스 토익
- 도로안내서

App 컴포넌트

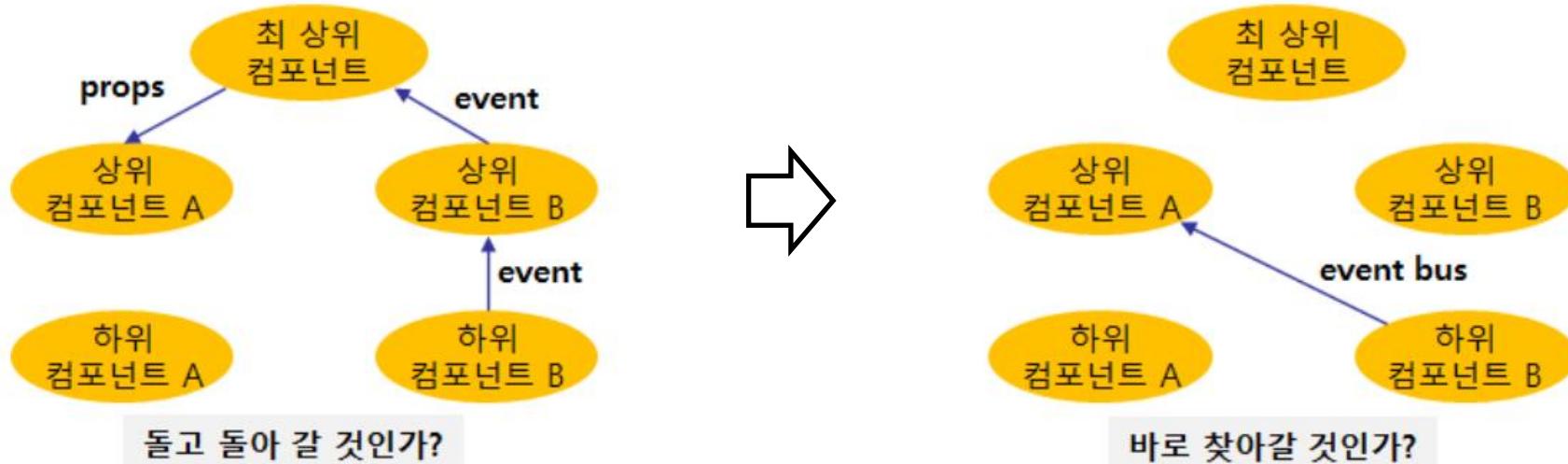
<book-component>
컴포넌트

4) 상/하위 레벨이 아닌 컴포넌트 통신 (이벤트 버스)

상/하위 레벨이 아닌 컴포넌트 간은 직접적인 정보 전달이 불가능하다.
정보를 전달하기 위해서는 1단계, 2단계 작업이 필요하다.



두 컴포넌트 간 직접적으로 이벤트를 연결하기 위해서 ‘이벤트 버스’를 이용한다.



4) 상/하위 레벨이 아닌 컴포넌트 통신 (이벤트 버스)

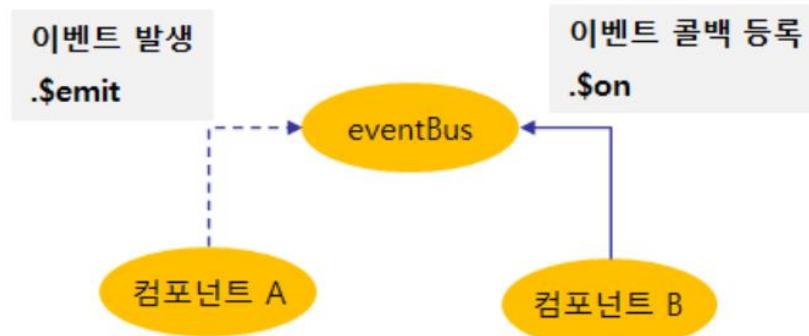
Vue.js

이벤트 버스는 단순한 Vue 객체이다.

```
let eventBus = new Vue();
```

이벤트 버스는 데이터를 보내려는 쪽과 데이터를 받는 쪽 모두에서 사용하게 된다.

자식에서는 이벤트를 발신(emit)하고 부모에서는 v-on 이용하여 이벤트를 수신한다



```
eventBus.$emit("gogo", "파라미터");
```

```
created () {
  eventBus.$on("gogo", this.update);
},
```

4) 상/하위 레벨이 아닌 컴포넌트 통신 (이벤트 버스)

Vue.js

```
<div id="app">  
  <comp-1></comp-1>  
  <comp-2></comp-2>  
</div>  
  
let eventBus = new Vue();
```

```
Vue.component("comp-1", {  
  template: `<input type="text" @change="trans">`,  
  methods: {  
    trans(e) {  
      eventBus.$emit("gogo", e.target.value);  
    }  
  }  
})
```

eventBus에 gogo 이벤트
발행 및 값 전달

```
Vue.component("comp-2", {  
  template: `<span>{{msg}}</span>`,  
  data: function() {  
    return {  
      msg: "message"  
    };  
  },  
  created() {  
    // mounted도 괜찮음  
    eventBus.$on("gogo", this.update);  
  },  
  methods: {  
    update(data) {  
      this.msg = data;  
    }  
  }  
});
```

eventBus에 gogo
이벤트 콜백 등록

8장. axios 라이브러리



axios 개요

기본 API

Config 구성요소

Restful + SpringBoot 연동

1) axios 개요

Vue.js

axios는 Ajax 처리를 위한 비동기 처리 라이브러리이다.

<https://github.com/axios/axios>

axios

npm v0.21.1 cdnjs v0.21.1 build passing Gitpod Ready-to-Code coverage 94% install size 388 kB downloads 75M/month
chat on gitter code helpers 140

Promise based HTTP client for the browser and node.js

New axios docs website: [click here](#)

Table of Contents

- Features
- Browser Support
- Installing
- Example
- Axios API
- Request method aliases
- Concurrency (Deprecated)
- Creating an instance
- Instance methods
- Request Config
- Response Schema
- Config Defaults
 - Global axios defaults
 - Custom instance defaults

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

axios API

Requests can be made by passing the relevant config to `axios`.

axios(config)

```
// Send a POST request
axios({
  method: 'post',
  url: '/user/12345',
  data: {
    firstName: 'Fred',
    lastName: 'Flintstone'
  }
});
```

Performing a `POST` request

```
axios.post('/user', {
  firstName: 'Fred',
  lastName: 'Flintstone'
})
.then(function (response) {
  console.log(response);
})
.catch(function (error) {
  console.log(error);
});
```

axios는 jQuery의 ajax 함수 만큼이나 편리하게 ajax를 처리할 수 있도록 다양한 API들을 제공한다.

axios.request(config)

axios.get(url[, config])

axios.delete(url[, config])

axios.head(url[, config])

axios.options(url[, config])

axios.post(url[, data[, config]])

axios.put(url[, data[, config]]))

axios.patch(url[, data[, config]]))

3) Config 주요 구성과 활용

Vue.js

axios 함수에 전달하는 config 객체는 다음과 같은 주요 속성을 갖는다.

```
let axios = axios({
  url: "./food.json", // 호출할 서버의 경로

  method: "get", // 사용하는 http method(post, get, put, delete)로 default는 get

  params: {
    name: "hong"
  }, // url 즉 쿼리스트링을 구성하는 파라미터 요소

  data: {
    age: 10,
    addr: "seoul"
  }, // request body를 통해서 서버로 전송되는 값(post, put, patch에서 사용)
});
```

요청에 대한 응답 결과는 then과 catch 콜백함수로 처리한다.

```
axios.then(
  success_callback
).catch(
  error_callback
).finally(
  finally_callback
);
```

```
{
  // 서버가 출력한 값은 언제나 data 속성 안에 존재한다.
  data: {},
  // HTTP status code
  status: 200,
  // HTTP status message from the server response
  statusText: 'OK',
  // `headers` the headers that the server responded with All header names are lower cased
  headers: {},
  // `config` is the config that was provided to `axios` for the request
  config: {}
}
```

4) Axios 기본실습

Vue.js

<https://jsonplaceholder.typicode.com/users>

← → ⌂ ⌂ https://jsonplaceholder.typicode.com/users

```
{  
    "id": 1,  
    "name": "Leanne Graham",  
    "username": "Bret",  
    "email": "Sincere@april.biz",  
    "address": {  
        "street": "Kulas Light",  
        "suite": "Apt. 556",  
        "city": "Gwenborough",  
        "zipcode": "92998-3874",  
        "geo": {  
            "lat": "-37.3159",  
            "lng": "81.1496"  
        }  
    },  
    "phone": "1-770-736-8031 x56442",  
    "website": "hildegard.org",  
    "company": {  
        "name": "Romaguera-Crona",  
        "catchPhrase": "Multi-layered client-server neural-net",  
        "bs": "harness real-time e-markets"  
    }  
},  
{  
    "id": 2,  
    "name": "Ervin Howell",  
    "username": "Antonette",  
    "email": "Shanna@melissa.tv",  
    "address": {  
        "street": "Victor Plains",  
        "suite": "Suite 879",  
        "city": "Wisokyburgh",  
        "zipcode": "90566-7771",  
        "geo": {  
            "lat": "-43.9509",  
            "lng": "-34.4618"  
        }  
    },  
    "phone": "1-464-946-5464",  
    "website": "anastasia.net",  
    "company": {  
        "name": "Deckow-Crist",  
        "catchPhrase": "Proactive didactic contingency",  
        "bs": "synergies"
```

```
<script>  
var app = new Vue({  
    el: '#app',  
    data:{  
        list:[]  
    },  
    methods:{  
        posts(){  
            var xxx = this.list;  
            axios.get('https://jsonplaceholder.typicode.com/users')  
                .then(function (response) {  
                    console.log(response.data)  
                    response.data.map(function(ele,idx){  
                        xxx.push(ele);  
                    });  
                })  
                .catch(function (error) {  
                    // handle error  
                    console.log(error);  
                })  
        }  
    }  
});
```

axios

posts

- 1,Leanne Graham,92998-3874
- 2,Ervin Howell,90566-7771
- 3,Clementine Bauch,59590-4157
- 4,Patricia Lebsack,53919-4257
- 5,Chelsey Dietrich,33263
- 6,Mrs. Dennis Schulist,23505-1337
- 7,Kurtis Weissnat,58804-1099
- 8,Nicholas Runolfsdottir V,45169
- 9,Glenna Reichert,76495-3109
- 10,Clementina DuBuque,31428-2261

In 16 Col 65 Spaces: 4

5) SpringBoot + Vue.js 연동 실습

Vue.js

customer-controller : Customer Controller

Show/Hide | List Operations | Expand Operations

GET	/del/v0.8/{cstmld}	고객삭제
GET	/exist/rest/v0.8/{cstmld}	고객존재여부조회
POST	/rest/v0.8	고객등록
GET	/rest/v0.8/all	모든 고객기본조회
GET	/rest/v0.8/{cstmld}	고객기본조회
PUT	/update/v0.8	고객수정

사용자 관리

전체조회						ID검색
사원ID	사원명	나이	성별	전화번호	주소	
1111	홍길동	20	1	01012341234	마곡	
2222	홍길순	20	2	01043214321	발산	
3333	이순신	65	1	01099996666	여수	

사원ID	3333
사원명	이순신
나이	65
성별	1
전화번호	01099996666
주소	여수

[추가](#) [삭제](#) [수정](#)



Thank you