The Majesty of Vue.js 2

Alex Kyriakidis

Kostas Maniatis

The Majesty of Vue.js 2 (Korean)

The Majesty of Vue.js 2의 한국어판 입니다.

Alex Kyriakidis, Kostas Maniatis, ChangJoo Park(박창주)

This book is for sale at http://leanpub.com/vuejs2-korean

This version was published on 2017-06-26

차례

소개

Vue.js 소개

Vue.js 개요

Vue(뷰/vju：/라고 발음합니다. view와 같음)는 사용자 인터페이스를 개발하기 위한 프로그레시브 프레임워크입니다. 다른 프레임워크와 다르게 Vue는 점진적으로 채택할 수 있게 설계하였습니다. 코어 라이브러리는 **보이는 부분**에만 초점을 맞추어 다른 라이브러리나 기존 프로젝트와 통합하기 매우 쉽습니다. Vue는 현대적인 도구 및 지원하는 라이브러리[1]와 함께 사용하면 정교한 싱글 페이지 애플리케이션을 완벽하게 만들 수 있습니다.

글을 읽는 여러분이 숙련된 프론트엔드 개발자인 경우 Vue.js를 다른 라이브러리/프레임워크와의 비교한 공식 가이드의 다른 프레임워크와의 비교[2]를 확인하세요.

Vue.js의 핵심적인 자세한 내용은 Vue.js 공식 가이드[3]를 참조하세요.

Vue.js를 사용하는 사람들의 의견

“Vue.js는 자바스크립트를 사랑하게 만들었습니다. 사용법이 매우 쉽고 즐겁습니다. 기본 서비스를 확장할 수 있는 플러그인 및 도구를 포함하는 훌륭한 생태계를 갖추고 있습니다. 작거나 큰 모든 프로젝트에 빠르게 추가할 수 있으며 이를 위해 코드 몇 줄만 작성하면 시작할 수 있습니다. Vue.js는 빠르고 가벼운 프론트엔트 개발의 미래입니다!”

—Alex Kyriakidis

“자바스크립트를 시작했을 때 기쁘게도 많은 가능성을 보았습니다. 친구가 Vue.js를 배우라고 제안했을 때 나는 친구의 조언을 따랐습니다. 튜토리얼을 읽는 동안 이전에 Vue에 시간을 투자했다면 지금까지 했던 모든 것을 더 잘 할 수 있지 않았을까 생각했습니다. 개인적인 생각이지만 업무를 빠르게 하고 싶다면 Vue.js는 정말 필요했던 멋지고 쉬운 자바스크립트 프레임워크 입니다.”

—Kostas Maniatis

---

[1] https://github.com/vuejs/awesome-vue#libraries--plugins

[2] https://kr.vuejs.org/v2/guide/comparison.html

[3] http://vuejs.org/guide/overview.html

"내 말을 기억하세요. Vue.js는 2016년에 하늘에 쏘아올린 로켓과 같은 인기를 얻을 것 입니다. 정말 좋습니다."

— Jeffrey Way

"Vue는 자바스크립트 프레임워크 중 항상 찾았던 것입니다. 이것은 당신과 함께 성장할 프레임워크 입니다. 한 페이지에 화면을 보여주거나 Vuex 및 Vue Router를 사용하여 고급 싱글 페이지 애플리케이션을 만들 수 있습니다. 내가 본 것 중에 가장 세련된 자바스크립트 프레임워크 입니다."

— Taylor Otwell

"Vue.js는 SPA와 마찬가지로 서버 측 렌더링 된 애플리케이션에서 사용하는 것이 자연스럽다고 느낀 첫 번째 프레임워크 입니다. 싱글 페이지 안에 작은 위젯이 필요한지 아니면 복잡한 클라이언트를 만들 것인지 관계없이 과하거나 부족한 느낌이 전혀 없었습니다."

— Adam Wathan

"Vue.js는 사용하기 쉽고 이해하기 쉬운 프레임워크 입니다. 다른 사람들이 누가 더 복잡하게 만들 수 있는지 경쟁하는 상황에서 신선한 공기 같습니다."

— Eric Barnes

"내가 Vue.js를 좋아하는 이유는 내가 하이브리드 디자이너/개발자 이기 때문입니다. 나는 React, Angular 및 다른 몇가지 프레임워크를 살펴 보았지만 학습 곡선과 용어의 어려움 때문에 사용하기 어려웠습니다. Vue.js는 내가 이해한 첫 번째 프레임워크 입니다. 또한 나와 같은 자바스크립트 사용자들에게 쉬울 뿐 아니라 Angular 및 React 에 대한 경험이 풍부한 개발자도 주목하고 Vue.js를 좋아하고 있는 것을 알고 있습니다."

—Jack Barham

환영합니다

The Majesty of Vue 2에 관하여

이 책은 빠르게 유행하고 있는 자바스크립트 프레임워크인 Vue.js에 대한 안내서입니다!

얼마 전에, Laravel과 Vue.js를 기반으로 새 프로젝트를 시작했습니다. Vue.js 가이드와 몇 가지 예제를 모두 읽은 후 웹에 Vue.js에 대한 자료가 부족함을 알게 되었습니다. 프로젝트를 개발하면서 쌓인 경험을 사람들과 나누고자 책을 쓰게 되었습니다. 최근 Vue.js 2 버전이 출시 되었습니다. 그래서 모든 예제와 관련된 내용을 다시 작성해서 새로운 책을 쓰기로 결정했습니다.

이 책은 격식을 차리는 대신, 직관적이며 쉽게 따라 할 수 있게 쓰였으며 모든 예제는 모든 사람에게 적절한 방향을 제공할 수 있도록 자세하게 설명합니다.

가장 기초적인 내용부터 시작하여 많은 예제와 함께 Vue.js의 중요한 내용을 다룹니다.

이 책이 끝날 때쯤이면 Vue.js를 이용해 빠른 프론트엔드 애플리케이션을 만들거나 기존 프로젝트의 성능을 향상할 수 있을 것입니다.

이 책은 어떤 사람이 읽어야 하나요?

현대적인 웹 개발에 대해 배워본 사람은 부트스트랩과 많은 자바스크립트 프레임워크를 사용해 봤을 것입니다. 이 책은 가볍고 간단한 자바스크립트 프레임워크를 배우려는 사람을 위해 쓰여졌습니다. 아주 많은 사전지식이 필요하지는 않으나 기본적으로 HTML과 자바스크립트에 익숙해져야 합니다. 지금 시점에서 문자열과 객체의 차이를 설명할 수 없다면 조금 더 공부를 해야할 필요가 있습니다. 이 책은 Vue.js를 처음 사용하는 개발자는 물론 이미 Vue.js를 사용하고 더 깊이 공부하려는 개발자 모두에게 유용할 것 입니다. 또한 다른 라이브러리에서 Vue.js 2로 옮기려는 개발자에게도 유용합니다.

저자와 연락하려면

이 책에 대해 문의하거나, 피드백을 보내거나, 관심이 필요한 기타 사항에 대해 궁금한 점이 있으면 언제든지 문의 해 주십시오.

Name	Email	Twitter
The Majesty of Vue.js	hello@tmvuejs.com	@tmvuejs
Alex Kyriakidis	alex@tmvuejs.com	@hootlex
Kostas Maniatis	kostas@tmvuejs.com	@kostaskafcas

이 책의 번역자는 박창주입니다. 번역에 대한 의견 및 제안은 pcjpcj2@gmail.com 또는 트위터(@pcjpcj2)로 문의하십시오.

혼자 해보기

코드를 작성하면서 익히는 것이 가장 좋은 방법입니다. 스스로 해결하고 실제로 테스트할 수 있도록 풀어볼 만한 문제를 각 장의 마지막에 준비해 두었습니다. 가능한 한 많이 시도하여 문제를 해결하고 Vue.js에 대해 더 잘 이해하는 것을 적극적으로 권장합니다. 생각하고 있는 것을 시도하는 것을 두려워하지 마세요. 조금만 노력하면 더 많은 것을 할 수 있습니다. 어쩌면 몇 가지 다른 예제나 방법으로 적절한 아이디어를 얻을 수 있습니다. 물론 너무 과한 문제는 없을 것이며 힌트와 잠재적인 해결 방법을 함께 제공합니다.

이제 긴 여행을 시작할 때입니다!

예제 코드

이 책에서 사용된 대부분의 예제 코드는 Github에 있습니다. 코드를 살펴보기 위해 여기[4]에 들어가보세요.

다운받아서 보는 것이 더 편하면 '.zip' 파일을 내려받으세요[5].

예제 코드를 사용하면 책에 있는 내용을 복사하고 붙여넣으며 테스트하는 끔찍한 일을 피할 수 있습니다.

오탈자

정확한 컨텐츠를 제공하기 위해 모든 노력을 기울였음에도 실수가 있을 수 있습니다. 이 책을 읽으시면서 실수를 발견하면 알려주세요. 이는 다른 독자들을 혼란으로 부터 막을 수 있고 이 책의 개선된 후속 버전에 큰 도움이 됩니다. 만약 오탈자를 발견했다면 Github 저장소[6]에 이슈를 남겨주세요.

표기 규칙

이 책에서는 다음의 표기 규칙이 사용됩니다.

코드 블럭은 아래와 같습니다.

---

[4] https://github.com/hootlex/the-majesty-of-vuejs-2

[5] https://github.com/hootlex/the-majesty-of-vuejs-2/archive/master.zip

[6] https://github.com/hootlex/the-majesty-of-vuejs-2

JavaScript {lang="javascript"} function(x, y){ // 이 것은 주석입니다 }

글의 코드는 다음과 같이 표시 됩니다 "고정된 폭을 가지는 반응형 컨테이너는 `.container`를 사용합니다."

새 용어와 **중요 단어**는 굵게 표시 합니다.

팁, 노트 및 경고는 다음과 같이 표시됩니다.

이 것은 경고 입니다.

이 곳에는 경고나 주의 사항을 표시합니다.

이 것은 팁 입니다.

이 곳에는 팁 또는 추천 사항을 표시합니다.

이 것은 정보 입니다.

이 곳에는 약간의 중요한 내용을 표시합니다.

이 것은 노트 입니다.

주제에 대한 노트를 표시합니다.

이 것은 힌트 입니다.

주제에 대한 힌트를 표시합니다.

이 것은 터미널 명령어 입니다.

터미널에서 실행할 명령어를 표시합니다.

이 것은 비교를 위한 내용입니다.

주제와 관련된 비교에 대한 내용을 표시합니다.

이 것은 Github 링크 입니다.

링크는 이 책의 저장소로 연결되며 코드 샘플 및 각 장의 과제를 찾을 수 있습니다.

Vue.js Fundamentals

상호작용

이 장에서는 이전 예제를 확장하여 '메소드', '이벤트 핸들링' 및 '계산된 속성'과 관련된 새로운 내용들을 알아봅니다. 몇가지 예제를 만드는데 여러가지 다양한 접근을 해보겠습니다. 계산기와 같이 작은 애플리케이션을 만들며 Vue의 상호작용에 대한 기능을 사용해 보겠습니다.

이벤트 핸들링

`HTML` 이벤트는 DOM 엘리먼트에서 일어납니다. `HTML` 페이지에서 Vue.js를 사용하면 **반응적으로** 이벤트가 발생합니다.

이벤트는 기본적인 사용자와의 상호작용에서 렌더링 모델에서 일어나는 일까지 모든 것을 할 수 있습니다. `HTML` 이벤트의 몇가지 예 입니다.

- 웹 페이지가 완전히 불러진 경우
- 사용자 입력 폼이 변경된 경우
- 버튼이 클릭된 경우
- 폼이 제출된 경우

이벤트 핸들링의 가장 중요한 점은 이벤트가 발생할 때마다 무언가 할 수 있다는 것 입니다. Vue.js는 DOM 이벤트를 **청취**할 수 있게 **`v-on`** 디렉티브를 사용해야 합니다. **`v-on`** 디렉티브는 이벤트 리스너를 엘리먼트에 붙입니다. 이벤트의 유형은 전달인자로 표시됩니다. 예를 들어 **`v-on:keyup`**은 **`keyup`** 이벤트를 청취합니다.

정보

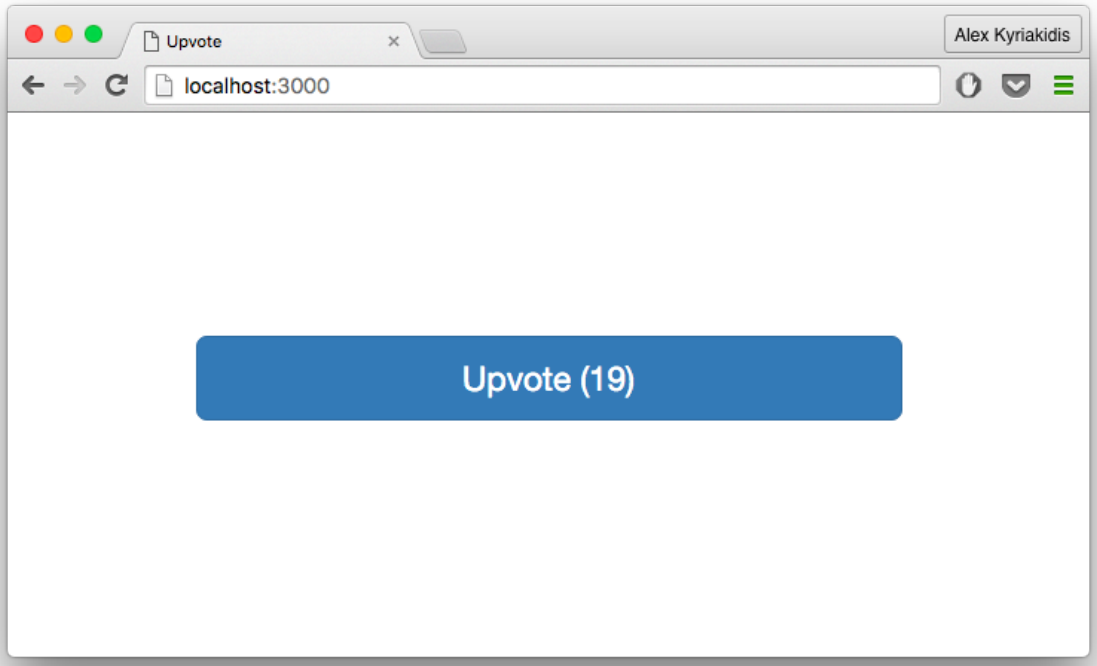
`keyup` 이벤트는 사용자가 키를 놓을 때 발생합니다. `HTML` 이벤트의 전체 목록[7]입니다.

인라인 이벤트 핸들링

말하는 것만으로도 충분하지만 좀 더 나아가 이벤트 핸들링에 대해 알아봅시다. 아래에 클릭한 만큼 투표 수를 추가하는 'Upvote' 버튼 예제가 있습니다.

[7] http://www.w3schools.com/tags/ref_eventattributes.asp

```
<html>
<head>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" rel="st\
ylesheet">
<title>Upvote</title>
</head>
<body>
    <div class="container">
        <button v-on:click="upvotes++">
            Upvote! {{upvotes}}
        </button>
    </div>
</body>
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.3.4/vue.j\
s"></script>
<script type="text/javascript">
new Vue({
    el: '.container',
    data: {
        upvotes: 0
    }
})
</script>
</html>
```



투표 카운터

데이터에는 **upvotes** 변수가 있습니다. 이 경우, **click** 이벤트 리스너를 그 옆의 코드로 바인딩 합니다. 따옴표 안에 있는 내용은 버튼을 클릭할 때 마다 upvotes를 하나씩 증가합니다. 증가 연산자(**upvotes++**)를 이용합니다.

메소드를 이용한 이벤트 핸들링

이제 메소드를 사용하여 이전과 똑같은 예제를 만들어 보겠습니다. Vue.js의 메소드는 특정 작업을 하도록 설계된 코드 블럭입니다. 메소드를 실행하려면, 메소드를 정의한 다음 호출해야 합니다.

```
1  <html>
2  <head>
3  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"  rel="st\
4  ylesheet">
5  <title>Upvote</title>
6  </head>
7  <body>
8      <div class="container">
9          <button v-on:click="upvote">
```

```
10              Upvote! {{upvotes}}
11          </button>
12      </div>
13  </body>
14  <script type="text/javascript"  src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.3.4/vue.j\
15  s"></script>
16  <script type="text/javascript">
17  new Vue({
18      el:  '.container',
19      data: {
20          upvotes: 0
21      },
22      // **`methods`**객체 아래에 메소드를 정의하세요
23      methods: {
24          upvote: function(){
25              // **`this`**는 메소드 안에서 Vue인스턴스를 가리킵니다
26              this.upvotes++;
27          }
28      }
29  })
30  </script>
31  </html>
```

메소드 'upvote'에 클릭 이벤트를 바인딩 했습니다.

이전과 마찬가지로 동작합니다. 하지만 코드를 읽을 때 명확하며 이해하기 쉽습니다.

주의사항

이벤트 핸들러는 **하나의 명령문만** 실행하도록 제한됩니다.

v-on 축약형

프로젝트에서 **v-on**을 사용하여 작업을 하고 있다면 HTML이 빠르게 지저분해 질 것입니다. 고맙게도, **v-on**의 축약형인 **@**이 있습니다. **@**은 **v-on:**을 대체합니다. 코드를 매우 깔끔하게 만들어 줍니다. 이러한 축약형은 선택 사항입니다. **@**를 사용하면 이전 예제의 버튼이 다음과 같이 됩니다.

v-on:을 이용한 'click' 이벤트 처리

```
<button v-on:click="upvote">
    Upvote! {{upvotes}}
</button>
```

@ 축약형을 이용한 'click' 이벤트 처리

```
<button @click="upvote">
    Upvote! {{upvotes}}
</button>
```

이벤트 수식어

이제 계산기 애플리케이션을 만들어 보겠습니다. 이를 위해 2개의 입력과 하나의 드롭다운이 있는 폼을 사용합니다. 다음 코드는 문제 없으나 계산기가 예상한대로 작동하지 않습니다.

```
<html>
<head>
  <title>Calculator</title>
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"  rel="\
stylesheet">
</head>
<body>
  <div class="container">
    <h1>Type 2 numbers and choose operation.</h1>
    <form class="form-inline">
        <!-- 입력을 숫자로 파싱하기 위해
             특수 수식어 'number'를 사용합니다. -->
        <input v-model.number="a"  class="form-control">
        <select v-model="operator"  class="form-control">
           <option>+</option>
           <option>-</option>
           <option>*</option>
           <option>/</option>
        </select>
        <!-- 입력을 숫자로 파싱하기 위해
             특수 수식어 'number'를 사용합니다. -->
        <input v-model.number="b"  class="form-control">
        <button type="submit"  @click="calculate"
        class="btn btn-primary">
            Calculate
```

```
26          </button>
27      </form>
28      <h2>Result: {{a}}  {{operator}}  {{b}} = {{c}}</h2>
29      <pre>
30          {{ $data }}
31      </pre>
32  </div>
33  </body>
34  <script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.3.4/vue.js"></script>
35  <script type="text/javascript">
36    new Vue({
37      el: '.container',
38      data: {
39        a: 1,
40        b: 2,
41        c: null,
42        operator: "+",
43    },
44    methods:{
45        calculate: function(){
46          switch (this.operator) {
47            case "+":
48                this.c = this.a + this.b
49                break;
50            case "-":
51                this.c = this.a - this.b
52                break;
53            case "*":
54                this.c = this.a * this.b
55                break;
56            case "/":
57                this.c = this.a / this.b
58                break;
59        }
60    }
61  },
62  });
63  </script>
64  </html>
```

이 코드를 실행하려고 “계산” 버튼을 클릭하면 계산하는 대신 페이지를 다시 로드합니다.

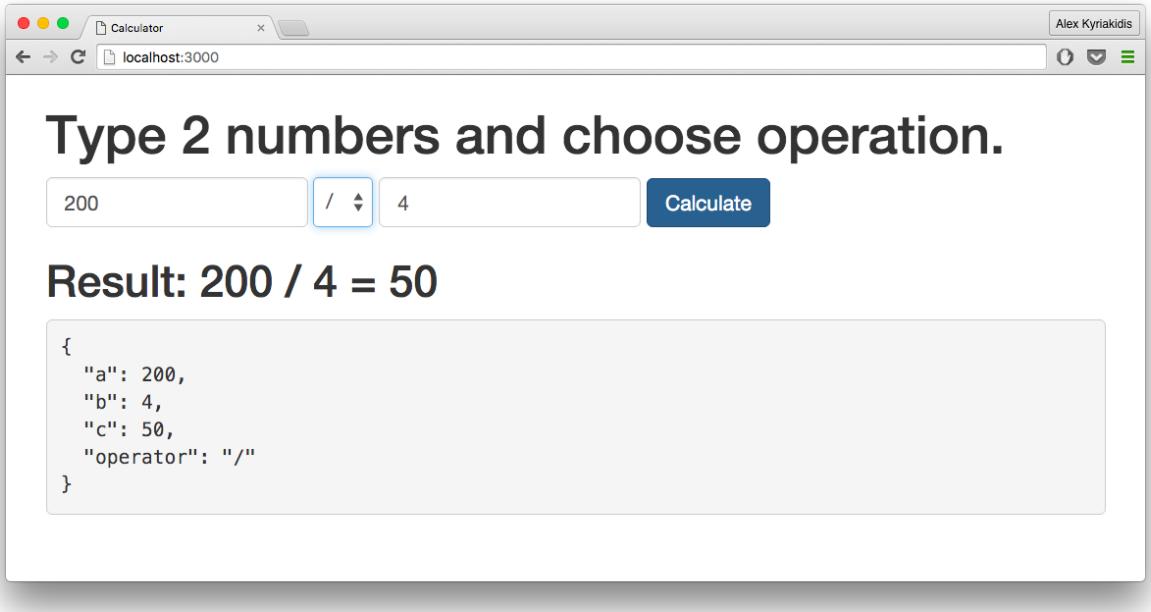
이는 당연합니다. “계산” 버튼을 클릭하면 폼 내용을 제출하기 때문에 페이지가 다시 로드됩니다.

제출을 막으려면 **onsubmit** 이벤트의 기본 동작을 막아야 합니다. 이벤트를 핸들링하는 메

소드에서 **event.preventDefault()**를 호출하는 것은 매우 보편적입니다. 이번 예제의 경우 이벤트 핸들링 메소드는 **calculate**입니다.

따라서 이렇게 될 것입니다.

```
calculate: function(event){
    event.preventDefault();
    switch (this.operator) {
        case "+":
            this.c = this.a + this.b
            break;
        case "-":
            this.c = this.a - this.b
            break;
        case "*":
            this.c = this.a * this.b
            break;
        case "/":
            this.c = this.a / this.b
            break;
    }
}
```



계산기를 위한 이벤트 수식어

메소드 안에서 더 쉽게 할 수 있지만, 아예 DOM 이벤트를 막는 코드가 없이 데이터를 처리하는 로직만 있다면 더 좋을 것 입니다.

Vue.js는 이벤트의 기본 동작을 방지하기 위해 **v-on**에 대해 네가지 이벤트 수식어를 제공합니다.

1. **.prevent**
2. **.stop**
3. **.capture**
4. **.self**

따라서 **.prevent**를 사용하여 아래의 제출 버튼을 변경하면

```
<button type="submit" @click="calculate">Calculate</button>
```

이렇게 됩니다.

```
<!-- submit 이벤트는 더이상 페이지를 새로고침하지 않습니다 -->
<button type="submit" @click.prevent="calculate">Calculate</button>
```

그리고 **calculate** 메소드에서 안전하게 **event.preventDefault()**를 제거할 수 있습니다.

노트

.capture와 **.self**는 거의 사용 되지 않으므로 더이상 신경쓰지 않아도 됩니다. 이벤트 순서 에 관심이 있으시면 이 튜토리얼[8]을 보세요.

키 수식어

입력 중 하나에 포커스를 주고 엔터 키를 누르면 calculate 메소드가 호출 되는 것을 알 수 있습니다. 버튼이 폼에 없거나 버튼이 전혀 없는 경우 키보드 이벤트를 대신 청취할 수 있습니다.

키보드 이벤트를 청취할 때 종종 키 코드를 확인해야 합니다. **엔터** 버튼의 키 코드는 13입니다. 그래서 다음과 같이 사용할 수 있습니다.

[8] http://www.quirksmode.org/js/events_order.html

```
<input v-model="a" @keyup.13="calculate">
```

모든 키코드를 외우기는 어렵습니다. 그래서 Vue는 가장 일반적으로 사용되는 키의 별칭을 제공합니다.

- enter
- tab
- delete
- esc
- space
- up
- down
- left
- right

따라서 이번 예제에서 엔터 키를 누를 때 `calculate` 메소드를 실행하려면 입력은 다음과 같이 바꿔야합니다.

```
<input v-model="a" @keyup.enter="calculate">
<input v-model="b" @keyup.enter="calculate">
```

팁

입력과 버튼 등 많은 폼의 종류가 있고 기본 제출 동작을 방지하려면 폼의 **submit** 이벤트를 수정해야 합니다. 예를 들어 이렇게 변경합니다. **<form @submit.prevent="calculate">**

마지막으로 계산기를 실행해 봅시다.

계산된 속성

Vue.js의 인라인 표현식은 매우 편리합니다. 하지만 조금 더 복잡한 로직이 필요해지면 반드시 계산된 속성을 사용해야합니다. 계산된 속성이란 다른 변수의 변화에 따라 값이 바뀌는 변수입니다. 계산된 속성은 함수처럼 작동하고 객체의 속성처럼 사용할 수 있습니다. 그러나 상당히 큰 차이점이 있습니다. 계산된 속성의 종속되는 속성이 변경될 때마다 계산된 속성의 값은 다시 계산됩니다.

Vue.js 에서는 **Vue** 인스턴스 내부의 **computed** 객체 내에서 계산된 속성을 정의 합니다.

```
<html>
<head>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" rel="st\
ylesheet">
<title>Hello Vue</title>
</head>
<body>
<div class="container">
    a={{ a }}, b={{ b }}
  <pre>
      {{ $data }}
  </pre>
</div>
</body>
<script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.3.4/vue.js"></script>
<script type="text/javascript">
new Vue({
    el: '.container',
    data: {
      a: 1,
    },
    computed: {
      // 계산된 getter
      b: function () {
        // **`this`**는 Vue 인스턴스 입니다
        return this.a + 1
      }
    }
});
</script>
</html>
```

두개의 변수를 만들었습니다. **a**는 1로 설정했고 **b**는 계산된 속성으로 함수의 반환된 결과로 설정됩니다. 이 예제에서 **b**는 2가 됩니다.

```
<html>
<head>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" rel="st\
ylesheet">
<title>Hello Vue</title>
</head>
<body>
<div class="container">
    a={{ a }}, b={{ b }}
  <input v-model="a">
```

```
11      <pre>
12          {{ $data }}
13      </pre>
14  </div>
15  </body>
16  <script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.3.4/vue.js"></script>
17  <script type="text/javascript">
18  new Vue({
19      el: '.container',
20      data: {
21        a: 1,
22      },
23      computed: {
24        // 계산된 getter
25        b: function () {
26          // **`this`**는 Vue 인스턴스 입니다
27          return this.a + 1
28        }
29      }
30  });
31  </script>
32  </html>
```

위의 예제는 이전 예제와 동일하지만 한가지 차이점이 있습니다. 변수 **a**가 사용자 입력에 바인딩 되어있습니다. 원하는 결과는 바인딩 된 속성의 값을 변경하고 **b**를 즉시 수정하는 것 입니다. 그러나 우리가 기대했던대로 작동하지 않습니다.

a의 값을 5로 입력했다면, 아마 **b**가 6이 될거라 예상했을 것 입니다. 하지만 **b**는 51 이 됩니다.

왜 이런일이 일어났을까요? 이미 아는바와 같이, **b**는 **a**의 값을 문자열로 인식합니다. 그래서 숫자 1을 마지막 위치에 추가합니다.

한가지 방법은 **parseFloat()** 함수를 이용하여 문자열을 부동 소수점 숫자로 변경하여 사용하는 것 입니다.

```
new Vue({
    el: '.container',
    data: {
      a: 1,
    },
    computed: {
      b: function () {
        return parseFloat(this.a) + 1
      }
    }
});
```

또 다른 방법으로는 **<input type=" number ">**을 사용하는 것 입니다. 항상 숫자를 사용하게 됩니다.

그러나 이보다 더 좋은 방법을 Vue.js가 제공하고 있습니다. 무엇을 입력하든 숫자로 만들어 줄 수 있습니다. 특별한 수식어인 **.number**를 붙여주면 됩니다.

```
<body>
<div class=" container ">
    a={{ a }}, b={{ b }}
    <input v-model.number=" a ">
    <pre>
        {{ $data }}
    </pre>
</div>
</body>
```

별 다른 노력이 필요없이 **number** 수식어를 이용하면 원하는 결과를 얻을 수 있습니다.

계산된 속성의 더 큰 그림을 보여주기 위해 이를 이용해 계산기를 만들어 보겠습니다. 이미 계산기를 만들어 보았지만 이번에는 메소드 대신 계산된 속성을 사용하겠습니다.

간단한 예제로 시작하겠습니다. 계산된 속성 **c**에는 **a**와 **b**의 합을 저장합니다.

```
1  <html>
2  <head>
3      <link href=" https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css "  rel\
4  = " stylesheet " >
5      <title>Hello Vue</title>
6  </head>
7  <body>
8      <div class=" container " >
9          <h1>Enter 2 numbers to calculate their sum.</h1>
10         <form class=" form-inline " >
11             <input v-model.number=" a "  class=" form-control " >
12             +
13             <input v-model.number=" b "  class=" form-control " >
14         </form>
15         <h2>Result: {{a}} + {{b}} = {{c}}</h2>
16         <pre> {{ $data }} </pre>
17     </div>
18 </body>
19 <script src=" https://cdnjs.cloudflare.com/ajax/libs/vue/2.3.4/vue.js " ></script>
20 <script type=" text/javascript " >
21 new Vue({
22   el:  ' .container ' ,
23   data: {
```

```
    a: 1,
    b: 2
  },
  computed: {
    c: function () {
      return this.a + this.b
    }
  }
});
</script>
</html>
```

기초적인 코드를 준비했습니다. 이 시점에서 사용자가 2개의 숫자를 입력하면 그 합을 얻을 수 있습니다. 목표는 사칙연산을 할 수 있는 계산기 입니다. 계속 만들어 보겠습니다.

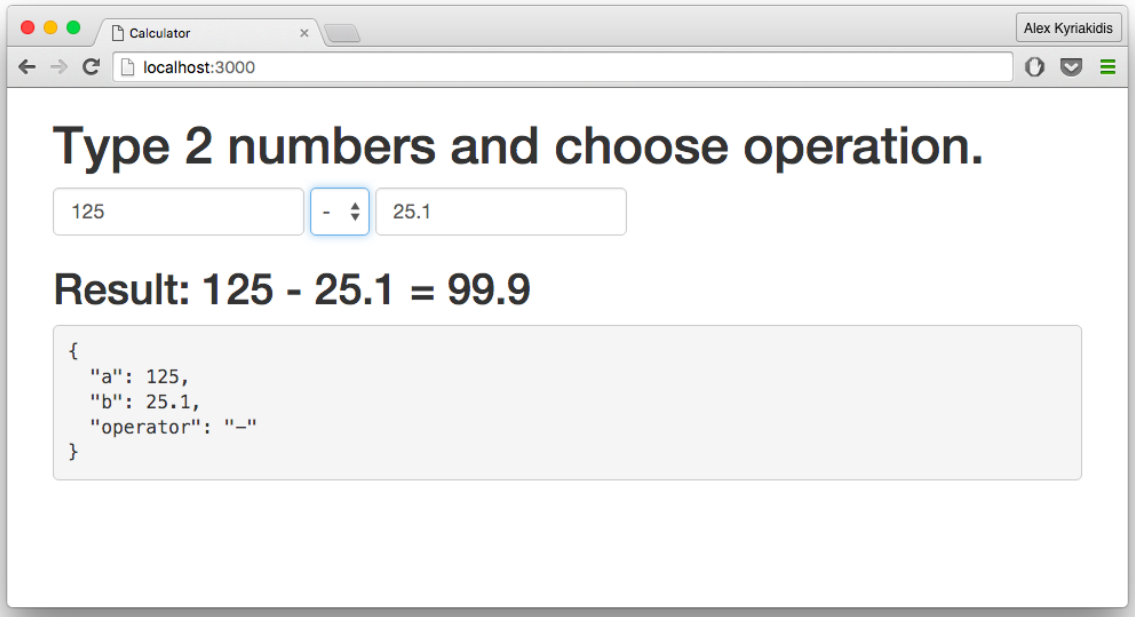
HTML 코드는 이전 장에서 작성한 계산기와 같으므로 (지금은 버튼이 필요 없습니다.) 자바스크립트 블럭만 보시면 됩니다.

```
new Vue({
  el: '.container',
  data: {
    a: 1,
    b: 2,
    operator: "+",
  },
  computed: {
    c: function () {
      switch (this.operator) {
        case "+":
        return this.a + this.b
        break;
        case "-":
        return this.a - this.b
        break;
        case "*":
        return this.a * this.b
        break;
        case "/":
        return this.a / this.b
        break;
      }
    }
  },
});
```

계산기를 사용할 준비가 끝났습니다. 할 일은 **calculate** 메소드 안에 있는 것이 무엇이든 계산된 속성 **c**로 옮기는 것 밖에 없습니다! **a**와 **b**가 어떤 값으로 변경되든 실시간으로 결과가 바뀝니다. 버튼이나 이벤트 그 어떤 것도 필요없습니다. **이 얼마나 대단한 일인가요?**

노트

일반적인 접근 방법은 나누기에 대한 오류를 피하기 위해 **if** 문을 사용하는 것 입니다. 그러나 이미 이러한 종류의 결함에 대한 예측을 하고 있습니다. 사용자가 1/0을 입력하면 결과는 자동으로 무한대가 됩니다! 사용자가 텍스트를 입력하면 표시된 결과는 "숫자가 아닙니다."



계산된 속성을 이용한 계산기 만들기

예제 코드

이 장의 예제 코드는 GitHub[9]에 있습니다.

[9] https://github.com/hootlex/the-majesty-of-vuejs-2/tree/master/codes/chapter5

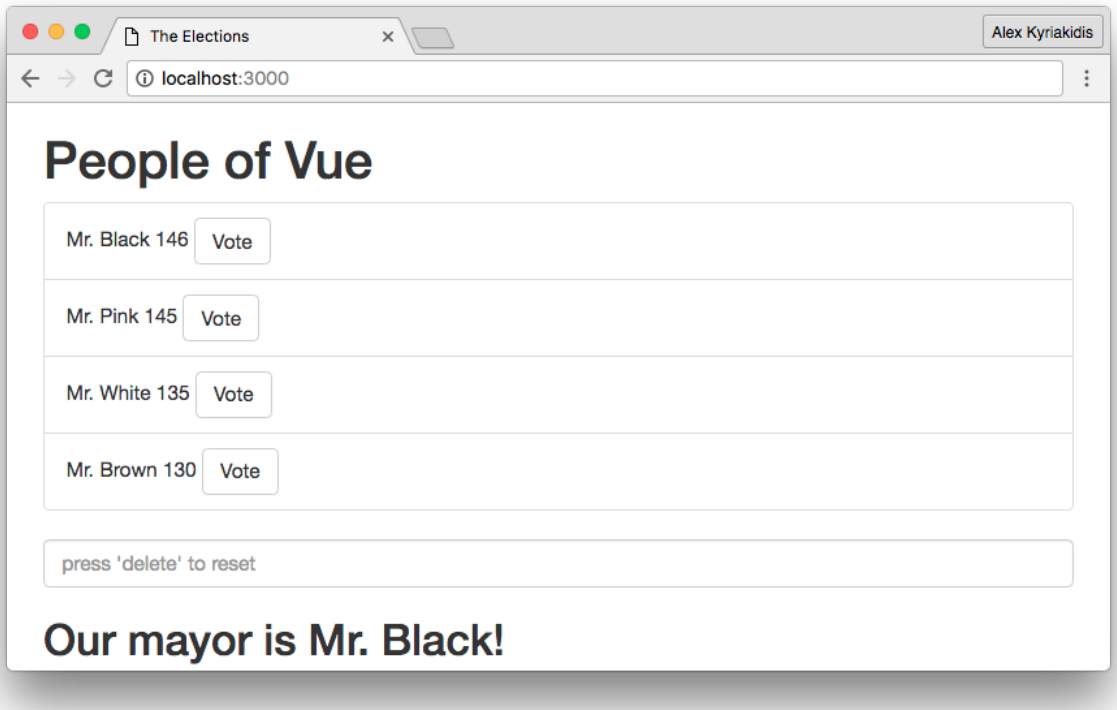
혼자 해보기

이제 Vue의 이벤트 처리, 메소드, 계산된 속성 등에 대한 기본적인 이해를 하게 되었으므로 조금 더 도전해보겠습니다. “시장” 후보들의 배열을 만드는 것 부터 시작하겠습니다. 각 후보자는 “이름(name)”과 “투표수(votes)”를 가지고 있습니다. 버튼을 사용하여 각 후보자의 득표수를 추가합니다. 계산된 속성을 사용하여 누가 현재 “시장”이 될 수 있는지 결정하고 이름을 표시하세요.

마지막으로 사용자 입력을 추가하세요. 사용자 입력이 포커스 되고 ‘delete’ 키가 눌리면 선거는 처음부터 시작됩니다. 이 말은 모든 투표수를 0으로 만드는 것 입니다.

힌트

자바스크립트의 내장 함수인 `sort()`와 `map()` 메소드는 매우 유용할 것입니다. 그리고 키 수식어를 사용하면 완성하는데 도움이 될 것입니다.



결과 예시

해결방법의 예

이 문제의 잠재적인 해결 방법은 예제[10]를 참조하세요.

[10] https://github.com/hootlex/the-majesty-of-vuejs-2/blob/master/homework/chapter5.html