



진보적인 프레임워크

# Vue JS *Progressive Framework*

# What Why?

왜? Vue를 선택했나?



 한글로 작성된 개발문서 제공

웹 애플리케이션을 제작하는데 요구되는 기능을 제공하는  
Vue.js 프론트엔드 프레임워크는 간단한 API와 직관적인 사용성,  
탬플릿 활용이 무척 쉬워 입문자도 도입하기 쉽습니다.

### 접근성

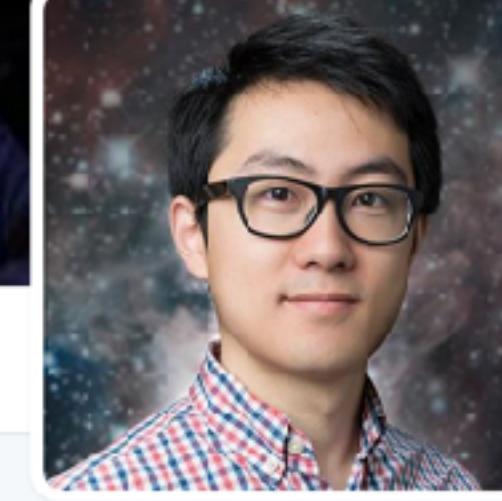
HTML, CSS 및 JavaScript를 아시나요?  
가이드를 읽고 즉시 시작하십시오!

### 유연성

규모가 작은 앱을 처리 할 수 있는 점진적  
으로 채택 가능한 단순하고 최소의 핵심  
스택을 구성할 수 있습니다.

### 고성능

17kb min+gzip Runtime  
엄청나게 빠른 가상 DOM을 제공합니다.  
최소한의 최적화 노력만 필요합니다.

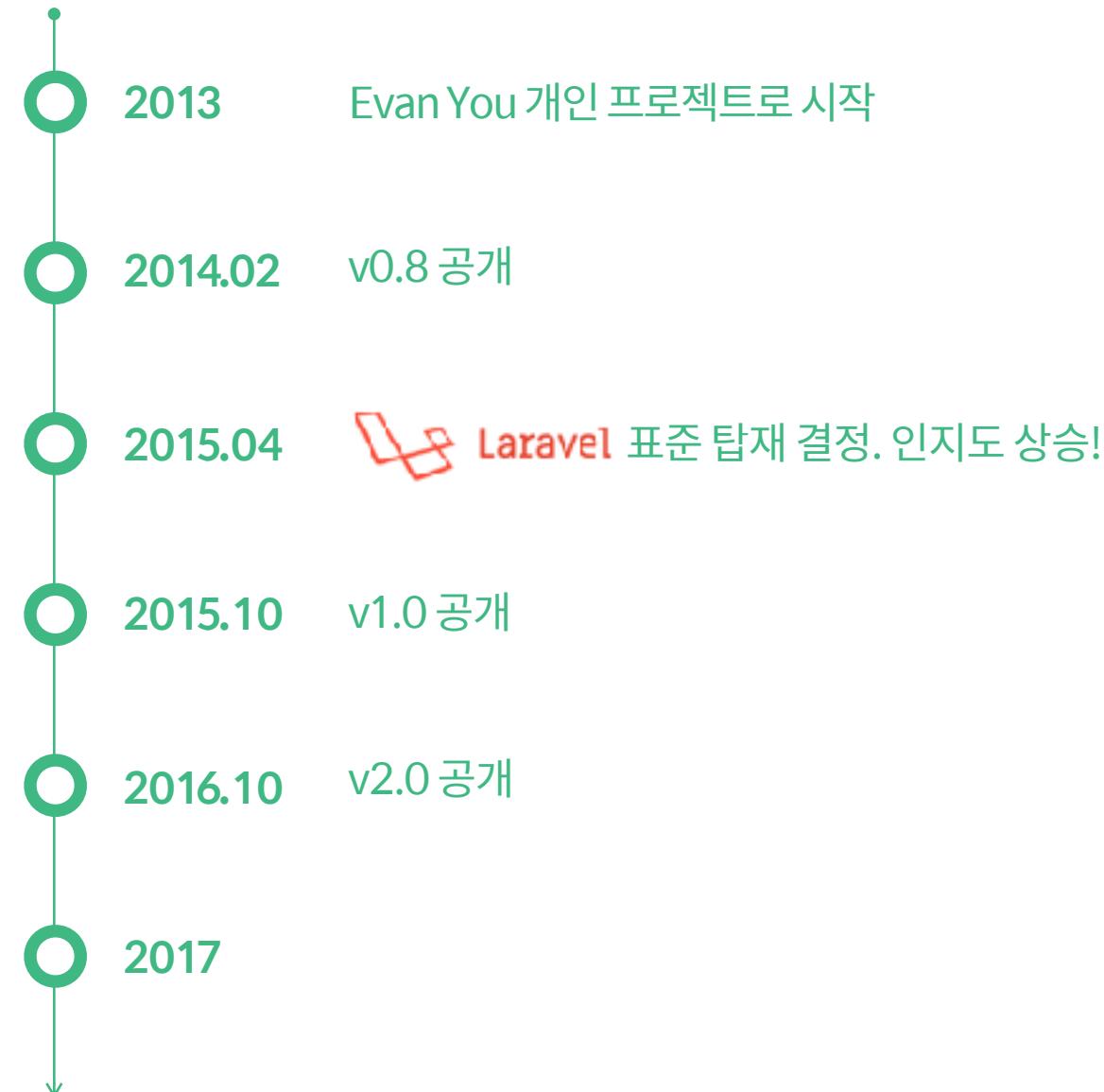


**Evan You**  
@youyuxi

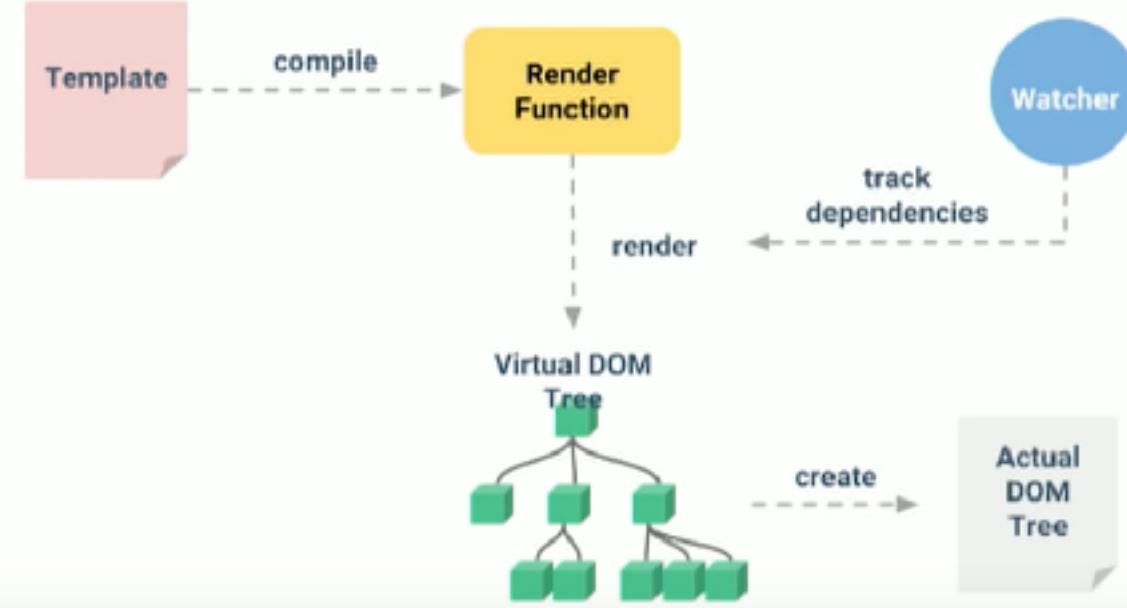
Design, code & things in between. Living the dream working on [@vuejs](#). Previously [@meteorjs](#) & [@google](#), [@parsonsamt](#) alumnus.

New York  
[evanyou.me](http://evanyou.me)

<https://twitter.com/youyuxi>



# Virtual DOM



Template → compile → Render Function → render → Virtual DOM Tree → track dependencies → Watcher → create → Actual DOM Tree

▶ ▶| ⏴ 35:05 / 38:57

▶ HD □ ☰

EVAN YOU

MODERN FRONTEND UI WITH VUE.JS

LARACon EU AMSTERDAM 2016

Evan You - Modern Frontend with Vue.js - Laracon EU 2016

## Vue.js

- Started in late 2013
- First release Feb. 2014

## Apr. 2015: this happened



Taylor Otwell

@taylorotwell

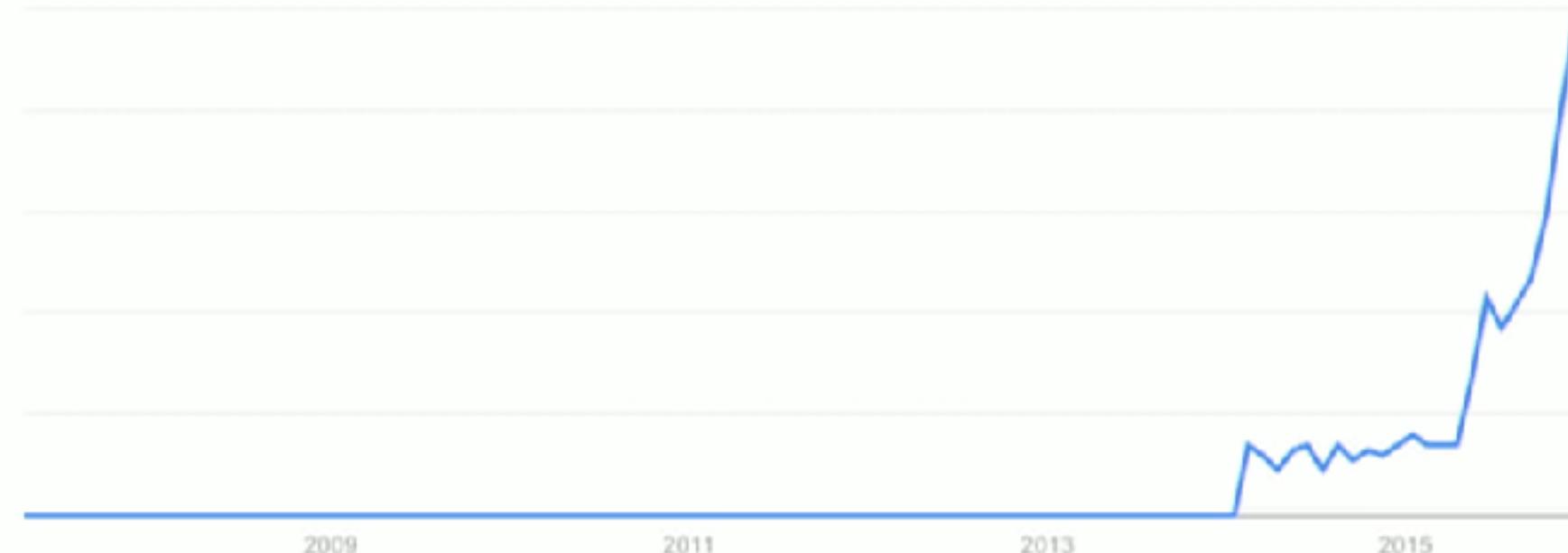
 Follow

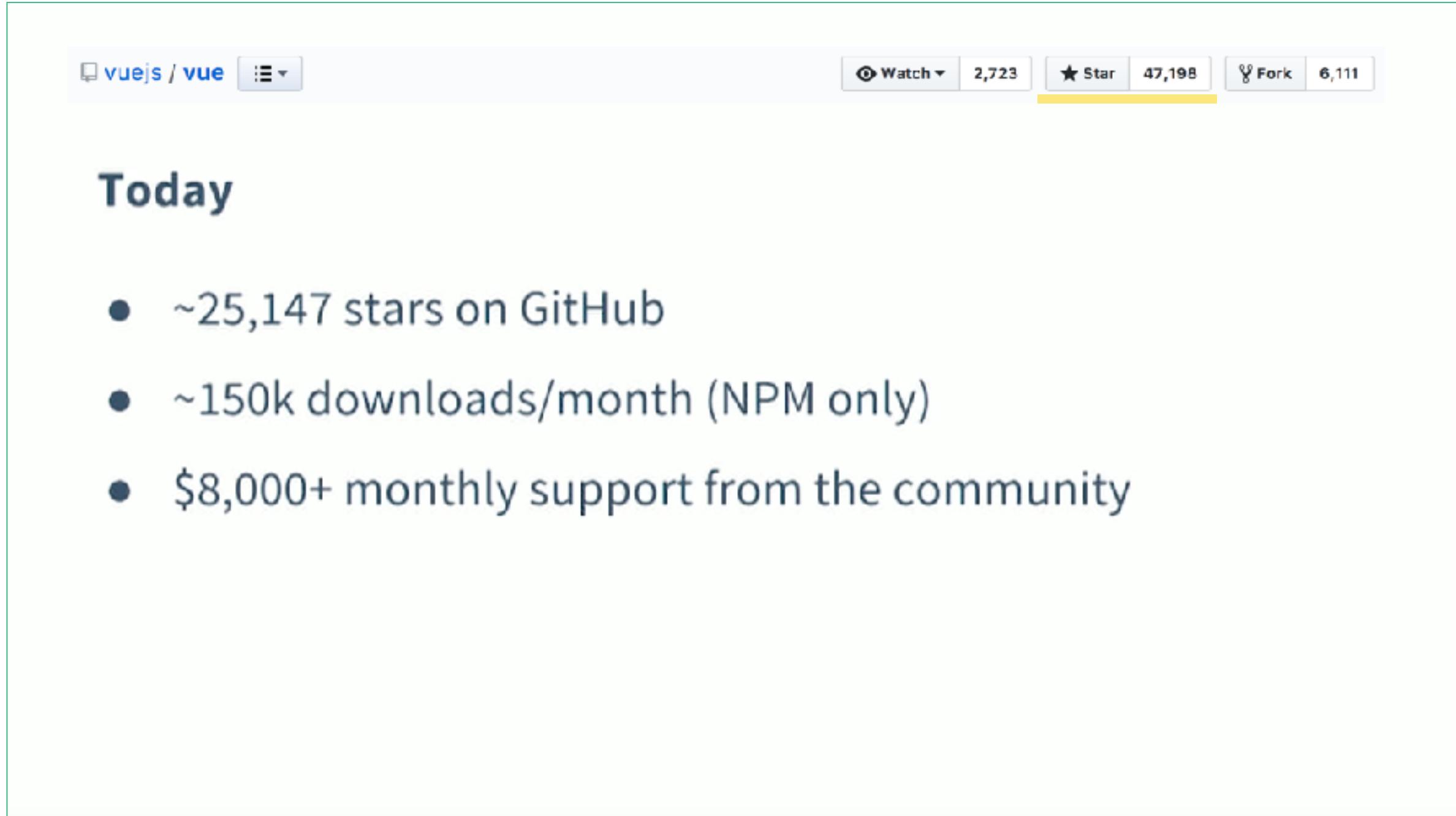
Current React learning status: overwhelmed. Learning [@vuejs](#) because it looks easy and has pretty website. 

6:31 PM - 20 Apr 2015 · Benton, AR, United States

  11  51

## Apr. 2015: this happened





The screenshot shows the GitHub repository page for 'vuejs / vue'. At the top, there's a search bar with 'vuejs / vue' and a dropdown menu. To the right are buttons for 'Watch' (2,723), 'Star' (47,198), 'Fork' (6,111), and a yellow 'Edit' button. Below this, the word 'Today' is displayed in large blue letters. A bulleted list follows:

- ~25,147 stars on GitHub
- ~150k downloads/month (NPM only)
- \$8,000+ monthly support from the community

**Big Thanks to the Laravel  
Community for making it  
possible.**

# The Progressive Framework

# pro·gres·sive

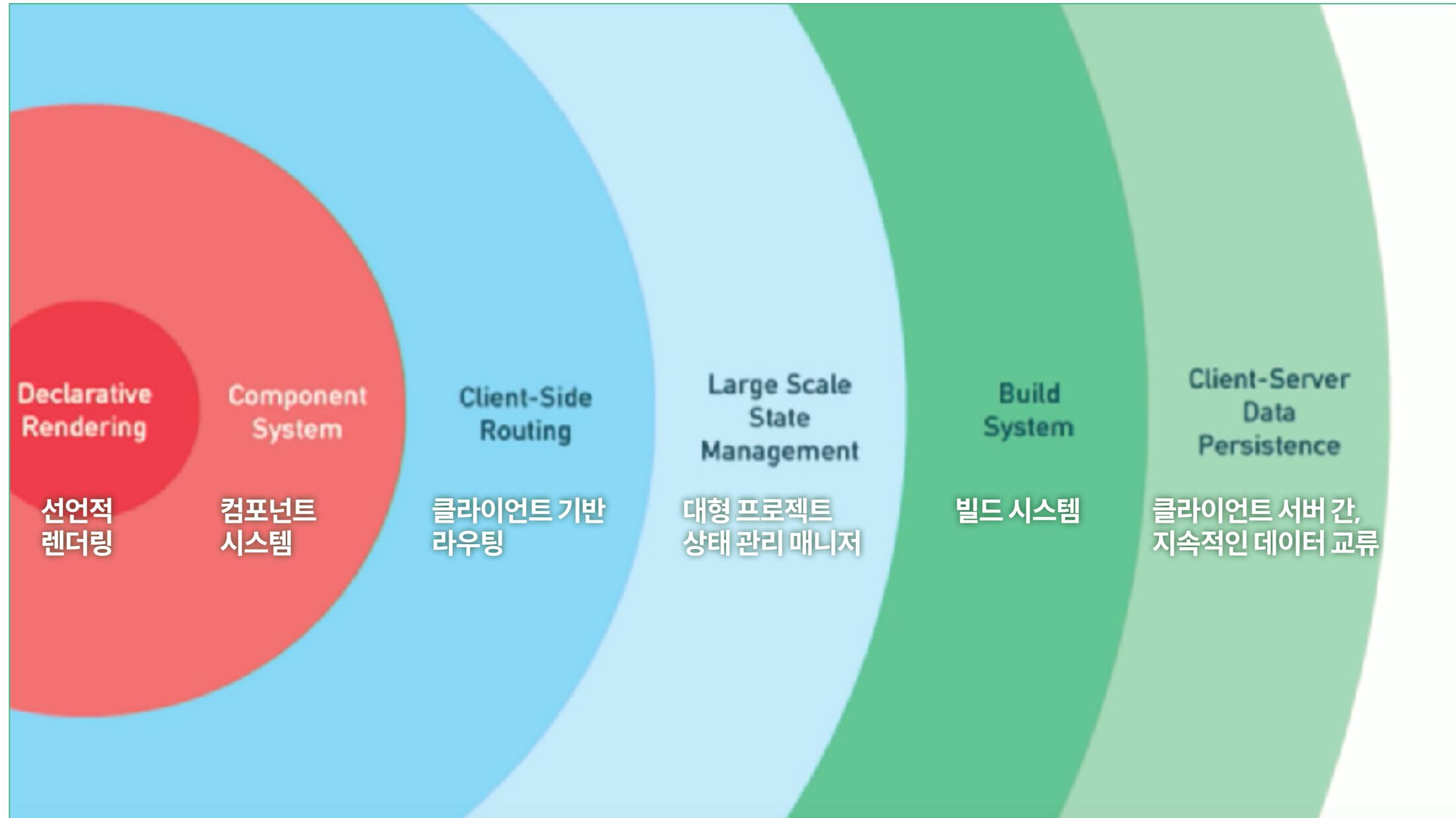
/prə'gresiv/

*adjective* 단계적(점진적)으로 진행된다.

1. happening or developing gradually or in stages; proceeding step by step.

"a progressive decline in popularity"

*synonyms:* continuing, [continuous](#), increasing, growing, developing, [ongoing](#), accelerating, escalating; [More](#)

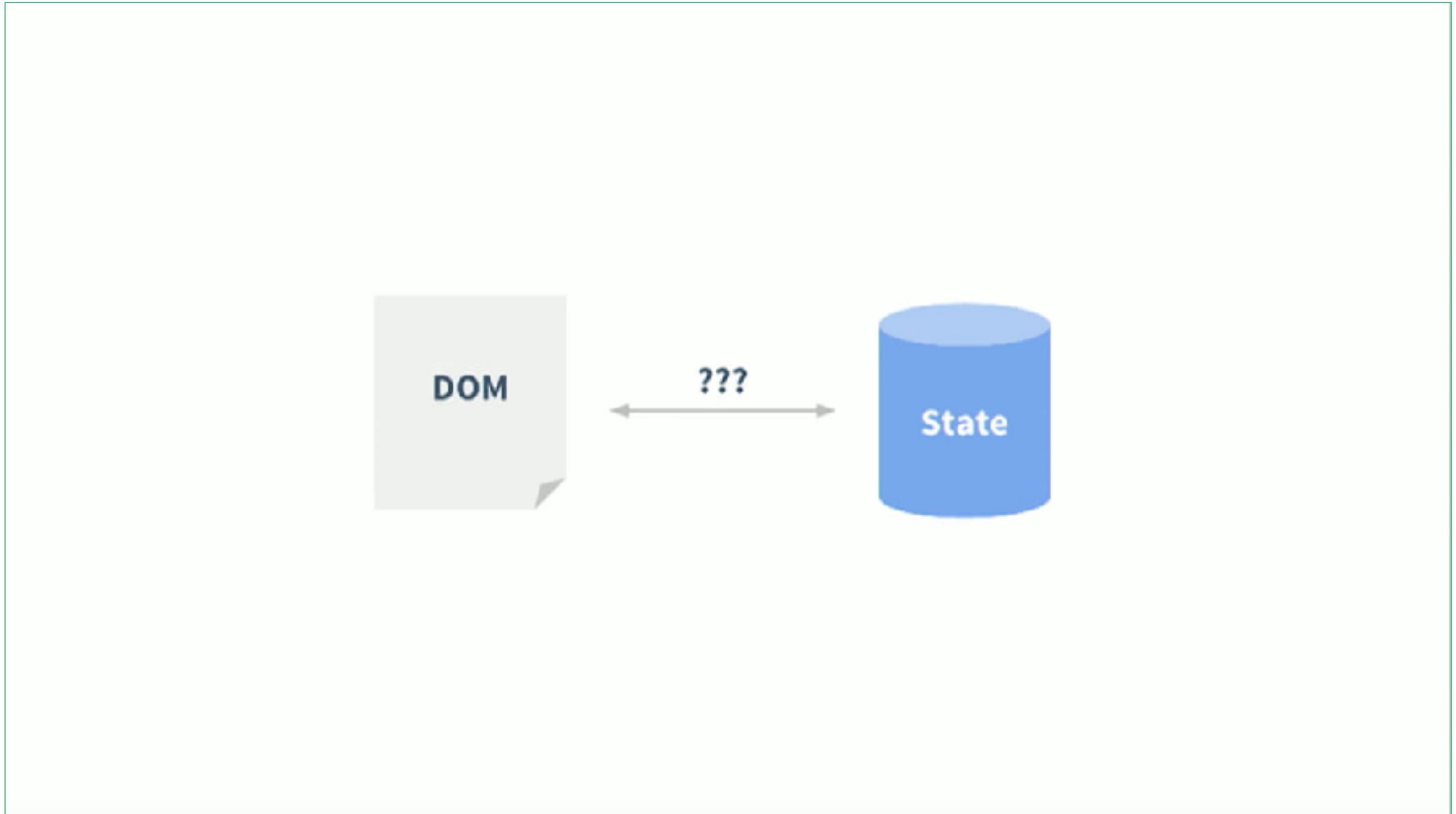


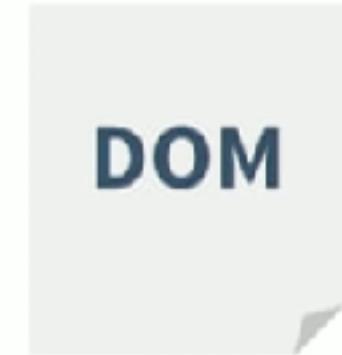


프로그래시브 JavaScript 프레임워크

FAST CAMPUS | Front-End Develop SCHOOL

# Declarative Rendering



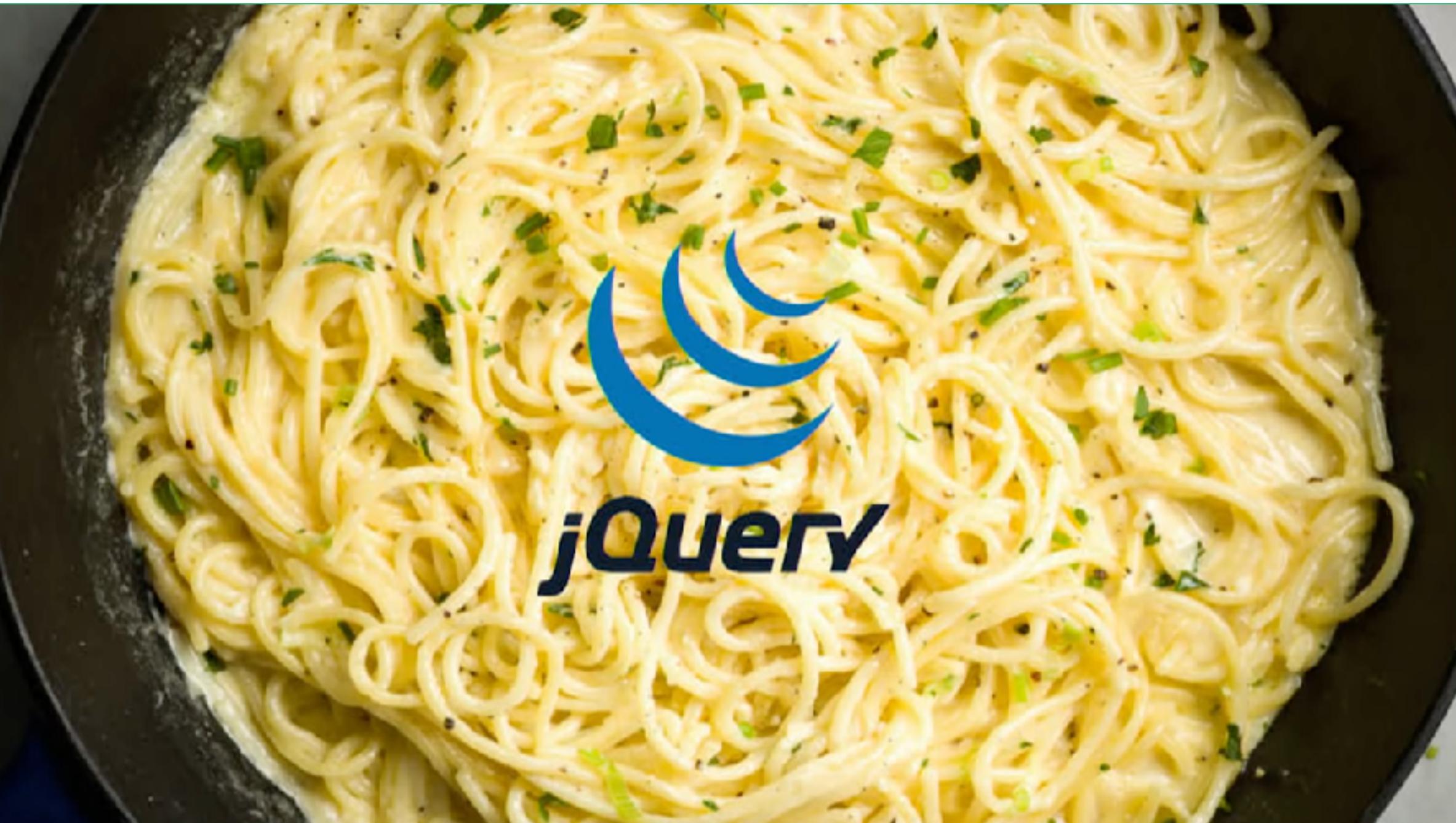


## Problems with the DOM

- Re-rendering entire chunks of DOM is expensive and disruptive
- Imperatively keeping the DOM in sync with the state is tedious and error-prone

DOM을 통해 화면을 다시 그리는 것은 비싸고, 지장을 준다.

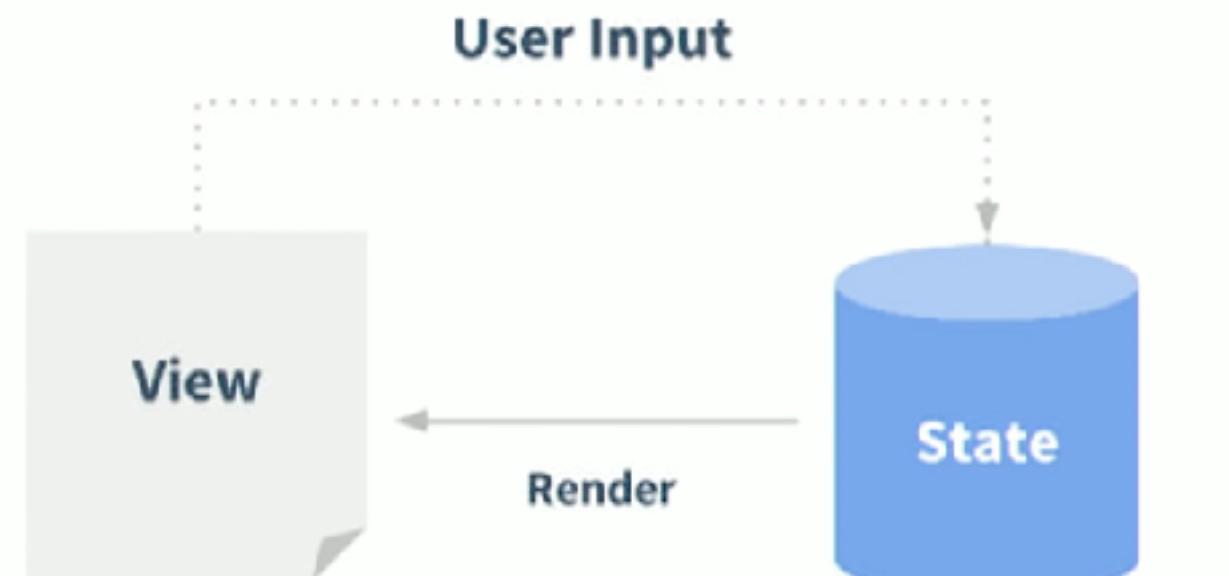
상태가 변경됨에 따라 DOM을 처리하는 과정은 오류가 잦고, 지루하다.



## Declarative & Reactive Rendering

프로그래시브 JavaScript 프레임워크

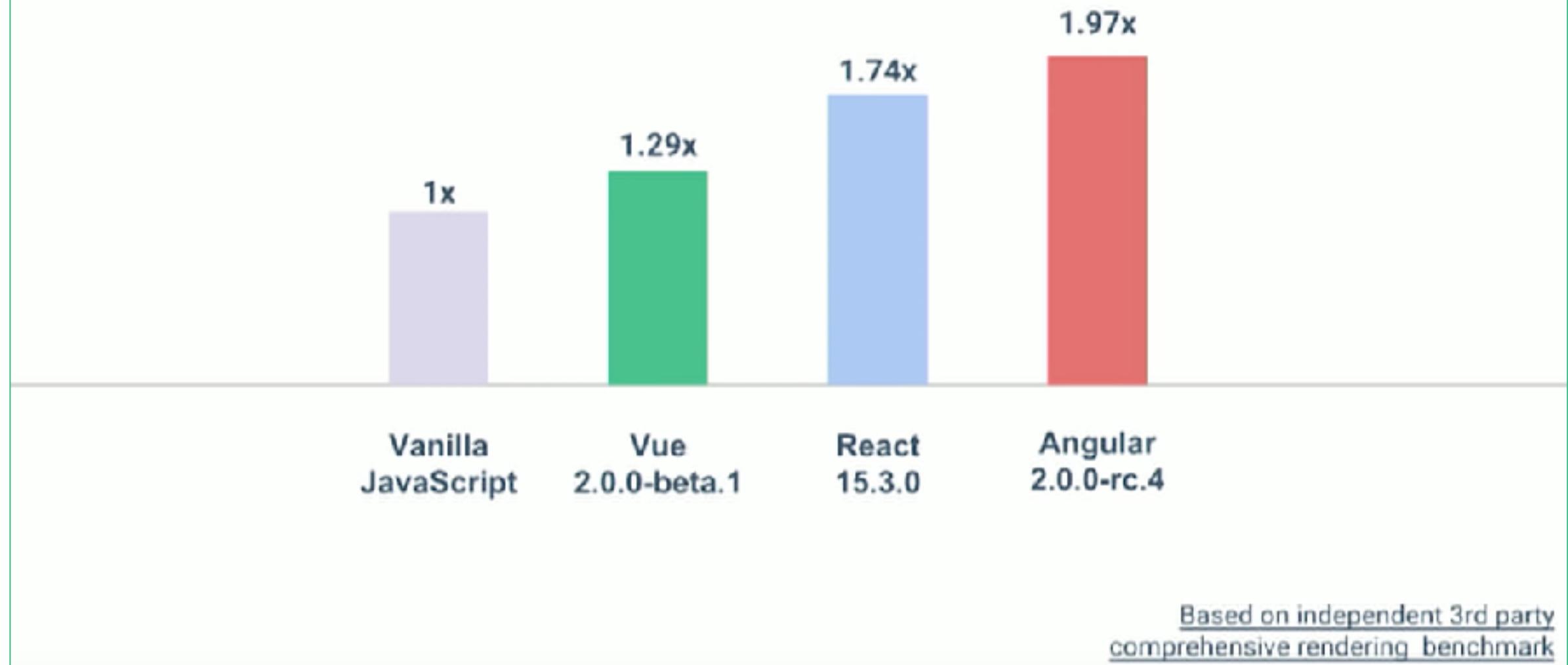
FAST CAMPUS | Front-End Develop SCHOOL



View is just a  
declarative mapping  
from the state

State should be the  
single source of truth

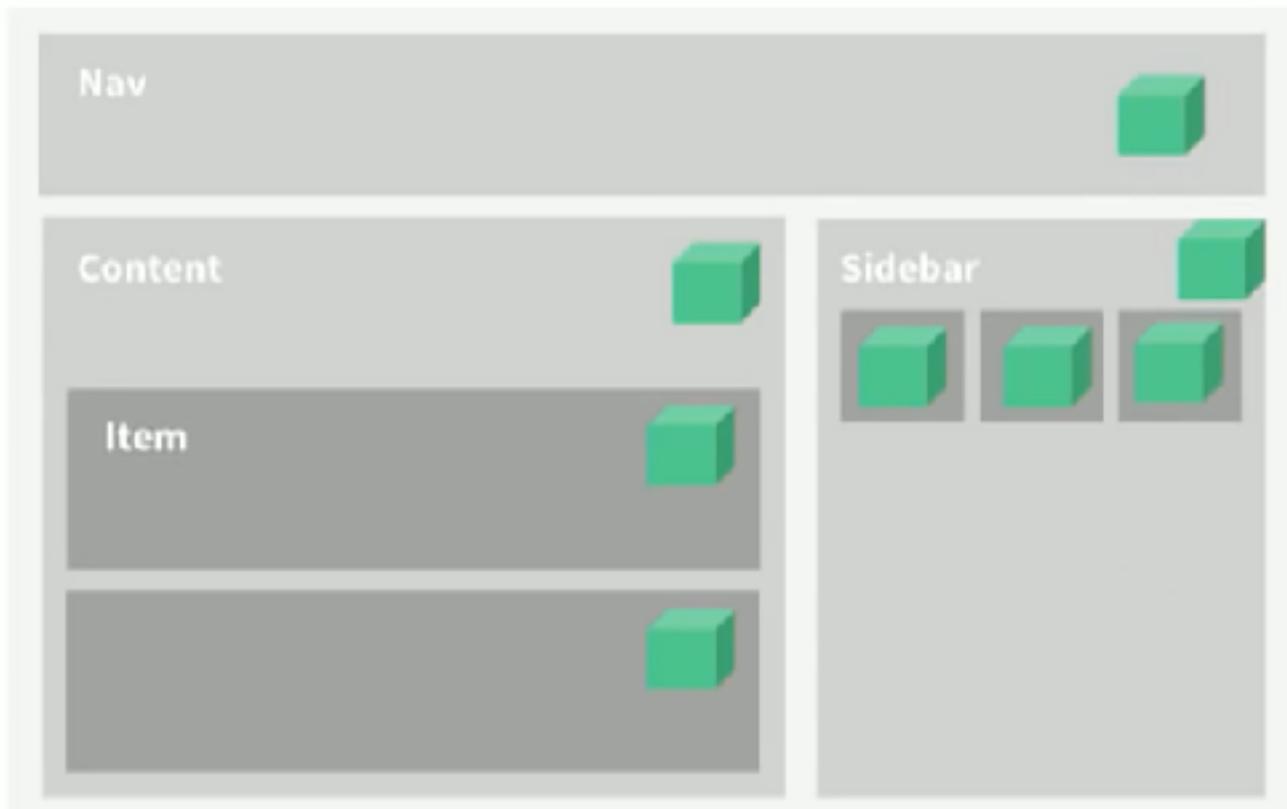
...while being fast



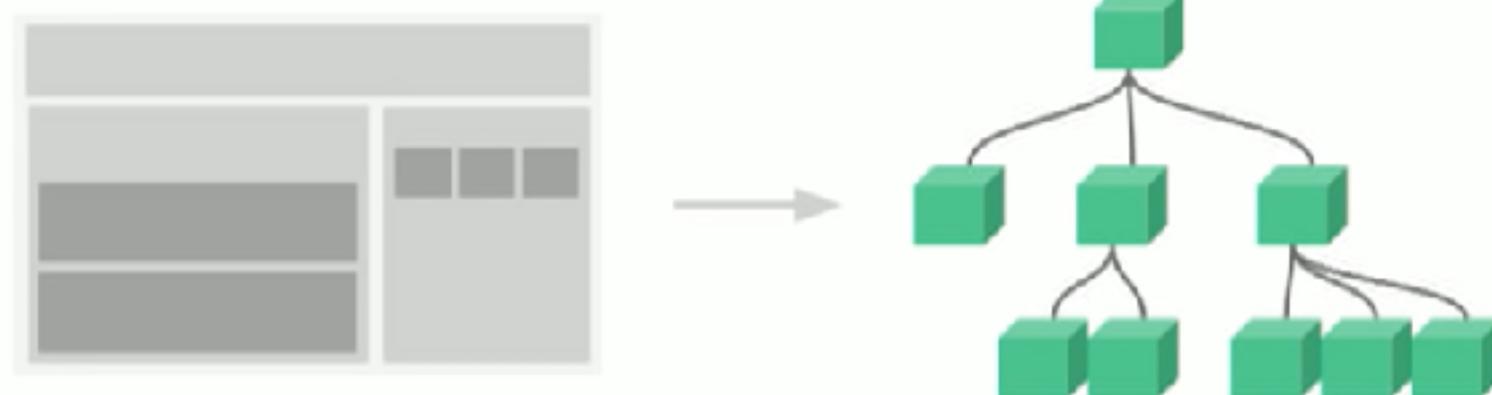
...and approachable.

```
<script src="vue.js"></script>
<script>
  new Vue({
    // ...
  })
</script>
```

Every component is responsible  
for managing a piece of DOM



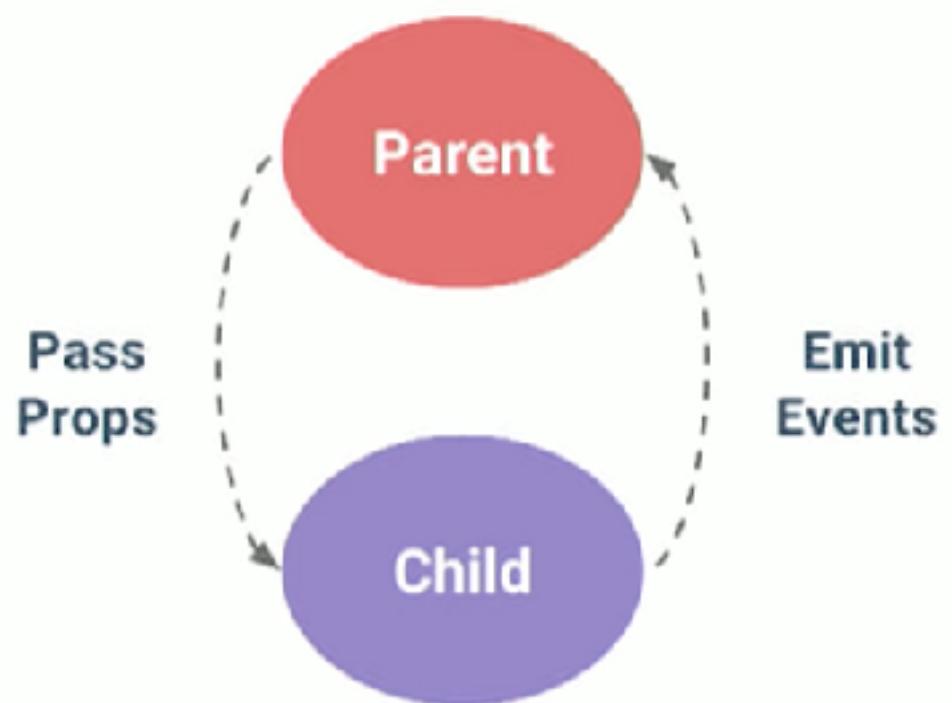
The entire UI can be abstracted into a tree of components



## Nesting Components with Custom Elements

```
<side-bar></side-bar>
<tabs>
  <tab>...</tab>
  <tab>...</tab>
</tabs>
```

# Component Communication: Props in, Events out



# Client-Side Routing

<https://github.com/vuejs/vue-router>

# vue-router

## Component-based client-side routing

/app/



App

Home

/app/post/1



App

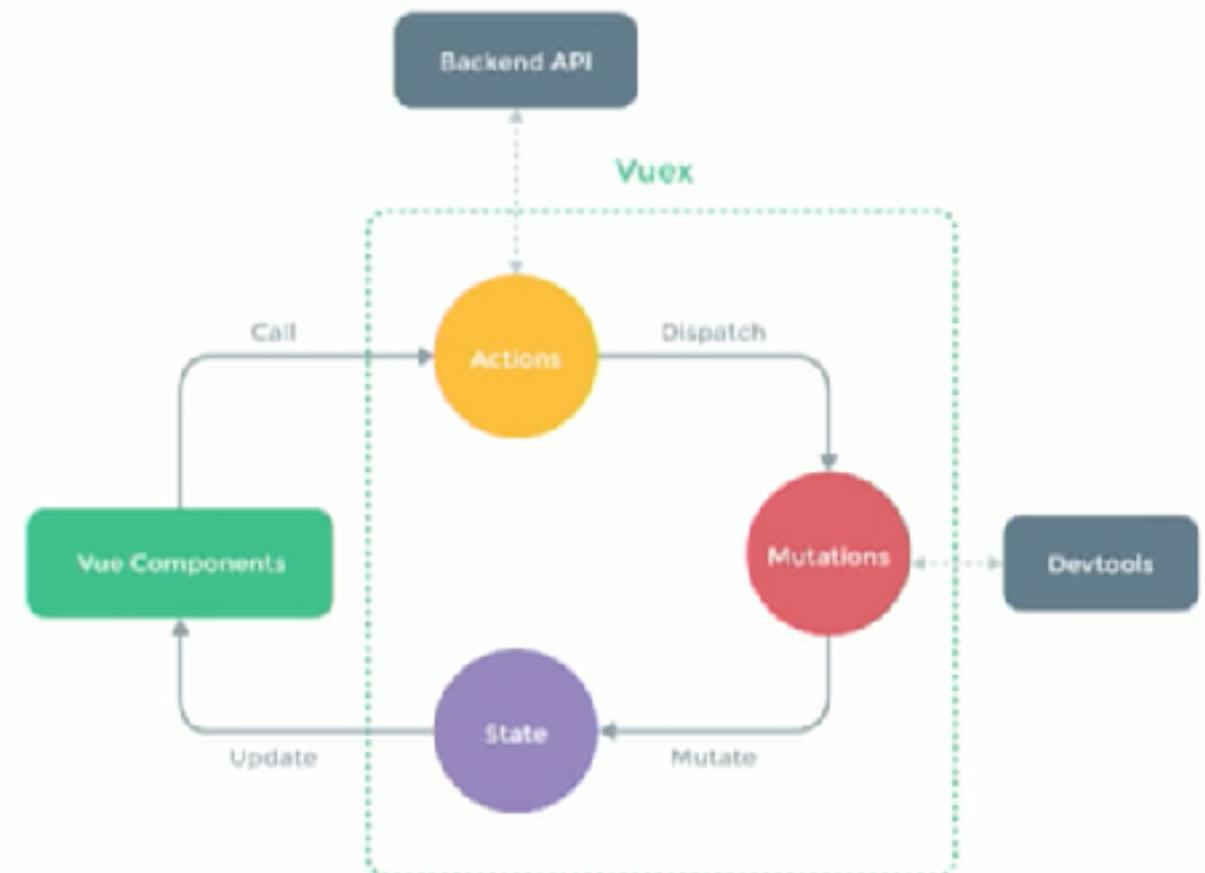
Post with { id: 1 }

## Common Problems In Large Frontend Apps

1. Sharing state between multiple components
2. Unpredictable state mutations

# Vuex

## Centralized & Predictable State Management



# Build System & Development Experience

## Single File Vue Components



```
<template lang="jade">
  div.my-component
    h1 {{ msg }}
    other-component
  </template>

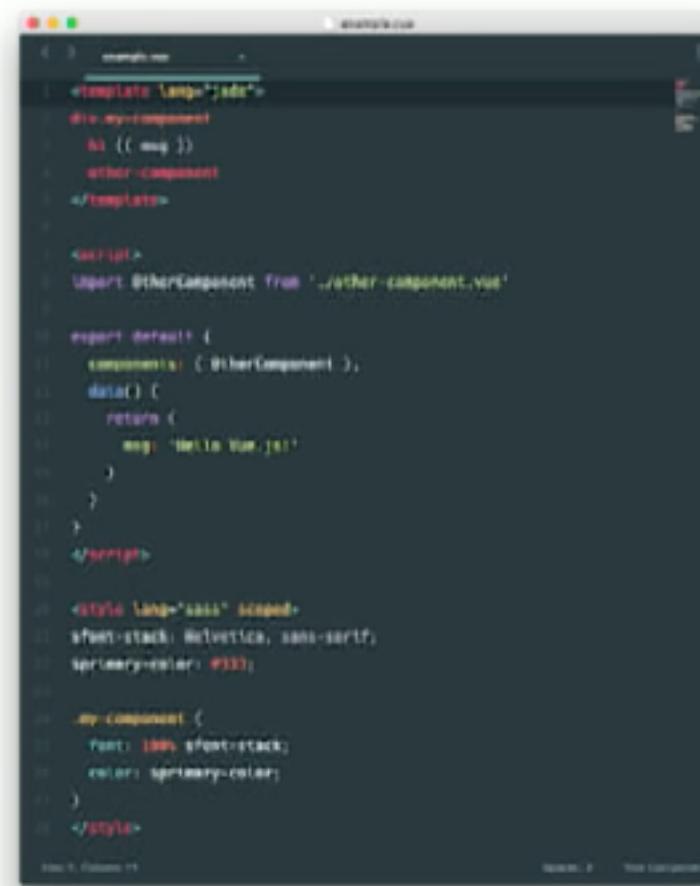
<script>
  import OtherComponent from './other-component.vue'

  export default {
    components: { OtherComponent },
    data() {
      return {
        msg: 'Hello Vue.js!'
      }
    }
  }
</script>

<style lang="sass" scoped>
  $font-stack: helvetica, sans-serif;
  $primary-color: #333;

  .my-component {
    font: 100% $font-stack;
    color: $primary-color;
  }
</style>
```

## Single File Vue Components



```
<template lang="jade">
  div.my-component
    h1 (( msg ))
    other-component
</template>

<script>
  import OtherComponent from './other-component.vue'

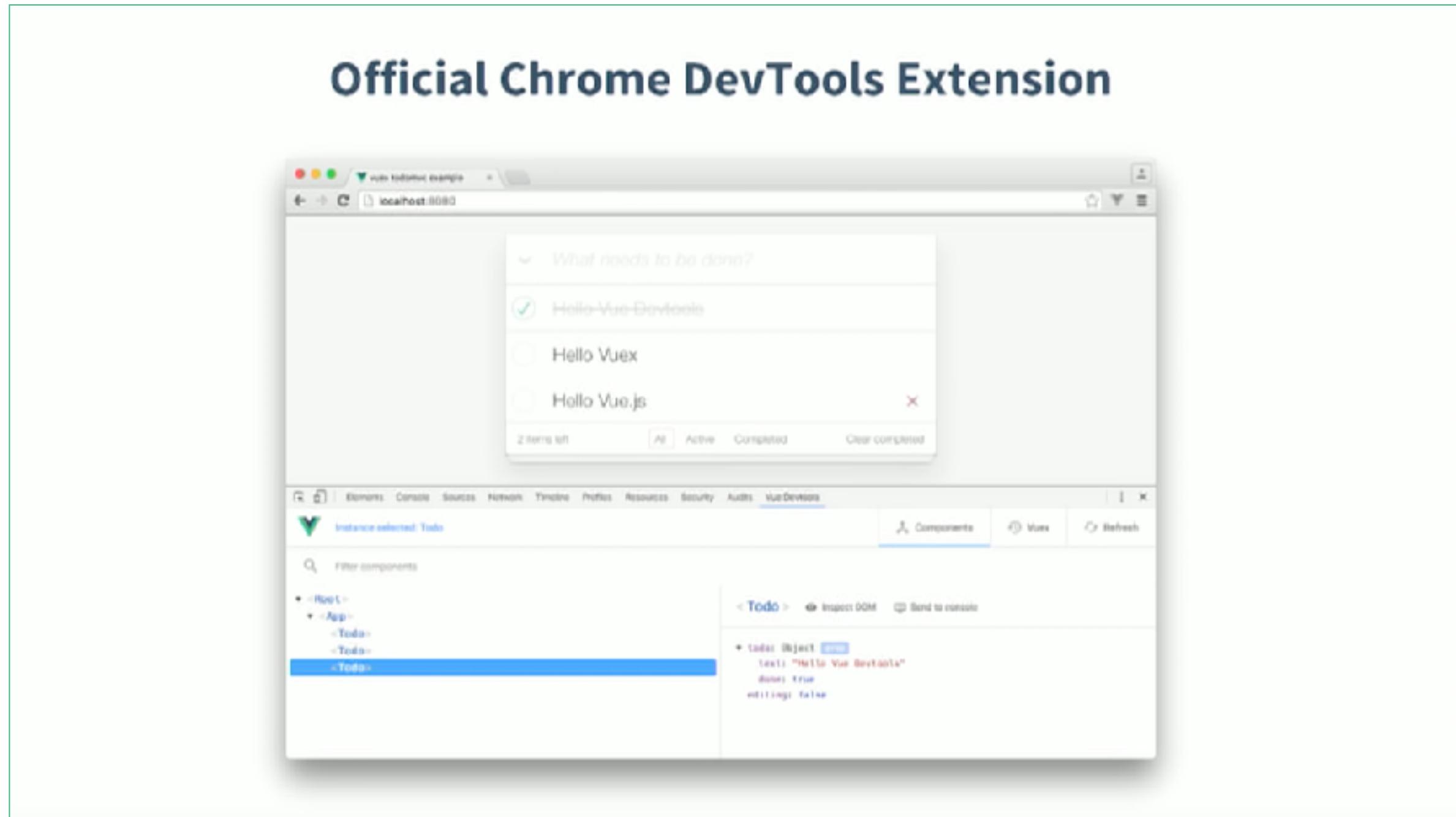
  export default {
    components: { OtherComponent },
    data() {
      return {
        msg: 'Hello Vue.js!'
      }
    }
  }
</script>

<style lang="sass" scoped>
  font-stack: Helvetica, sans-serif;
  color: #333;
</style>

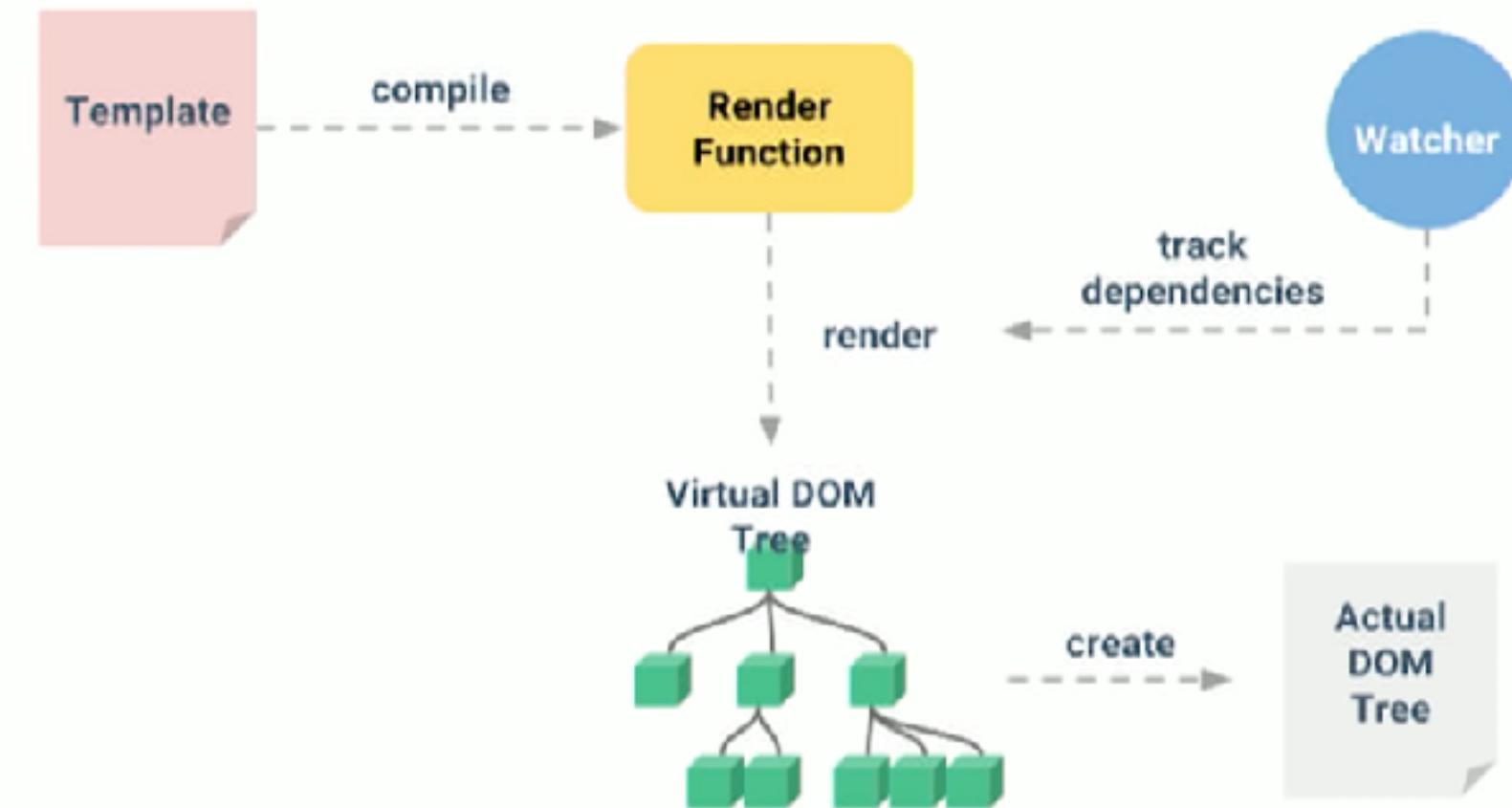
.my-component {
  font: 10px avenir;
  color: #333;
}
```

- Imported as a ES2015 module (thus easily testable)
- Collocation of Template, Logic & Style
- Just use what you already know: HTML, CSS & JavaScript
- Embedded pre-processor support: seamlessly use Babel, SASS or even Pug in the same file
- Component-scoped CSS with a single attribute

```
npm install -g vue-cli
vue-cli init webpack-simple-2.0 my-app
cd my-app
npm install
npm run dev
```

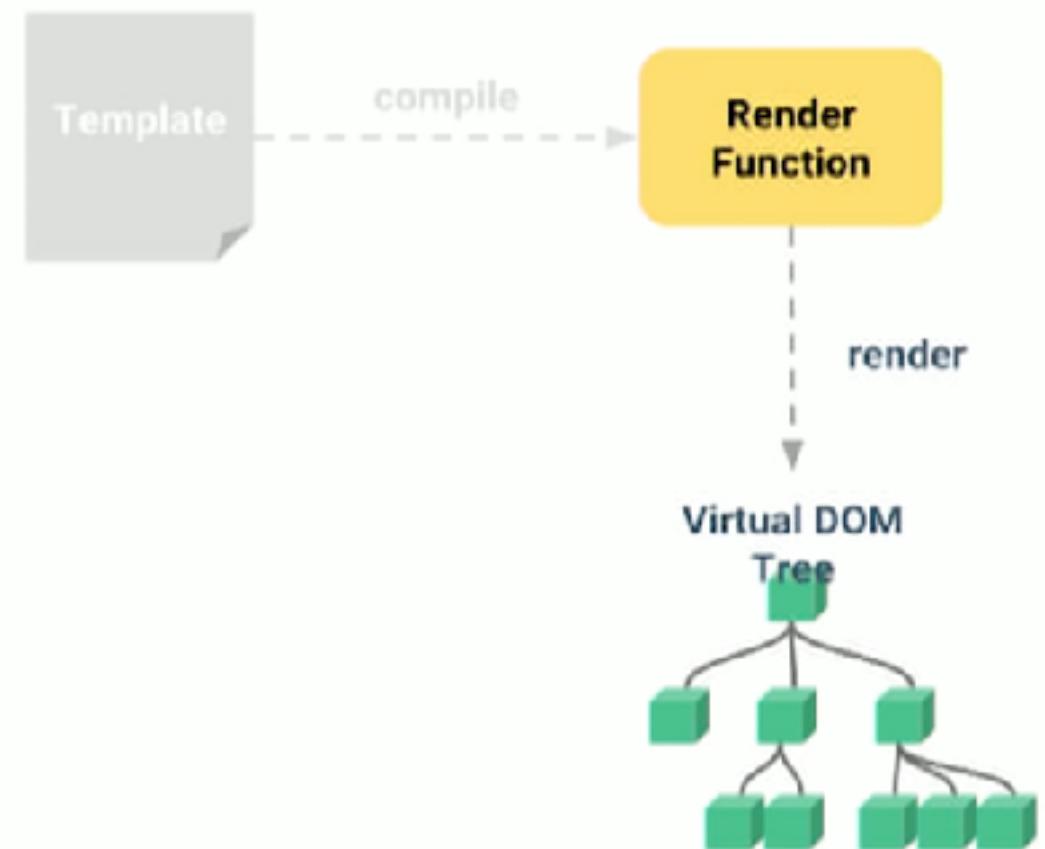


# Virtual DOM

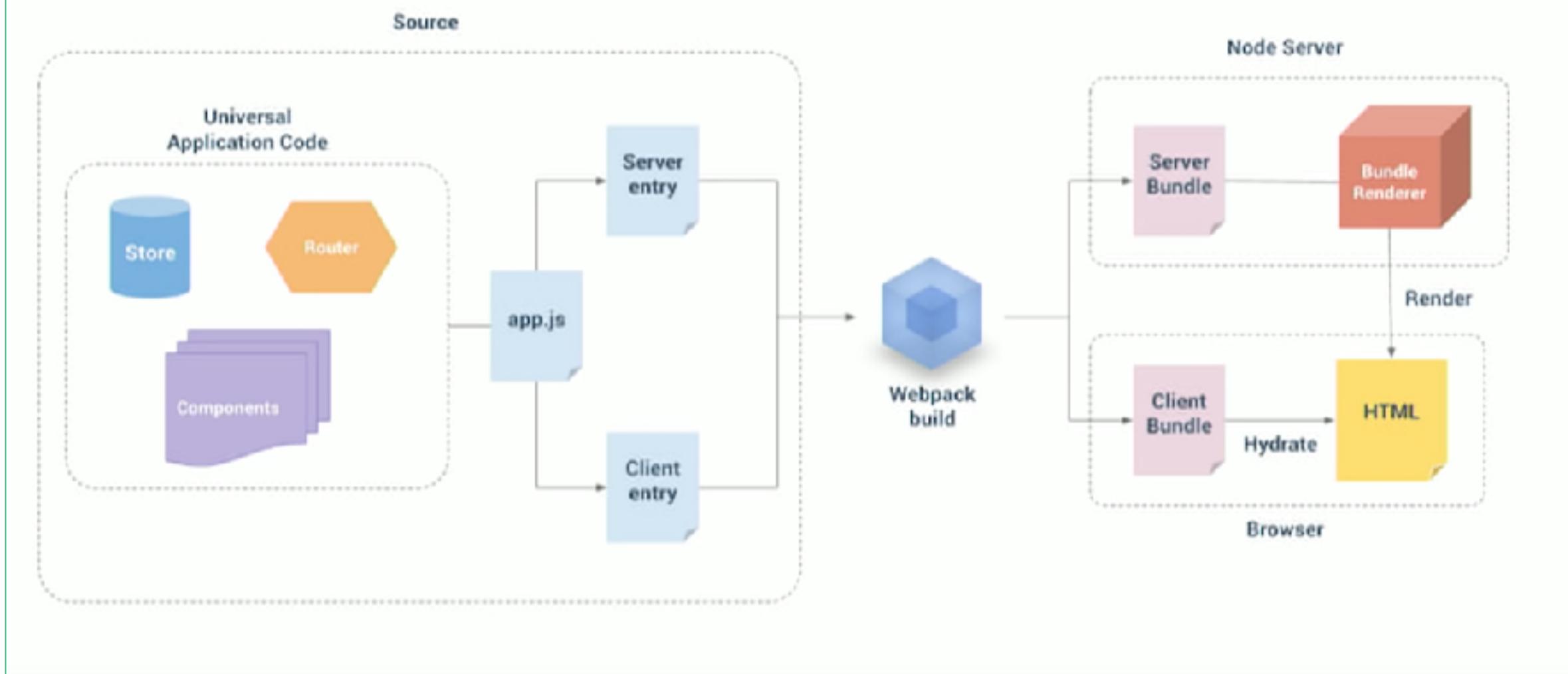


## Render Functions:

### Power beyond templates (when you need it)



# Server Side Rendering





프로그래시브 JavaScript 프레임워크

FAST CAMPUS | Front-End Develop SCHOOL



# Thanks!



vuejs.org



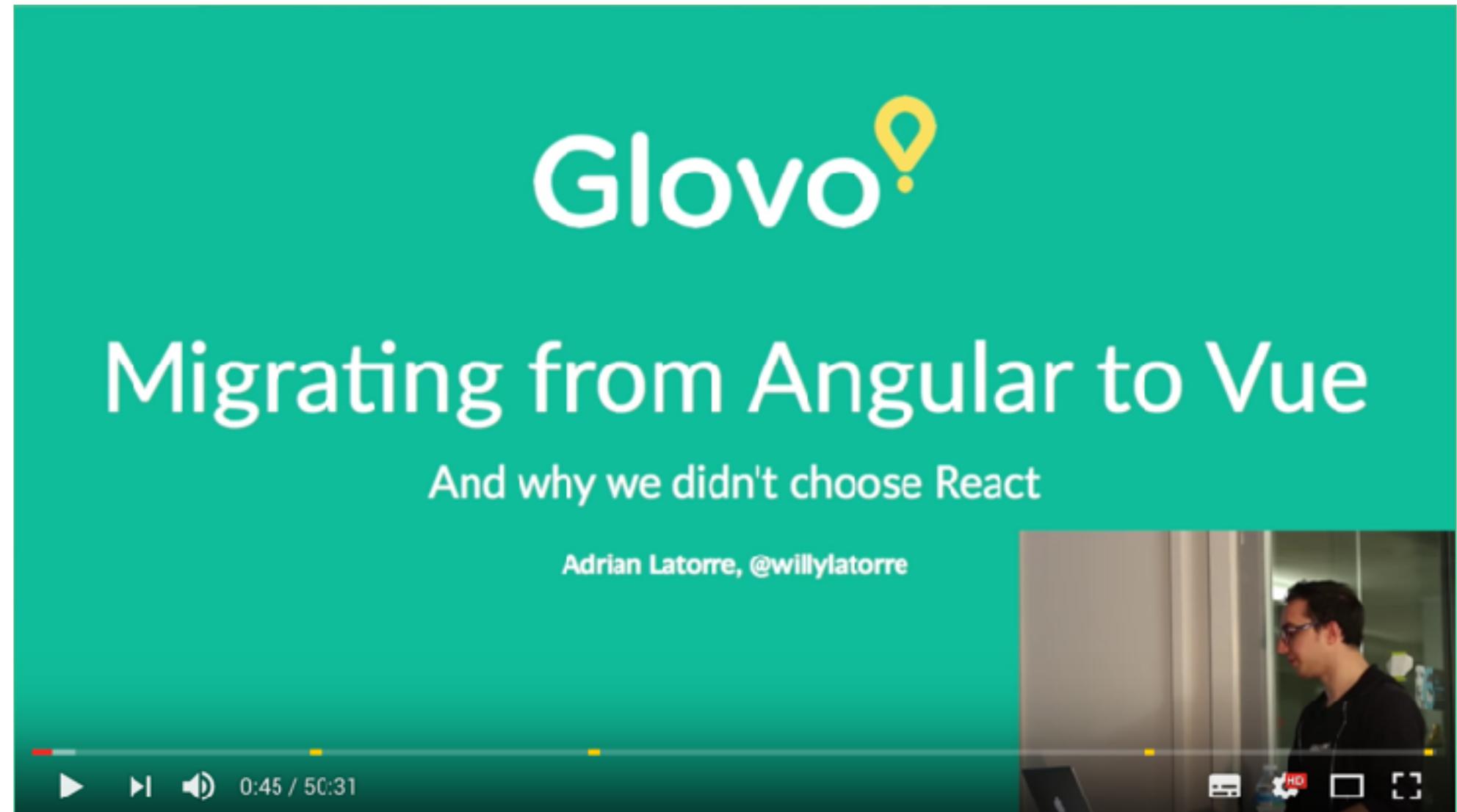
@vuejs



vuejs/vue



## Why We Chose Vue.js - Inside GitLab



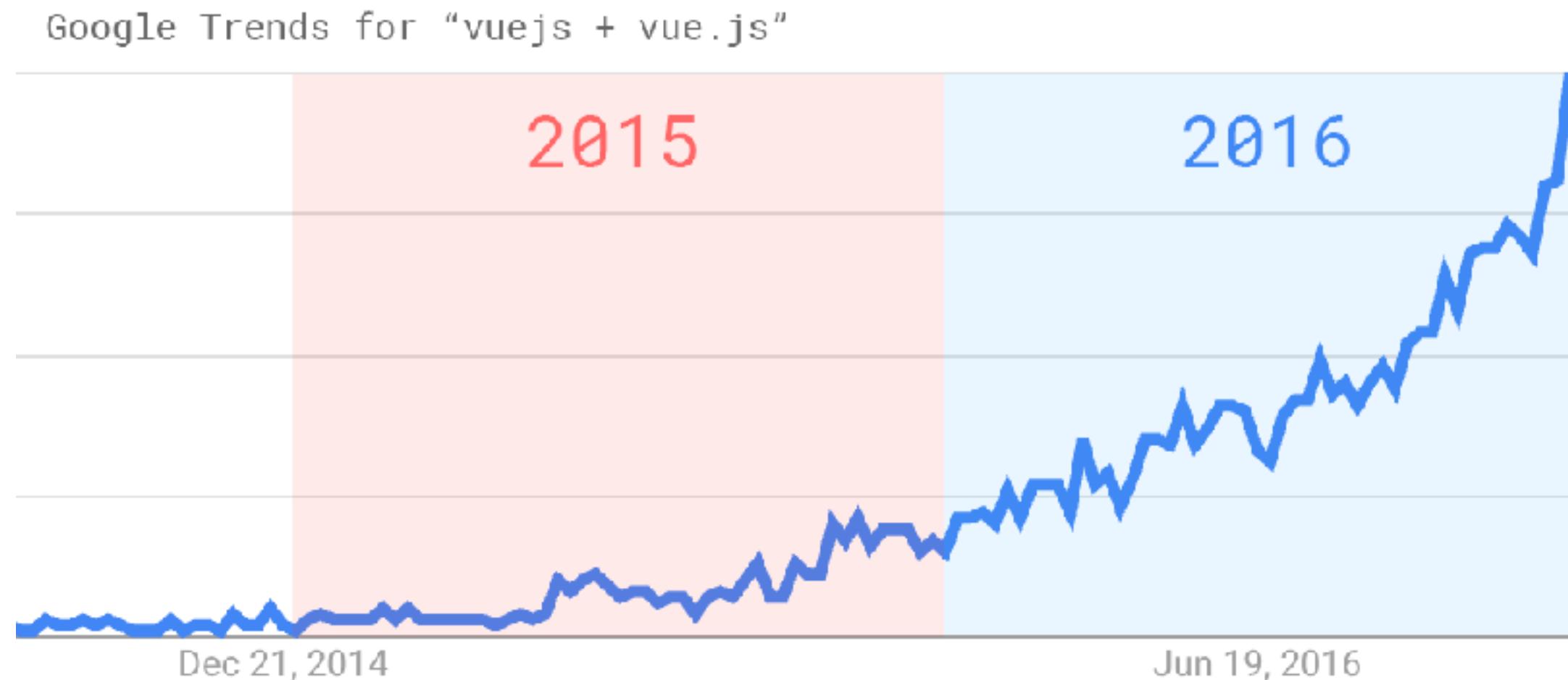
why we didn't choose React - Adrian Latorre

# Repositories Ranking

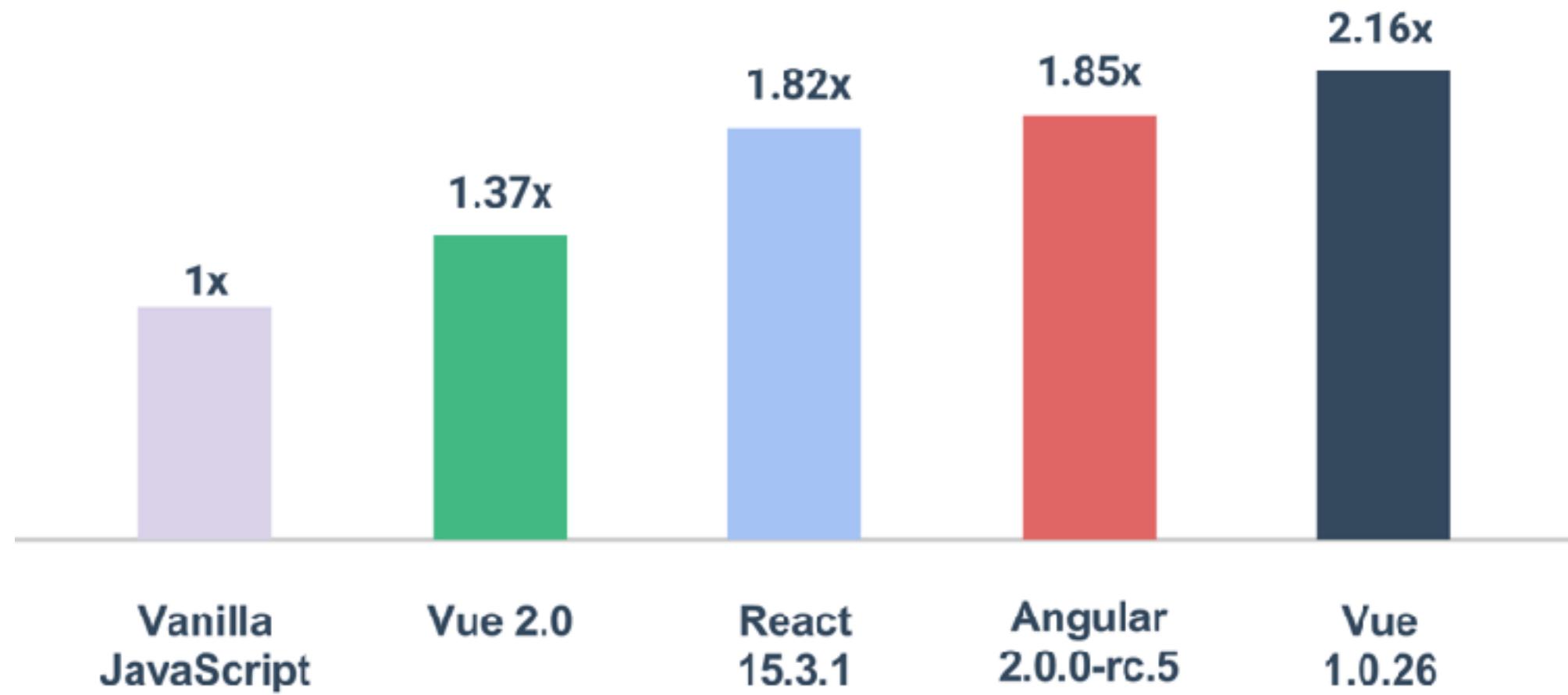
1 2 3 ... 3

	1. twbs/bootstrap	★ 77733
	2. vhf/free-programming-books	★ 35984
	3. angular/angular.js	★ 35311
	4. joyent/node	★ 34715
	5. mbostock/d3	★ 34492
	6. jquery/jquery	★ 33394
 Star	47,198	
	7. vuejs/vue	★ 31473
	7. FontAwesome/Font-Awesome	★ 30484
	8. h5bp/html5-boilerplate	★ 28736
	9. jkbrzt/httpie	★ 27316
	9. sindresorhus/awesome	★ 25849
	10. rails/rails	★ 24954
	11. kennethreitz/requests	★ 23941
	11. bartaz/impress.js	★ 23174
	42. Modernizr/Modernizr	★ 14412
	44. h5bp/Front-end-Developer-In...	★ 14073
	45. TryGhost/Ghost	★ 14059
	46. dypsilion/frontend-dev-bookm...	★ 13953
	47. driftyco/ionic	★ 13674
	48. jashkenas/underscore	★ 13666
	49. astaxie/build-web-applicati...	★ 13552
	49. discourse/discourse	★ 13495
	50. nznick/Chart.js	★ 13250
	51. select2/select2	★ 13155
	52. ariya/phantomjs	★ 13027
	53. django/django	★ 12926
	54. emberjs/ember.js	★ 12719
	55. vinta/awesome-python	★ 12673

## 엄청난 속도로 인지도를 높이고 있는 프론트엔드 프레임워크



## Vue JS 2.0 최대 강점은 고속 렌더링!



2016.10 벤치마크 결과





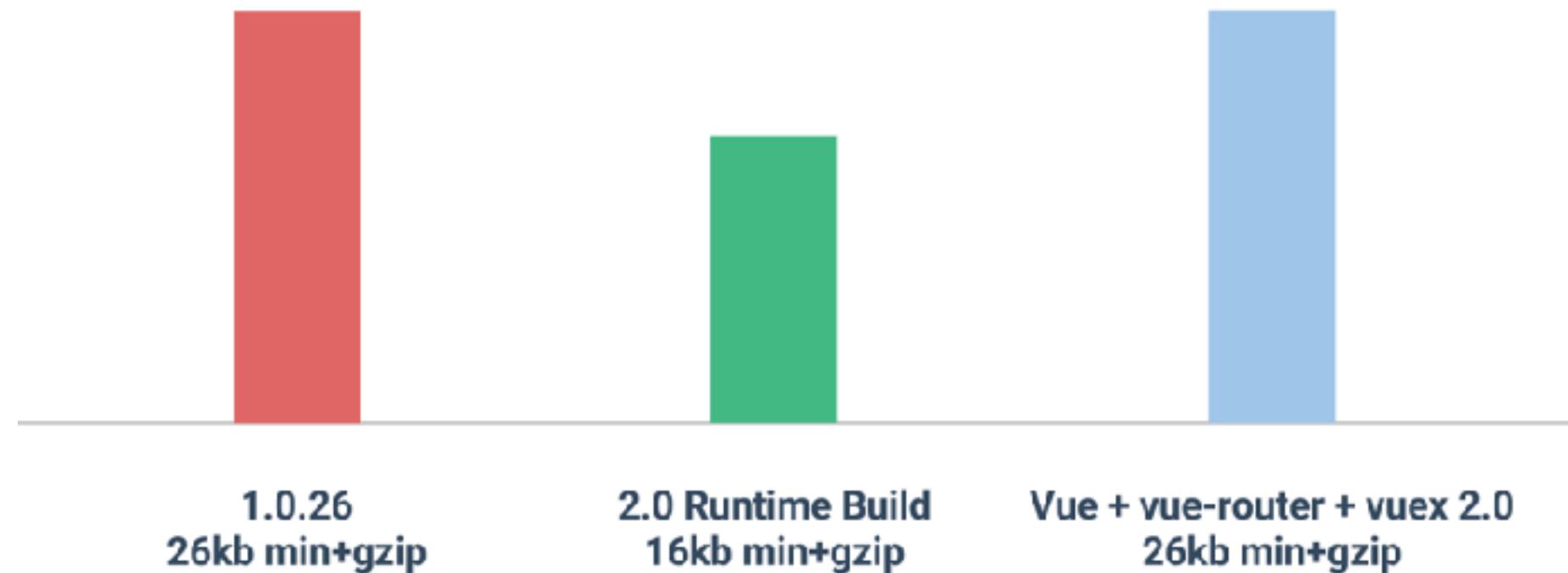
## 다른 프레임워크와의 비교

이 섹션은 작성하기 가장 까다로운 페이지이지만, 중요하다고 생각합니다. 당신은 해결하려는 문제가 있었을 것이고 문제를 해결하기 위해 다른 라이브러리를 사용했을 것입니다. 그리고 Vue가 특정 문제를 더 잘 해결할 수 있는지 알고 싶기 때문에 이 것을 보고 있을 것입니다. 이것이 우리가 당신을 위해 말하고자 하는 것입니다.

우리는 편견을 피하려고 아주 열심히 하고 있습니다. 코어 팀으로서 우리는 Vue를 좋아합니다. 우리는 우리가 더 잘 해결할 수 있다고 생각하는 몇 가지 문제가 있습니다. 이를 믿지 않는다면, 우리는 더 이상 연구하지 않을 것입니다. 우리는 공정하고 정확하기를 원합니다. React의 대체 렌더러에 대한 React의 광범위한 생태계 또는 Knockout의 IE6에 대한 브라우저 지원과 같은 다른 라이브러리가 중요한 이점을 제공하는 등이 우리가 다루려는 것 입니다.

자바 스크립트 세계가 빠르게 움직이기 때문에 이 문서를 최신 상태로 유지하는 당신의 도움을 받길 바랍니다. 정확하지 않다고 생각되는 부분이나 잘못된 부분이 있으면 [문제 제기](#)로 알려주십시오.).

## Vue JS 2.0 파일 크기 또한 초경량 프레임워크!



## Vue JS 주요 특징



- \* 낮은 학습 비용 (비교적 쉽게 느껴지는 API, HTML 기반 템플릿 문법 지원)
- \* 고속 렌더링 속도 (React 보다 빠르다고 주장함)
- \* 초 경량 프레임워크 (min + gzip 압축, 16kb)
- \* 컴포넌트 지향 UI (Page 단위가 아닌, 컴포넌트 중심/의존 개발)
- \* 리 액티브 시스템 (데이터 변경 되면, 뷰 또한 즉시 변경)
- \* 데이터 바인딩 (모델 데이터를 템플릿에 바인딩하는 시스템)

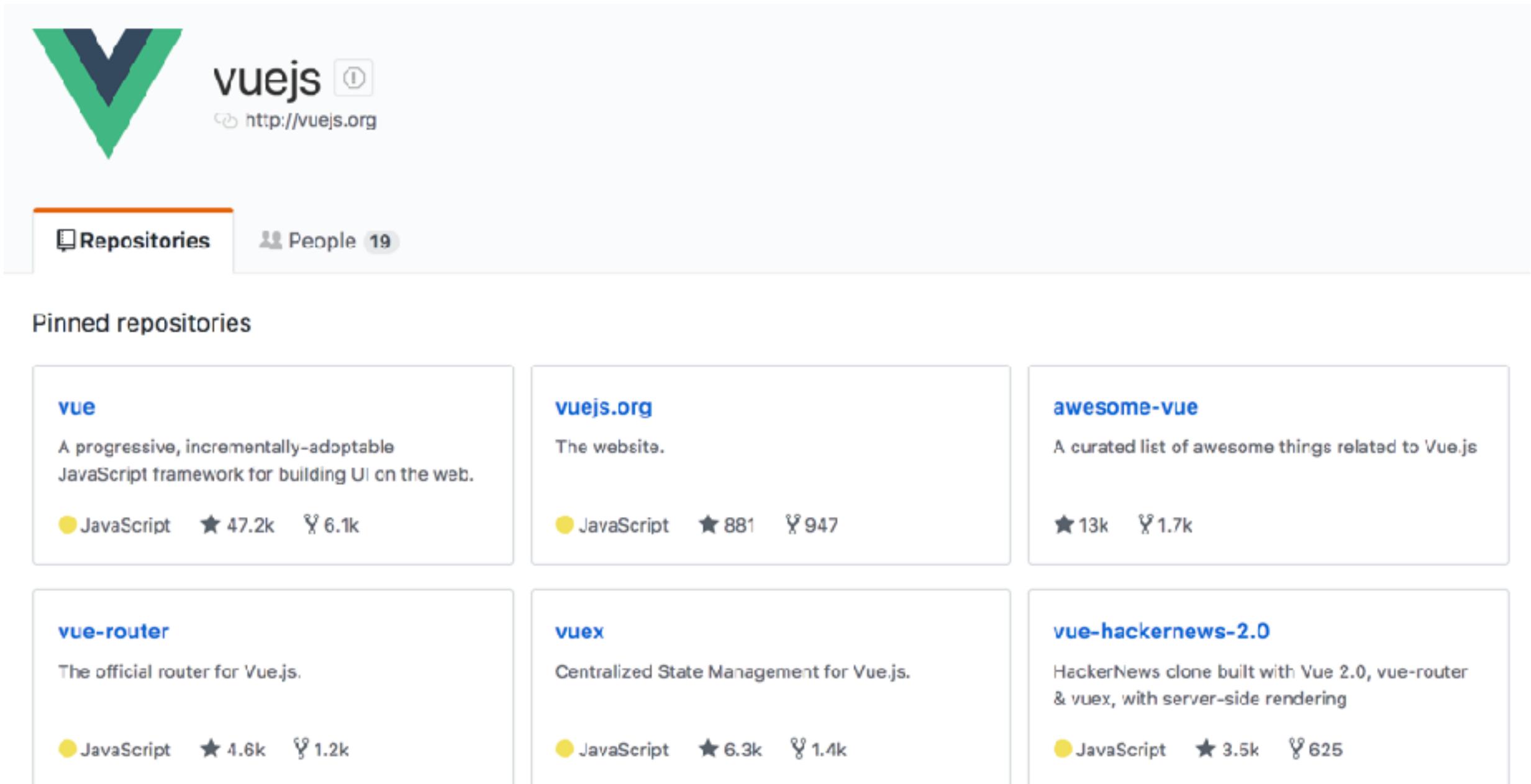
## 브라우저 지원 범위

 Android	 Firefox	 Chrome	 IE	 iPhone	 Edge	 Safari
4.2  * ✓	45  * ✓	51  7 ✓	9  7 ✓	7.0  10.9 ✓	13  10 ✓	8  10.10 ✓
5.1  * ✓			10  8 ✓	9.2  10.10 ✓		
			11  8.1 ✓			



IE 9+

Vue는 ECMAScript 5 기능을 사용하기 때문에 IE8 이하 버전을 지원하지 않습니다.  
하지만 모든 ECMAScript 5 호환 브라우저를 지원합니다



The screenshot shows the GitHub repository page for `vuejs`. The page features the Vue.js logo at the top left. Below it, there are two tabs: `Repositories` (selected) and `People 19`. The main content area is titled `Pinned repositories` and displays six repository cards:

- vue**: A progressive, incrementally-adoptable JavaScript framework for building UI on the web. (JavaScript, 47.2k stars, 6.1k forks)
- vuejs.org**: The website. (JavaScript, 881 stars, 947 forks)
- awesome-vue**: A curated list of awesome things related to Vue.js. (13k stars, 1.7k forks)
- vue-router**: The official router for Vue.js. (JavaScript, 4.6k stars, 1.2k forks)
- vuex**: Centralized State Management for Vue.js. (JavaScript, 6.3k stars, 1.4k forks)
- vue-hackernews-2.0**: HackerNews clone built with Vue 2.0, vue-router & vuex, with server-side rendering. (JavaScript, 3.5k stars, 625 forks)

# Installation 설치

## 직접 <script> 에 추가

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

! 개발 중에는 최소화 버전을 사용하지 마십시오. 일반적인 실수에 대한 모든 훌륭한 경고를 놓 치게됩니다!

개발용 버전

모든 경고 및 디버그 모드 포함

배포용 버전

경고가 제거 되고, 26.87kb min+gzip 로 압축하였습니다.

### # CDN

주천 : <https://unpkg.com/vue>는 npm에 올라간 최신 버전을 반영합니다. <https://unpkg.com/vue/>에서 npm 패키지의 원본을 찾아 볼 수도 있습니다.

또한 [JsDelivr](#)와 [cdnjs](#)를 사용할 수 있으나, 두 서비스는 동기화에 시간이 걸리므로 아직 최신 버전을 사용지 못할 수 있습니다.



프로그래시브 JavaScript 프레임워크

# Preview

데이터 바인딩 ■ 프리뷰

```
// Data
let data = {
  'message': 'Hello Vue!' // Data binding
};

// Markup (Will Change DOM)
<div class="demo">
  <input type="text">
    <p>사용자가 입력한 데이터 값은 <span class="user-input"></span> 입니다.</p>
</div>
```

```
// DOM API + Vanilla JavaScript[#]

var input      = document.querySelector('.demo input');
var user_input = document.querySelector('.demo .user-input');

user_input.textContent = data.message;

input.setAttribute('value', data.message);

input.addEventListener('keyup', function(e) {
  var user_input_data = e.target.value;
  data.message = user_input_data;
  user_input.textContent = data.message;
});
```

```
// jQuery Library

var $input      = $('.demo input');
var $user_input = $('.demo .user-input');

$user_input.text(data.message);

$input
  .val(data.message)
  .on('keyup', function(e) {
    var user_input = e.target.value;
    data.message = user_input;
    $user_input.text(data.message);
  });
}
```

// Vue JS Framework

**VM** new Vue({  
 'el': 'demo',  
 'data': data ←.....

// Markup (Will Change DOM)

<div class="demo">  
 <input type="text" v-model="message">  
 <p>사용자가 입력한 데이터 값은 {{ message }} 입니다.</p>  
 </div>

**View**

**Model**

// Data

```
let data = {  

    'message': 'Hello Vue!'  

};
```



## 생성자

모든 Vue vm은 **Vue** 생성자 함수로 root Vue 인스턴스를 생성하여 부트스트래핑됩니다.

```
var vm = new Vue({  
  // 옵션  
})
```

.js

엄격히 **MVVM 패턴**과 관련이 없지만 Vue의 디자인은 부분적으로 그것에 영감을 받았습니다. 컨벤션으로, Vue 인스턴스를 참조하기 위해 종종 변수 **vm** (ViewModel의 약자)을 사용합니다.

Vue 인스턴스를 인스턴스화 할 때는 데이터, 템플릿, 마운트할 엘리먼트, 메소드, 라이프사이클 큐브 등 의 옵션을 포함 할 수 있는 **options 객체**를 전달 해야합니다. 전체 옵션 목록은 **API reference**에서 찾을 수 있습니다.

**Vue** 생성자는 미리 정의 된 옵션으로 재사용 가능한 **컴포넌트 생성자**를 생성하도록 확장 될 수 있습니다

```
var MyComponent = Vue.extend({  
  // 옵션 확장  
}  
  
// 'MyComponent'의 모든 인스턴스는  
// 미리 정의된 확장 옵션과 함께 생성됩니다.  
var myComponentInstance = new MyComponent()
```

.js

확장된 인스턴스를 만들수는 있으나 대개 템플릿에서 사용자 지정 엘리먼트로 신인적으로 작성하는 것이 좋습니다. 나중에 **컴포넌트 시스템**에 대해 자세히 설명합니다. 지금은 모든 Vue 컴포넌트가 본질적으로 확장된 Vue 인스턴스라는 것을 알아야 합니다.

## Vue.js Data Binding

vue\_data-binding.html

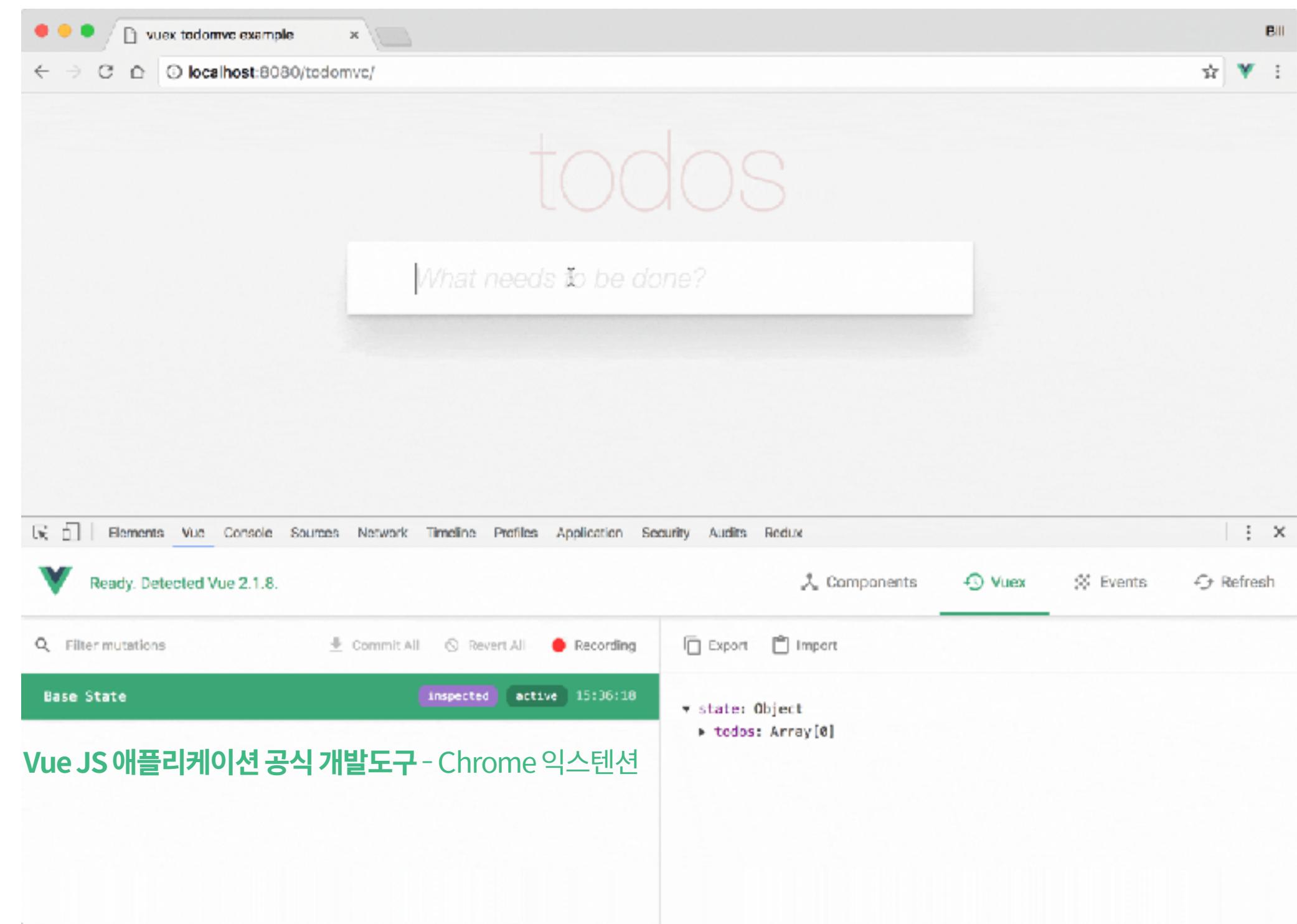
Raw

```
1 <!DOCTYPE html>
2 <html lang="ko-KR">
3 <head>
4   <meta http-equiv="X-UA-Compatible" content="IE=Edge">
5   <meta charset="UTF-8">
6   <title>VueJS - Data Binding</title>
7   <script src="https://code.jquery.com/jquery.min.js"></script>
8   <script src="https://unpkg.com/vue/dist/vue.js"></script>
9 </head>
10 <body>
11
12   <div class="demo1">
13     <input type="text">
14     <p>사용자가 입력한 데이터 값은 <span class="user-input">...</span> 입니다.</p>
15   </div>
16
17   <div class="demo2">
18     <input type="text">
19     <p>사용자가 입력한 데이터 값은 <span class="user-input">...</span> 입니다.</p>
20   </div>
21
22   <div class="demo3">
23     <input type="text" v-model="message">
24     <p>사용자가 입력한 데이터 값은 [[message]] 입니다.</p>
25   </div>
26
```



# Vue Devtools

Vue—개발 도구

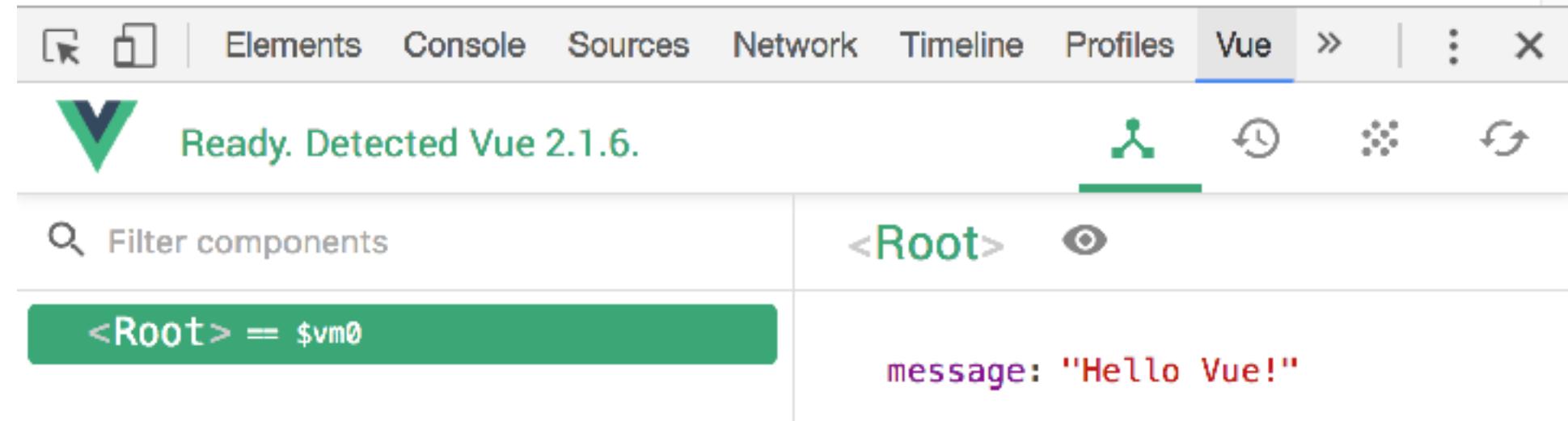


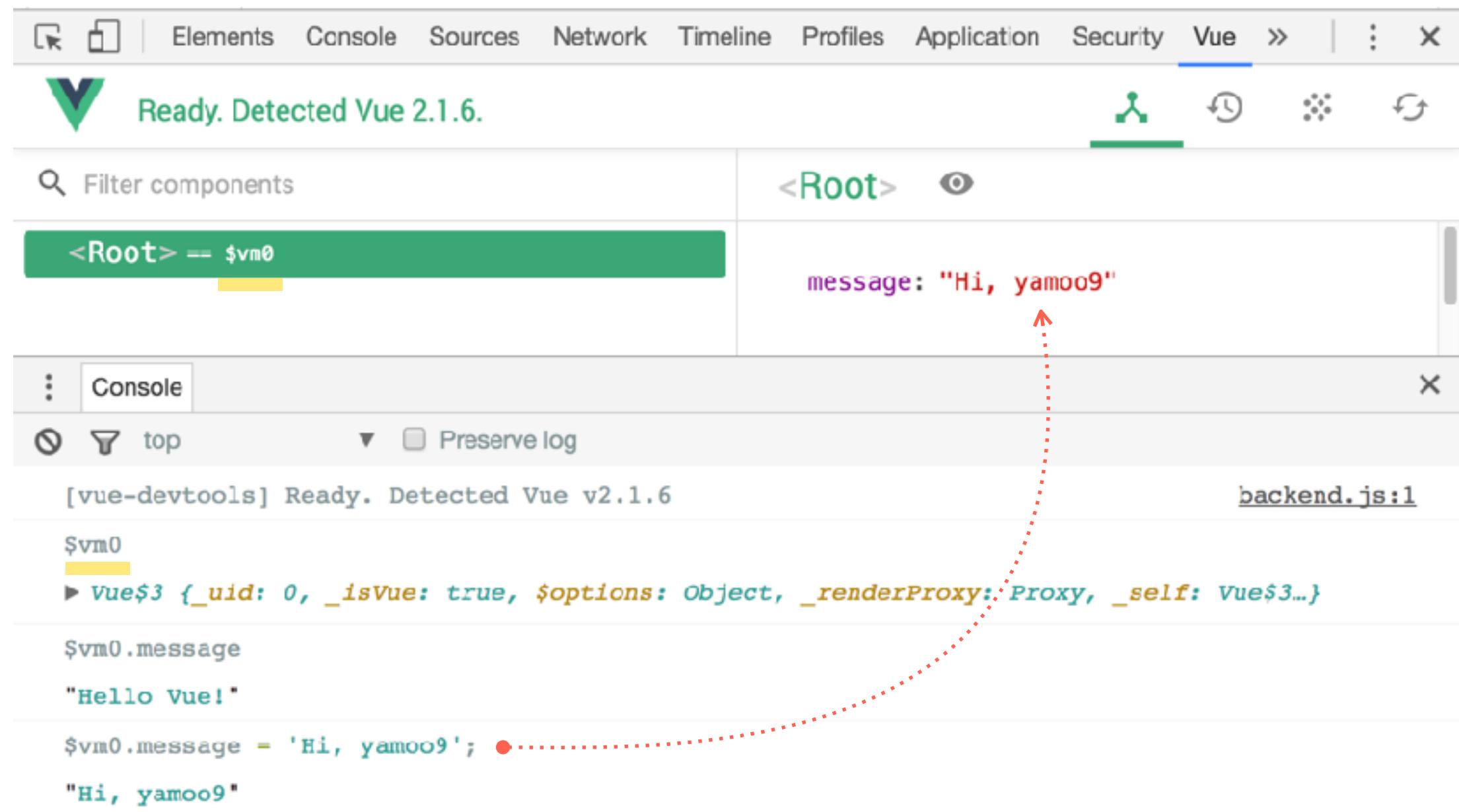
 **Vue.js devtools 3.0.8**

Chrome devtools extension for debugging Vue.js applications.

[권한](#) [세부정보](#)

시크릿 모드에서 허용  파일 URL에 대한 액세스 허용





The screenshot shows the Vue Devtools extension in the Chrome DevTools interface. The top navigation bar includes tabs for Elements, Console, Sources, Network, Timeline, Profiles, Application, Security, and **Vue**. The Vue tab is active, indicated by a blue underline.

The main area displays a component tree. A green box highlights the root component: **<Root> == \$vm0**. To the right of the tree, a message component is shown with the text: **message: "Hi, yamoo9"**.

Below the tree, the **Console** tab is selected. It shows the following logs:

- [vue-devtools] Ready. Detected Vue v2.1.6
- \$vm0
- ▶ **Vue\$3 { \_uid: 0, \_isVue: true, \$options: Object, \_renderProxy: Proxy, \_self: Vue\$3... }**
- \$vm0.message
- "Hello Vue!"
- \$vm0.message = 'Hi, yamoo9';
- "Hi, yamoo9"

A red dotted arrow points from the highlighted line '\$vm0.message = 'Hi, yamoo9'' in the console to the 'message' prop in the component tree.



프로그래시브 JavaScript 프레임워크

# Vue Fundamental

파이-기초 학습

FAST CAMPUS | Front-End Develop SCHOOL



Copyright [yamoo9.net](#) All Rights Reserved

# Template & Directives

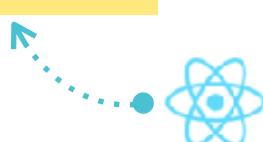
템플릿  
디렉티브(지시자)

## 템플릿 문법

Vue.js는 렌더링 된 DOM을 기본 Vue 인스턴스의 데이터에 선언적으로 바인딩 할 수 있는 HTML 기반 템플릿 구문을 사용합니다. 모든 Vue.js 템플릿은 스펙을 호환하는 브라우저 및 HTML 파서로 구문 분석 할 수 있는 유효한 HTML입니다.

내부적으로 Vue는 템플릿을 가상 DOM 렌더링 함수로 컴파일 합니다. 반응형 시스템과 결합된 Vue는 앱 상태가 변경 될 때 최소한으로 DOM을 조작하고 다시 적용할 수 있는 최소한의 컴포넌트를 지능적으로 파악할 수 있습니다.

가상 DOM 개념에 익숙하고 JavaScript의 기본 기능을 선호하는 경우 템플릿 대신 **렌더링 함수를 직접 작성할 수 있으며 선택사항으로 JSX를 지원합니다.**



## 디렉티브

디렉티브는 `v-` 접두사가 있는 특수 속성입니다. 디렉티브 속성 값은 단일 JavaScript 표현식이 됩니다. (나중에 설명할 `v-for` 는 예외입니다.) 디렉티브의 역할은 표현식의 값이 변경될 때 사이드이펙트를 반응적으로 DOM에 적용하는 것입니다. 아래 예제에서 살펴보겠습니다.

HTML

```
<p v-if="seen">이제 나를 볼 수 있어요</p>
```

여기서, `v-if` 디렉티브는 `seen` 표현의 진실성에 기반하여 `<p>` 엘리먼트를 제거 또는 삽입합니다.



# Loop & Update

반복 처리

업데이트

# Loop & Directives

반복 처리

■ 디렉티브(지시자)

```
<div class="demo">
  <h1>Vue JS 프레임워크 기능</h1>
  <ul class="vue-list">
    <li class="vue-list__item"></li>
  </ul>
</div>
```

```
new Vue({  
  'el': '.demo',  
  'data': {  
    'features': [  
      '빠른 랜더링 속도',  
      '초 경량 프레임워크',  
      '컴포넌트 UI 지향',  
      '데이터 바인딩',  
      'v-* 디렉티브 지원',  
      '리액티브 시스템',  
    ]  
  }  
});
```

```
<div class="demo">
  <h1>Vue JS 프레임워크 기능</h1>
  <ul class="vue-list">
    <li class="vue-list__item"
        v-for="feature in features"></li>
  </ul>
</div>
```

```
▼ <div class="demo">
  <h1>Vue JS 프레임워크 기능</h1>
  ▼ <ul class="vue-list">
    <li class="vue-list__item">빠른 랜더링 속도</li>
    <li class="vue-list__item">초 경량 프레임워크</li>
    <li class="vue-list__item">컴포넌트 UI 지향</li>
    <li class="vue-list__item">데이터 바인딩</li>
    <li class="vue-list__item">v-* 디렉티브 지원</li>
    <li class="vue-list__item">리액티브 시스템</li>
  </ul>
</div>
```

## # v-for

- 예상됨: `Array | Object | number | string`
- 사용방법:

원본 데이터를 기반으로 엘리먼트 또는 템플릿 블록을 여러번 렌더링합니다. 디렉티브의 값은 반복되는 현재 엘리먼트에 대한 별칭을 제공하기 위해 특수 구문인 `alias in expression` 을 사용해야 합니다.

HTML

```
<div v-for="item in items">
  {{ item.text }}
</div>
```

## Vue JS 데이터 순환 처리 및 데이터/뷰 업데이트

vue\_\_loop-list-add-items.html

Raw

```
1 <!DOCTYPE html>
2 <html lang="ko-KR">
3 <head>
4   <meta http-equiv="X-UA-Compatible" content="IE=Edge">
5   <meta charset="UTF-8">
6   <title>VueJS - Loop Lists &amp; Add List Items</title>
7   <script src="https://unpkg.com/vue@2.1.6/dist/vue.js"></script>
8 </head>
9 <body>
10
11  <div class="demo">
12    <h1>Vue JS 프레임워크 기능</h1>
13    <ul class="vue-list">
14      <li class="vue-list__item" v-for="feature in features">{{ feature }}</li>
15    </ul>
16  </div>
17
18  <script>
19  new Vue({
20    'el': '.demo',
21    'data': {
```

```
<div class="demo">
  <h1>Vue JS 프레임워크 기능</h1>
  <ul class="vue-list">
    <li
      class="vue-list__item"
      v-for="feature in features"
      v-text="feature"></li>
    </ul>
  </div>
```

## # v-text

- 예상됨: `string`
- 상세:

엘리먼트의 `textContent` 를 업데이트 합니다. `textContent` 의 일부를 갱신해야 하면 `{{ Mustache }}` 를 사용해야 합니다.

- 예제:

HTML

```
<span v-text="msg"></span>
<!-- 같습니다 -->
<span>{{msg}}</span>
```

## Vue JS 데이터 순환 처리 및 데이터/뷰 업데이트

vue\_\_loop-list-add-items.html

Raw

```
1 <!DOCTYPE html>
2 <html lang="ko-KR">
3 <head>
4   <meta http-equiv="X-UA-Compatible" content="IE=Edge">
5   <meta charset="UTF-8">
6   <title>VueJS - Loop Lists &amp; Add List Items</title>
7   <script src="https://unpkg.com/vue@2.1.6/dist/vue.js"></script>
8 </head>
9 <body>
10
11  <div class="demo">
12    <h1>Vue JS 프레임워크 기능</h1>
13    <ul class="vue-list">
14      <li class="vue-list__item" v-for="feature in features">{{ feature }}</li>
15    </ul>
16  </div>
17
18  <script>
19  new Vue({
20    'el': '.demo',
21    'data': {
```

```
<div id="app" v-cloak>
  <h1>{{app_name}}</h1>
  <ul class="reset-list vue-features">
    <li v-for="(feature, index) of vue_features">
      <div v-for="(value, key, index) of feature">
        {{ index }} {{ key }}: {{ value }}
      </div>
      <hr v-if="index !== vue_features.length - 1">
    </li>
  </ul>
```

## 배열 변경 감지

### # 변이 메소드

Vue는 감시중인 배열의 변이 메소드를 래핑하여 뷰 갱신을 트리거합니다. 래핑된 메소드는 다음과 같습니다.

- `push()`
- `pop()`
- `shift()`
- `unshift()`
- `splice()`
- `sort()`
- `reverse()`

콘솔을 열고 이전 예제의 `items` 배열로 변이 메소드를 호출하여 재생할 수 있습니다. 예:

```
example1.items.push({ message: 'Baz' })
```

## # Vue.set( target, key, value )

- 전달인자:

- `{Object | Array} target`
- `{string | number} key`
- `{any} value`

- 반환 값: 설정한 값.

- 사용방법:

객체에 대한 속성을 설정합니다. 객체가 반응형이면, 속성이 반응형 속성으로 만들어지고 뷰 업데이트를 발생시킵니다.

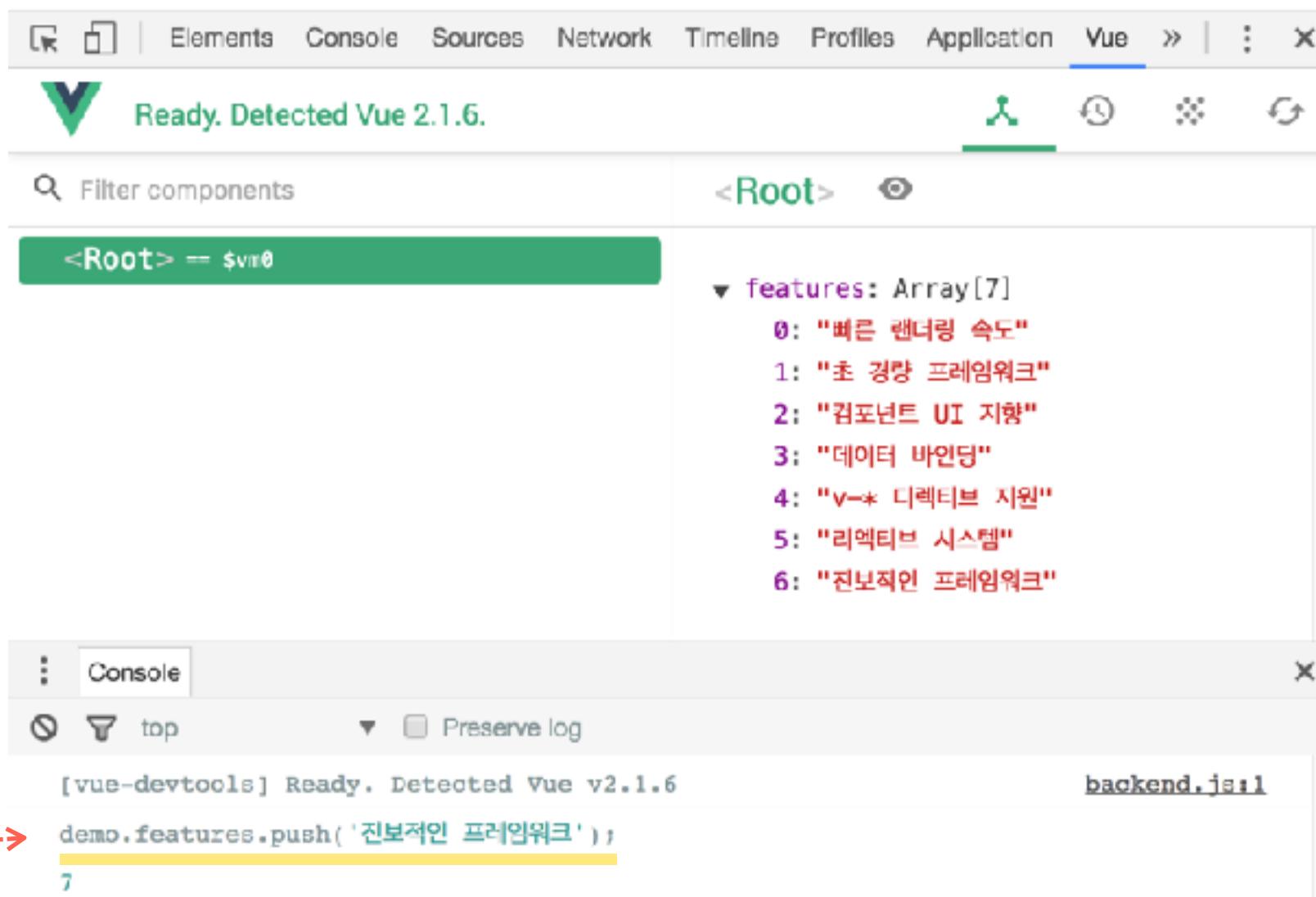
이는 Vue가 속성 추가를 감지하지 못하는 한계를 넘기 위해 사용합니다

객체는 Vue 인스턴스 또는 Vue 인스턴스의 루트 객체일 수 없습니다.

# 업데이트 Update & Hook

## Vue JS 프레임워크 기능

- 빠른 랜더링 속도
- 초 경량 프레임워크
- 컴포넌트 UI 지향
- 데이터 바인딩
- v-\* 디렉티브 지원
- 리액티브 시스템
- 진보적인 프레임워크



The screenshot shows the Vue Devtools extension integrated into the Chrome DevTools interface. At the top, the toolbar includes Elements, Console, Sources, Network, Timeline, Profiles, Application, and Vue. The Vue tab is active, displaying the message "Ready. Detected Vue 2.1.6." Below the toolbar, there's a search bar labeled "Filter components" and a component tree starting with "<Root>". A green bar highlights the component "<Root> = \$vm0". To the right of the tree, a list of features is shown:

```

    ▼ features: Array[7]
      0: "빠른 랜더링 속도"
      1: "초 경량 프레임워크"
      2: "컴포넌트 UI 지향"
      3: "데이터 바인딩"
      4: "v-* 디렉티브 지원"
      5: "리액티브 시스템"
      6: "진보적인 프레임워크"
  
```

At the bottom left, a snippet of JavaScript code is displayed:

```
let demo = new Vue({});
```

A red arrow points from this code up towards the component tree. On the right side, the "Console" tab is open, showing the output of the command "demo.features.push('진보적인 프레임워크');". The console also displays the message "[vue-devtools] Ready. Detected Vue v2.1.6".

```
<div class="demo">
  <h1>Vue JS 프레임워크 기능</h1>
  <p class="form-control">
    <label for="vue-update">아이템 추가</label>
    <input type="text" id="vue-update">
    <button type="button" class="vue-update_button">추가</button>
  </p>
  <ul class="vue-list">
    <li
      class="vue-list_item"
      v-for="feature in features"
      v-text="feature"></li>
  </ul>
</div>
```

```
// 이벤트 바인딩
var vue_update_input = document.querySelector('#vue-update');
var vue_update_button = document.querySelector('.vue-update_button');

vue_update_button.addEventListener('click', function() {
    var new_item = vue_update_input.value;
    demo.features.push(new_item);
});
```

## Vue JS 데이터 순환 처리 및 데이터/뷰 업데이트

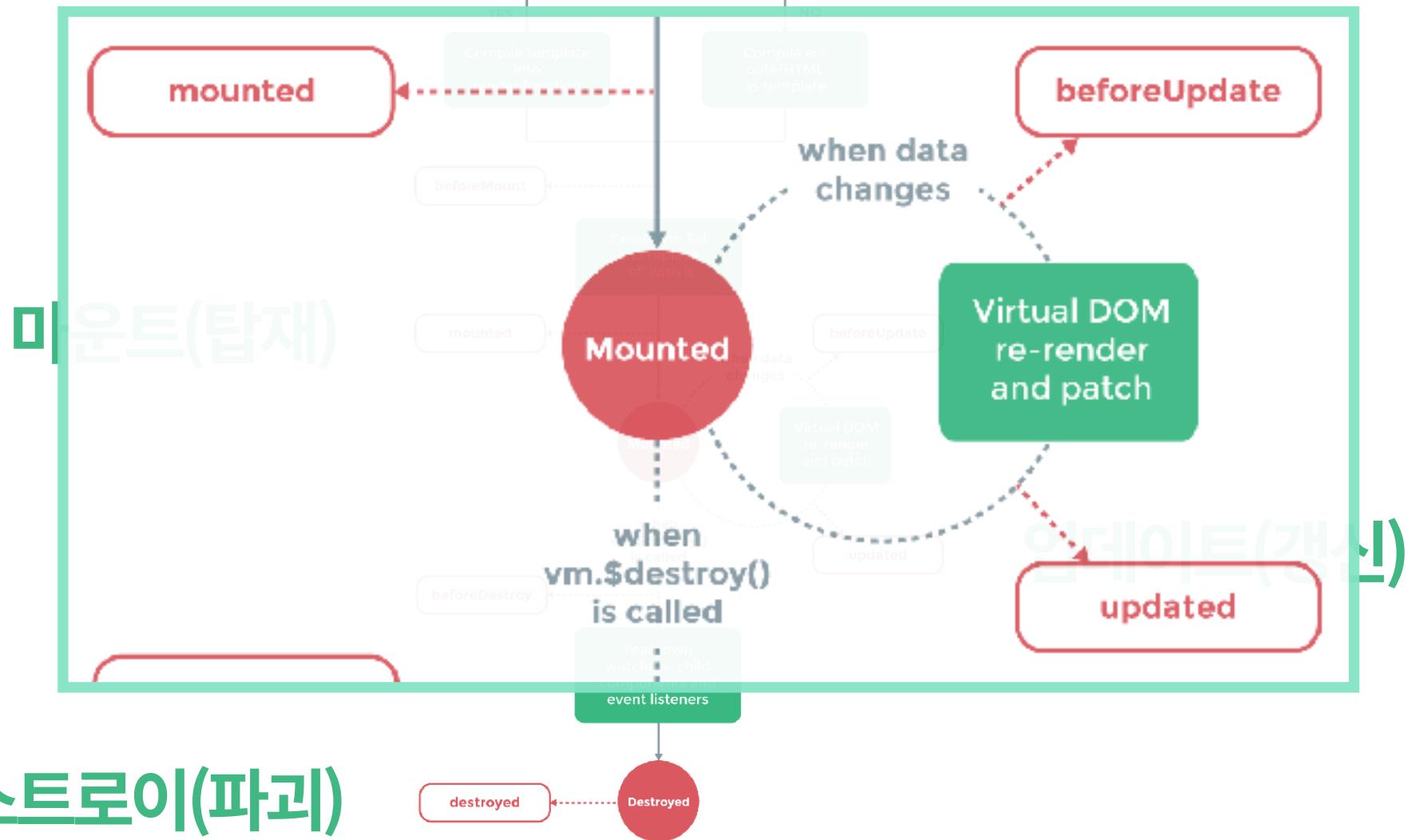
vue\_\_loop-list-add-items.html

Raw

```
1 <!DOCTYPE html>
2 <html lang="ko-KR">
3 <head>
4   <meta http-equiv="X-UA-Compatible" content="IE=Edge">
5   <meta charset="UTF-8">
6   <title>VueJS - Loop Lists &amp; Add List Items</title>
7   <script src="https://unpkg.com/vue@2.1.6/dist/vue.js"></script>
8 </head>
9 <body>
10
11  <div class="demo">
12    <h1>Vue JS 프레임워크 기능</h1>
13    <ul class="vue-list">
14      <li class="vue-list__item" v-for="feature in features">{{ feature }}</li>
15    </ul>
16  </div>
17
18  <script>
19  new Vue({
20    'el': '.demo',
21    'data': {
```

```
let demo = new Vue({  
  'el': '.demo',  
  'data': {...},  
  // 라이프사이클 후  
  mounted() {  
    // 이벤트 바인딩  
    var vue_update_input = document.querySelector('#vue-update');  
    var vue_update_button = document.querySelector('.vue-update_button');  
  
    vue_update_button.addEventListener('click', function() {  
      var new_item = vue_update_input.value;  
      demo.features.push(new_item);  
    });  
  }  
});
```

## 크리에이트(생성)



## 디스트로이(파괴)

## Vue JS 데이터 순환 처리 및 데이터/뷰 업데이트

vue\_\_loop-list-add-items.html

Raw

```
1 <!DOCTYPE html>
2 <html lang="ko-KR">
3 <head>
4   <meta http-equiv="X-UA-Compatible" content="IE=Edge">
5   <meta charset="UTF-8">
6   <title>VueJS - Loop Lists &amp; Add List Items</title>
7   <script src="https://unpkg.com/vue@2.1.6/dist/vue.js"></script>
8 </head>
9 <body>
10
11  <div class="demo">
12    <h1>Vue JS 프레임워크 기능</h1>
13    <ul class="vue-list">
14      <li class="vue-list__item" v-for="feature in features">{{ feature }}</li>
15    </ul>
16  </div>
17
18  <script>
19  new Vue({
20    'el': '.demo',
21    'data': {
```

# Event Listeners

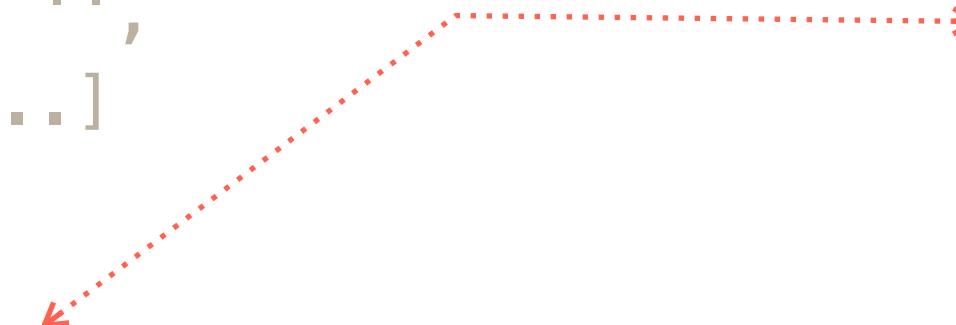
이벤트 처리기

```
let demo = new Vue({  
  'el': '.demo',  
  'data': {  
    'new_feature': '', <----->  
    'features': [  
      '빠른 랜더링 속도',  
      '초 경량 프레임워크',  
      '컴포넌트 UI 지향',  
      '데이터 바인딩',  
      'v-* 디렉티브 지원',  
      '리액티브 시스템',  
    ]  
},  
  <input  
    type="text"  
    id="vue-update"  
    v-model="new_feature">
```

```

let demo = new Vue({
  'el': '.demo',
  'data': {
    'new_feature': '',
    'features': [...]
  },
  'methods': {
    updateFeature() {
      this.features.push(this.new_feature);
      this.feature = '';
    }
  }
});
  
```

<button  
 type="button"  
 class="vue-update\_button"  
v-on:click="updateFeature">

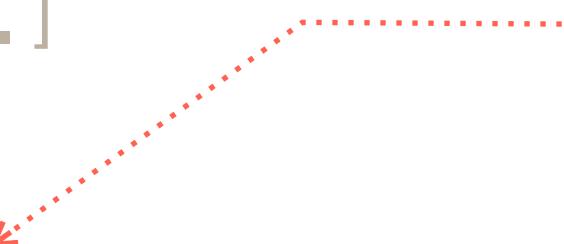


### 메소드 이벤트 핸들러

많은 이벤트 핸들러의 로직은 더 복잡할 것이므로, JavaScript를 `v-on` 속성 값으로 보관하는 것은 간단하지 않습니다. 이 때문에 `v-on` 이 호출하고자 하는 메소드의 이름을 받는 이유입니다.

```
let demo = new Vue({
  'el': '.demo',
  'data': {
    'new_feature': '',
    'features': [...]
  },
  'methods': {
    updateFeature() {
      this.features.push(this.new_feature);
      this.feature = '';
    }
  }
});
```

<input  
 type="text"  
 id="vue-update"  
 v-model="new\_feature"  
 v-on:keyup.enter="updateFeature">



### 키수식어

키보드 이벤트를 청취할 때, 종종 공동 키 코드를 확인해야 합니다. Vue는 키 이벤트를 수신할 때 `v-on`에 대한 키 수식어를 추가할 수 있습니다.

<!-- keyCode가 13일 때만 vm.submit()을 호출합니다 -->  
<input v-on:keyup.13="submit">

HTML

```

let demo = new Vue({
  'el': '.demo',
  'data': {
    'new_feature': '',
    'features': [...]
  },
  'methods': {
    updateFeature() {
      this.features.push(this.new_feature);
      this.feature = '';
    }
  }
});
  
```

약어

`<input type="text" id="vue-update" v-model="new_feature" @keyup.enter="updateFeature">`



접두사는 템플릿의 Vue 특정 속성을 식별하기 위한 시각적인 신호 역할을 합니다. 이 기능은 Vue.js를 사용하여 기존의 마크업에 동적인 동작을 적용할 때 유용하지만 일부 자주 사용되는 디렉티브에 대해서는 장점이라고 느껴질 수 있습니다. 동시에 Vue.js가 모든 템플릿을 관리하는 SPA를 만들 때 접두어의 필요성이 떨어집니다. 따라서 가장 자주 사용되는 두개의 디렉티브인 `v-bind` 와 `v-on`에 대해 특별한 약어를 제공합니다.

## # v-on

- 약어: `@`
- 예상됨: Function | Inline Statement
- 전달인자: event (required)
- 수식어:
  - `.stop` - `event.stopPropagation()` 을 호출합니다.
  - `.prevent` - `event.preventDefault()` 을 호출합니다.
  - `.capture` - 캡처 모드에서 이벤트 리스너를 추가합니다.
  - `.self` - 이벤트가 이 엘리먼트에서 전달된 경우에만 처리 됩니다
  - `.{keyCode | keyAlias}` - 특정 키에 대해서만 처리 됩니다.
  - `.native` - 컴포넌트의 루트 엘리먼트에서 네이티브 이벤트를 수신합니다.
  - `.once` - 단 한번만 처리됩니다.
  - `.left` - (2.2.0) 왼쪽 버튼 마우스 이벤트 트리거 처리기.
  - `.right` - (2.2.0) 오른쪽 버튼 마우스 이벤트 트리거 처리기.
  - `.middle` - (2.2.0) 가운데 버튼 마우스 이벤트 트리거 처리기.

## Vue JS 데이터 순환 처리 및 데이터/뷰 업데이트

vue\_\_loop-list-add-items.html

Raw

```
1 <!DOCTYPE html>
2 <html lang="ko-KR">
3 <head>
4   <meta http-equiv="X-UA-Compatible" content="IE=Edge">
5   <meta charset="UTF-8">
6   <title>VueJS - Loop Lists &amp; Add List Items</title>
7   <script src="https://unpkg.com/vue@2.1.6/dist/vue.js"></script>
8 </head>
9 <body>
10
11  <div class="demo">
12    <h1>Vue JS 프레임워크 기능</h1>
13    <ul class="vue-list">
14      <li class="vue-list__item" v-for="feature in features">{{ feature }}</li>
15    </ul>
16  </div>
17
18  <script>
19  new Vue({
20    'el': '.demo',
21    'data': {
```

# Attributes & Class Binding

속성  
클래스 속성  
연결

```
let demo = new Vue({  
  'el': '.demo',  
  'data': {  
    'link': {  
      'href' : 'http://fastcampus.co.kr/',  
      'label' : '패스트 캠퍼스',  
      'external' : true  
    }  
  }  
});
```

```
<div class="demo">

  <a v-bind:href="link.href"
    v-bind:target="link.external ? '_blank' : null"
    v-bind:title="link.label + (link.external ? '(새 탭 열림)' : '')">
    {{link.label}}
  </a>

</div>
```

```
<div class="demo">  
  
<a :href="link.href"  
  :target="link.external ? '_blank' : null"  
  :title="link.label + (link.external ? '(새 탭 열림)' : '')">  
  {{link.label}}  
</a>  
  
</div>
```

## 약어

v- 접두사는 템플릿의 Vue 특정 속성을 식별하기 위한 시각적인 신호 역할을 합니다. 이 기능은 Vue.js를 사용하여 기존의 마크업에 동적인 동작을 적용할 때 유용하지만 일부 자주 사용되는 디렉티브에 대해서 너무 장황하다고 느껴질 수 있습니다. 동시에 Vue.js가 모든 템플릿을 관리하는 SPA를 만들 때 v- 접두어의 필요성이 떨어집니다. 따라서 가장 자주 사용되는 두개의 디렉티브인 **v-bind** 와 **v-on** 에 대해 특별한 약어를 제공합니다.

## # v-bind

- 약어: `:`
- 예상됨: `any (with argument) | Object (without argument)`
- 전달인자: `attrOrProp (optional)`
- 수식어:
  - `.prop` - 속성 대신 DOM 속성으로 바인딩 ([무슨 차이가 있습니까?](#))
  - `.camel` - kebab-case 속성 이름을 camelCase로 변환합니다. (2.1.0이후 지원)

## Vue JS 속성 연결(Attributes Binding)

vue\_attributes-binding.html

```
1  <!DOCTYPE html>
2  <html lang="ko-KR">
3  <head>
4      <meta http-equiv="X-UA-Compatible" content="IE=Edge">
5      <meta charset="UTF-8">
6      <title>VueJS - Binding Attributes</title>
7      <style>
8          .demo {
9              width: 880px; margin: 0 auto;
10         }
11     </style>
12     <script src="https://unpkg.com/vue@2.1.6/dist/vue.js"></script>
13 </head>
14 <body>
15
16 <div class="demo">
17
18     <a v-bind:href="link.href"
19         v-bind:target="link.external ? '_blank' : null"
20         v-bind:title="link.label + (link.external ? '(새 탭 열린)': '')">
21         {{link.label}}
22     </a>
```

Raw

```
<a :href="link.href"
  :target="link.external ? '_blank' : null"
  :title="link.label + (link.external ? '(새 탭 열림)' : '')"
  :class="{
    'link-external': link.external
  }">
  {{link.label}}
</a>
```

- 사용방법:

동적으로 하나 이상의 컴포넌트 속성 또는 표현식을 바인딩 합니다.

`class` 또는 `style` 속성을 뮤는 데 사용될 때, Array나 Objects와 같은 추가 값 유형을 지원합니다. 자세한 내용은 아래 링크된 섹션을 참조하십시오.

속성 바인딩에 사용할 때 속성은 하위 컴포넌트에서 올바르게 선언되어야 합니다.

전달인자 없이 사용하면 속성 이름 - 값 쌍을 포함하는 객체를 바인딩 하는데 사용할 수 있습니다. 이 모드에서는 `class` 와 `style` 은 Array나 Objects를 지원하지 않습니다.

```
<a :href="link.href"
  :target="link.external ? '_blank' : null"
  :title="link.label + (link.external ? '(새 탭 열림)' : '')"
  :class="[
    { 'link-external': link.external },
    toggle_class
  ]"
  @click.prevent="toggleClass('toggle')">
  {{link.label}}
</a>
```

## 클래스와 스타일 바인딩

데이터 바인딩은 엘리먼트의 클래스 목록과 인라인 스타일을 조작하기 위해 일반적으로 사용됩니다. 이 두 속성은 `v-bind` 를 사용하여 처리할 수 있습니다. 우리는 표현식으로 최종 문자열을 계산하면 됩니다. 그러나 문자열 연결에 간섭하는 것은 짜증나는 일이며 오류가 발생하기 쉽습니다. 이러한 이유로, Vue는 `class` 와 `style` 에 `v-bind` 를 사용할 때 특별히 향상된 기능을 제공합니다. 표현식은 문자열 이외에 객체 또는 배열을 이용할 수 있습니다.

```
let demo = new Vue({  
  'el': '.demo',  
  'data': {  
    'toggle_class': '',  
    ...  
  },  
  'methods': {  
    toggleClass(class_name) {  
      var assign_class = this.toggle_class ? '' : class_name;  
      this.toggle_class = assign_class;  
    }  
  }  
});
```

## Vue JS 속성 연결(Attributes Binding)

vue\_attributes-binding.html

```
1  <!DOCTYPE html>
2  <html lang="ko-KR">
3  <head>
4      <meta http-equiv="X-UA-Compatible" content="IE=Edge">
5      <meta charset="UTF-8">
6      <title>VueJS - Binding Attributes</title>
7      <style>
8          .demo {
9              width: 880px; margin: 0 auto;
10         }
11     </style>
12     <script src="https://unpkg.com/vue@2.1.6/dist/vue.js"></script>
13 </head>
14 <body>
15
16 <div class="demo">
17
18     <a v-bind:href="link.href"
19         v-bind:target="link.external ? '_blank' : null"
20         v-bind:title="link.label + (link.external ? '(새 탭 열린)': '')">
21         {{link.label}}
22     </a>
```

Raw

# Computed Properties

계산된 속성

```
<div class="demo">  
  <p>{{ message.split(' ').reverse().join(' ') }}</p>  
</div>
```

```
let demo = new Vue({  
  el: '.demo',  
  data: {  
    message: '진보적인 프레임워크'  
  }  
});
```

### 계산된 속성

템플릿 내에서 사용하는 표현식은 매우 편리하지만 단순한 연산에만 사용해야 합니다. 너무 많은 로직을 템플릿에 넣으면 유지보수가 어려워 질 수 있습니다.

```
<div class="demo">
  <input type="text" v-model="message" aria-label="메시지 입력">
  <p><strong>메시지 반전</strong>: {{ reverse_message }}</p>
</div>
```

```
let demo = new Vue({
  'el': '.demo',
  'data': {
    'message': '진보적인 프레임워크'
  },
  'computed': {
    reverse_message() {
      return this.message.split('').reverse().join('');
    }
  }
});
```

## Vue JS 계산된 속성 활용

[Raw](#)

```
vue_computed_properties.html

1  <!DOCTYPE html>
2  <html lang="ko-KR">
3  <head>
4      <meta http-equiv="X-UA-Compatible" content="IE=Edge">
5      <meta charset="UTF-8">
6      <title>VueJS - Computed Properties</title>
7      <style>
8          .demo {
9              width: 880px; margin: 5rem auto;
10         }
11     </style>
12     <script src="https://unpkg.com/vue@2.1.6/dist/vue.js"></script>
13 </head>
14 <body>
15
16 <div class="demo">
17     <input type="text" v-model="message" aria-label="메시지 입력">
18     <p><strong>메시지 반전</strong>: {{ reverse_message }}</p>
19 </div>
```

## # computed

- 타입: `{ [key: string]: Function | { get: Function, set: Function } }`
- 상세:

Vue 인스턴스에 추가되는 계산된 속성입니다. 모든 getter와 setter는 자동으로 `this` 컨텍스트를 Vue 인스턴스에 바인딩 합니다.

**!** 계산된 속성을 정의 할 때 화살표 함수를 사용하면 안됩니다. 화살표 함수가 부모 컨텍스트를 바인딩하기 때문에 `this` 는 Vue 인스턴스가 아니며 `this.a` 는 정의되지 않습니다.

## # 계산된 캐싱 vs 메소드

표현식에서 메소드를 호출하여 같은 결과를 얻을 수 있다는 사실을 알고 있을 것입니다.

HTML

```
<p>뒤집한 메시지: "{{ reverseMessage() }}"</p>
```

JS

```
// 컴포넌트 내부
methods: {
  reverseMessage: function () {
    return this.message.split(' ').reverse().join(' ')
  }
}
```

계산된 속성 대신 메소드와 같은 함수를 정의할 수 있습니다. 최종 결과에 대해 두 가지 접근 방식은 서로 동일합니다. 하지만 차이점은 계산된 속성은 종속성에 따라 캐시된다는 것입니다. 계산된 속성은 종속성 중 일부가 변경된 경우에만 다시 계산 됩니다. 이것은 `message` 가 변경되지 않는 한, 계산된 속성인 `reversedMessage` 에 대한 다중 접근은 함수를 다시 수행할 필요 없이 이전에 계산된 결과를 즉시 반환한다는 것을 의미합니다.

계산된 속성은  
종속된 속성 값이 변경될 경우에만 다시 계산되고,  
그렇지 않은 경우는 처리되지 않습니다.

# Computed 계산된 속성 VS Watched 감시된 속성

```
<div id="app">
  <p class="fullname">{{ fullname }}</p>
  <p>
    <input type="text" class="first" v-model="first">
    <input type="text" class="last" v-model="last">
  </p>
</div>
```

```

let vm = new Vue({
  el: '#app',
  data: {
    first: 'yamoo9',
    last: 'gun',
    fullname: ''
  },
  mounted() {
    this.fullname = `${this.first} ${this.last}`;
  },
  watch: {
    first(value) {
      this.fullname = `${value} ${this.last}`;
    },
    last(value) {
      this.fullname = `${this.first} ${value}`;
    }
  }
});

```

## # 계산된 속성 vs 감시된 속성

Vue는 Vue 인스턴스의 데이터 변경을 관찰하고 이에 반응하는 보다 일반적인 속성 감시 방법을 제공합니다. 다른 데이터 기반으로 변경할 필요가 있는 데이터가 있는 경우, 특히 AngularJS를 사용하던 경우 `watch` 를 남용하는 경우가 있습니다. 하지만 `watch` 콜백보다 계산된 속성을 사용하는 것이 더 좋습니다. 다음 예제를 고려하십시오.

```
let vm = new Vue({  
  el: '#app',  
  data: {  
    first: 'yamoo9',  
    last: 'gun'  
  },  
  computed: {  
    fullname(){  
      return this.first + ' ' + this.last;  
    }  
  }  
});
```

# Computed Setter

설정 가능한 계산된 속성

```
<div id="app">
  <p class="fullname">{{ fullname }}</p>
  <p class="fullname">fullname: <input type="text" v-model="fullname"></p>
  <p>
    <input type="text" class="first" v-model="first">
    <input type="text" class="last" v-model="last">
  </p>
</div>
```

```
let vm = new Vue({
  el: '#app',
  data: {
    first: 'yamoo9',
    last: 'gun'
  },
  computed: {
    fullname: {
      get(){
        return this.first + ' ' + this.last;
      },
      set(newValue) {
        newValue = newValue.split(' ');
        let first = newValue[0];
        let last = newValue[newValue.length - 1];
        this.first = first;
        this.last = last;
      }
    }
  }
});
```

## # 계산된 Setter

계산된 속성은 기본적으로 getter만 가지고 있지만, 필요한 경우 setter를 제공할 수 있습니다.

# Computed DEMO

계산된 속성 × 데모

```
let demo = new Vue({  
  'el': '.demo',  
  'data': {  
    'tasks': [  
      { 'description': '서점 가서 신간 도서 읽기', 'completed': true },  
      { 'description': 'Vue.js 영상강의 녹화', 'completed': false },  
      { 'description': '해외 여행 계획', 'completed': false },  
      { 'description': '봄맞이 대청소', 'completed': false },  
      { 'description': 'Front-End 뉴스레터 읽기', 'completed': true }  
    ]  
  }  
});
```

```
<div class="demo">  
  
  <h2>완료 업무</h2>  
  <ul>  
    <li  
      v-for="task in tasks"  
      v-if="task.completed"  
      >{{ task.description }}</li>  
    </ul>  
  
</div>
```

## # v-if

- 예상됨: `any`
- 사용방법:

표현식 값의 참 거짓을 기반으로 엘리먼트를 조건부 렌더링 합니다. 엘리먼트 및 포함된 디렉티브 / 컴포넌트는 토글하는 동안 삭제되고 다시 작성됩니다. 엘리먼트가 `<template>` 엘리먼트인 경우 그 내용은 조건부 블록이 됩니다.

조건이 변경될 때 전환이 호출 됩니다.

- [Source](#)

## Vue JS 계산된 속성 활용

```
vue_computed_properties.html
```

```
1  <!DOCTYPE html>
2  <html lang="ko-KR">
3  <head>
4    <meta http-equiv="X-UA-Compatible" content="IE=Edge">
5    <meta charset="UTF-8">
6    <title>VueJS - Computed Properties</title>
7    <style>
8      .demo {
9        width: 880px; margin: 5rem auto;
10       }
11    </style>
12    <script src="https://unpkg.com/vue@2.1.6/dist/vue.js"></script>
13  </head>
14  <body>
15
16  <div class="demo">
17    <input type="text" v-model="message" aria-label="메시지 입력">
18    <p><strong>메시지 반전</strong>: {{ reverse_message }}</p>
19  </div>
```

Raw

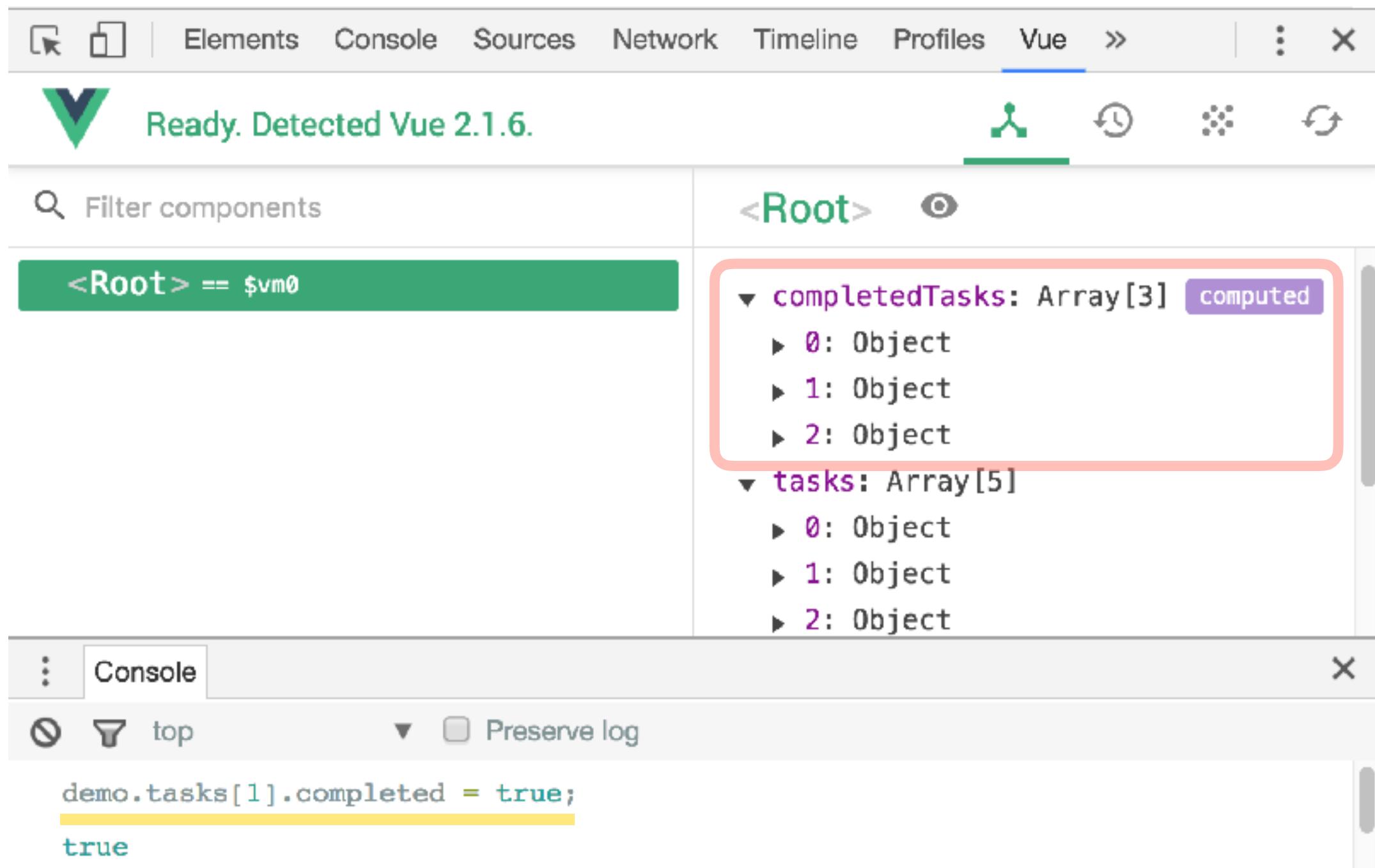
```
<div class="demo">

  <h2>수행할 업무</h2>
  <ul>
    <li v-for="task in tasks">{{ task.description }}</li>
  </ul>

  <h2>완료 업무</h2>
  <ul>
    <li v-for="task in completedTasks">{{ task.description }}</li>
  </ul>

</div>
```

```
let demo = new Vue({  
  'el': '.demo',  
  'data': {  
    'tasks': [...]  
  },  
  'computed': {  
    completedTasks() {  
      return this.tasks.filter((task)=>task.completed);  
    }  
  }  
});
```



The screenshot shows the Chrome DevTools interface with the Vue tab selected. The main area displays the state of the application's root component, which is identified as `<Root> == $vm0`. A search bar labeled "Filter components" is present. On the right, the component tree shows two arrays: `completedTasks` (an array of length 3) and `tasks` (an array of length 5). The `completedTasks` array is highlighted with a red border. The `tasks` array is also visible below it. In the bottom-left corner of the DevTools, there is a "Console" tab with a log entry:

```
demo.tasks[1].completed = true;
true
```

## Vue JS 계산된 속성 활용

[Raw](#)

```
vue_computed_properties.html

1  <!DOCTYPE html>
2  <html lang="ko-KR">
3  <head>
4      <meta http-equiv="X-UA-Compatible" content="IE=Edge">
5      <meta charset="UTF-8">
6      <title>VueJS - Computed Properties</title>
7      <style>
8          .demo {
9              width: 880px; margin: 5rem auto;
10         }
11     </style>
12     <script src="https://unpkg.com/vue@2.1.6/dist/vue.js"></script>
13 </head>
14 <body>
15
16 <div class="demo">
17     <input type="text" v-model="message" aria-label="메시지 입력">
18     <p><strong>메시지 반전</strong>: {{ reverse_message }}</p>
19 </div>
```



# Street Fighter

스트리트 파이터

