Величие Vue.js 2

Алекс Кириакидис

Костас Маниатис

Величие Vue.js 2

Kostas Maniatis, Alex Kyriakidis, Alexey Pyltsyn и Roman Sadzhenytsia

Эта книга предназначена для продажи на http://leanpub.com/vuejs2-russian

Эта версия была опубликована на 2018-09-20

Это книга с Leanpub book. Leanpub позволяет авторам и издателям участвовать в так называемом Lean Publishing - процессе, при котором электронная книга становится доступна читателям ещё до её завершения. Это помогает собрать отзывы и пожелания для скорейшего улучшения книги. Мы призываем авторов публиковать свои работы как можно раньше и чаще, постепенно улучшая качество и объём материала. Тем более, что с нашими удобными инструментами этот процесс превращается в удовольствие.

Оглавление

Введение

Про Vue.js

Обзор Vue.js

Vue (произносится как /vju:/, примерно как view) — это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от монолитных фреймворков, Vue разработан с нуля для постепенного внедрения. Ядро библиотеки ориентировано **только на слой представления**, и его легко взять и интегрировать с другими библиотеками или существующими проектами. С другой стороны, Vue также отлично подходит для создания одностраничных приложений (Single-Page Applications), когда он используется в сочетании с современными инструментами и доступными библиотеками[1].

Если вы опытный фронтенд-разработчик и хотите узнать, чем Vue.js отличается от других библиотек/фреймворков, ознакомьтесь с главой Сравнение с другими фреймворками.

Если вам интересно больше узнать о ядре Vue.js, посмотрите на официальное руководство Vue.js[2].

Что говорят люди о Vue.js

“Vue.js — это то, что заставило меня полюбить JavaScript. Его очень легко и приятно использовать. Он обладает большой экосистемой плагинов и инструментов, позволяющие расширить его основную функциональность. Вы можете быстро включить его в любой проект, маленький или большой, написать несколько строк кода, и вы уже готовы к работе. Vue.js быстрый, лёгкий и это будущее фронтенд-разработки!”

— Алекс Кириакидис (Alex Kyriakidis)

“Когда я начал улучшать свой уровень JavaScript, я был взволнован, посмотрев кучу возможностей, но когда мой друг предложил изучить Vue.js и я последовал его совету, и дела пошли очень хорошо. Во время чтения и просмотра учебных материалов, я всё время думал о том, что я делал до сих пор и насколько это могло быть легче, если бы я ранее потратил время на изучение Vue. Моё мнение таково, что если вы хотите работать быстро, приятно и легко, то Vue — это JS-фреймворк, который вам нужен.”

— Костас Маниатис (Kostas Maniatis)

---

[1] https://github.com/vuejs/awesome-vue#libraries--plugins
[2] https://ru.vuejs.org/v2/guide/

“Запомните мои слова: Vue.js сильно вырастит в популярности в 2016 году. Он того стоит.”

— Джеффри Уэй (Jeffrey Way)

“Vue — это тот фреймворк, который я всегда хотел найти в JavaScript. Этот фреймворк, который масштабируется вместе с вами как разработчиком. Вы можете использовать его на одной странице, или создать продвинутое одностраничное приложение, используя Vuex и Vue Router. Это действительно самый доведённый до совершенства JavaScript-фреймворк, который я когда-либо видел.”

— Тейлор Отуэлл (Taylor Otwell)

“Vue.js — это первый фреймворк, который я нашёл, подходящий как для обычного бекенд-приложения, так и для создания полномасштабного SPA. Вне зависимости от того, нужен мне небольшой виджет на какой-нибудь странице или я создаю сложное JavaScript-приложение, Vue никогда не ощущается как что-то недостаточное или слишком ненужное для этого.”

— Адам Ватан (Adam Wathan)

“Vue.js удалось стать фреймворком, который простой в использовании и лёгкий для понимания. Это глоток свежего воздуха в мире, где люди борются за то, чтобы узнать, кто из них сделает самый сложный фреймворк.”

— Эрик Барнс (Eric Barnes)

“Причина, по которой мне нравится Vue.js, — это то, что я одновременно и дизайнер, и разработчик. Я посмотрел на React, Angular и на несколько других, но кривая обучения и терминология меня всегда отталкивали. Vue.js — это первый JS-фреймворк, который я понимаю. Он не только лёгок на подъём для такого менее уверенного JS-разработчика, как я, но я также заметил, что опытные разработчики на Angular и React обращают на него внимание и испытывают симпатию к Vue.js. Это довольно необычно для мира JS, и именно поэтому я начал митап London Vue.js.”

— Джек Бархам (Jack Barham)

“В поисках альтернативы Angular и jQuery, я вспомнил, что однажды наткнулся на Vue.js. В том время я совсем немного изучил его, но, изучая его подробнее, я нашёл идеальным его для удовлетворения моих основных потребностей: повышения реактивности моих представлений в приложениях на ASP.NET MVC. Поэтому я стал использовать Vue.js вместо Angular и jQuery. Я научился любить и уважать Vue.js, потому что его было легко изучать и просто использовать. Между тем я пришёл к выводу, что мне больше не нужен jQuery или Angular, потому что есть Vue.js, и этот потрясающий фреймворк может использоваться как замена им обоим: как для улучшения представлений или как инструмент для разработки полноценных SPA и даже PWA.”

— ***Майкл Сигер (Michael Seeger)***

Добро пожаловать

О книге

Эта книга проведёт вас по пути стремительно распространяющегося JavaScript-фреймворка под названием Vue.js!

Некоторое время назад мы начали новый проект, используя Laravel и Vue.js. После тщательного ознакомления с руководством Vue.js и несколькими учебными материалами обнаружилась нехватка ресурсов по Vue.js во всём вебе. Во время разработки нашего проекта мы получили достаточное количество опыта, поэтому появилась идея написать эту книгу, чтобы поделиться приобретёнными знаниями с миром. Теперь, когда вышел Vue.js 2, мы решили, что пришло время обновить книгу, опубликовав вторую версию, в которой примеры и связанный с ним текст переписаны.

Эта книга написана в неформальном, интуитивно понятном и удобном формате, где все примеры достаточно подробны для обеспечения адекватного руководства для каждого.

Мы начнём с самых основ, и через многие примеры рассмотрим наиболее важные особенности Vue.js. К концу книги вы сможете создавать быстрые фронтенд-приложения и повысить производительность существующих проектов с помощью интеграции Vue.js 2.

Для кого эта книга

Каждый, кто потратил время на изучение современной веб-разработки, работал с Bootstrap, JavaScript и со многими его фреймворками. Эта книга предназначена для всех, кто интересуется изучением лёгкого и простого фреймворка на JavaScript. Никаких особых знаний не требуется, хотя знание HTML и JavaScript было бы кстати. Если вы не знаете, какая разница между строкой и объектом, вам, возможно, стоит сначала изучить основы.

Эта книга полезна как для разработчиков, которые не знают Vue.js, так и тем, кто уже использует Vue.js, но хочет расширить свои знания. Кроме того, книга пригодится тем, кто хочет перейти на Vue.js 2.

Как с нами связаться

Если вы хотите связаться с нами по поводу книги, прислать отзыв или задать вопрос по другой теме, не стесняйтесь обращаться к нам.

Имя	Электронная почта	Твиттер
The Majesty of Vue.js	hello@tmvuejs.com	@tmvuejs
Алекс Кириакидис (Alex Kyriakidis)	alex@tmvuejs.com	@hootlex
Костас Маниатис (Kostas Maniatis)	kostas@tmvuejs.com	@kostaskafcas

Домашние задания

Лучший способ учиться веб-разработке — писать код, поэтому мы подготовили по одному упражнению в конце большинства глав, чтобы вы могли решить их и действительно проверить, что вы изучили. Мы настоятельно рекомендуем как можно больше решить их для лучшего понимания Vue.js. Не бойтесь проверить свои идеи, всё начинается с малого. Возможно, несколько разных примеров или способов дадут вам правильное решение. Однако мы не безжалостны: будут даны подсказки и возможные решения!

Вы можете начать ваше путешествие!

Код примеров

Большинство примеров кода, используемых в книге, можно найти на GitHub. Вы можете посмотреть код здесь[3] (ветка `ru`). Или, если вам так удобнее, скачать его в файле формата `.zip` отсюда[4]. Это избавит вас от копирования и вставки кода из книги, что, вероятно, было бы утомительно.

Опечатки

Несмотря на то, что были приняты все меры для обеспечения точности нашей книги, ошибки всё же случаются. Если вы найдёте одну из них, мы будем благодарны, если вы сообщите нам о ней. Таким образом, вы можете уберечь других читателей от разочарования и заодно поможете нам улучшить последующие версии книги. Если вы обнаружите какие-либо ошибки, пожалуйста, создайте ишью в нашем репозитории на GitHub[5].

Принятые обозначения

В книге используются следующие условные обозначения.

Блок кода будет выглядеть следующим образом:

JavaScript

[3] https://github.com/hootlex/the-majesty-of-vuejs-2/tree/ru
[4] https://github.com/hootlex/the-majesty-of-vuejs-2/archive/ru.zip
[5] https://github.com/hootlex/the-majesty-of-vuejs-2

```
function (x, y) {
    // это комментарий
}
```

Слова с кодом в тексте и данные отображаются так: “Используйте `.container` для адаптивного контейнера с фиксированной шириной.”

Новые термины и **важные слова** выделены жирным шрифтом.

Советы, примечания и предупреждения отображаются подобным образом:

Это предупреждение

Этот элемент указывает на предупреждение или предостережение.

Это совет

Этот элемент означает совет, подсказка или предложение.

Это информационный блок

Некоторая специальная информация.

Это примечание

Примечание по теме.

Это подсказка

Подсказка по теме.

Это команда в терминале

Команды, выполняемые в терминале.

Это текст сравнения

Небольшой текст для сравнения по теме.

Это ссылка на GitHub.

Ссылки ведут на репозиторий книги, где вы можете найти примеры кода и возможные задания к каждой определённой главы.

I Основы Vue.js

1. Интерактивность

В этой главе мы собираемся создавать и расширять предыдущие примеры, изучать что-то новое, касающееся методов, обработки событий и вычисляемых свойств. Мы разработаем несколько примеров, используя разные подходы. Пришло время посмотреть, как мы можем реализовать интерактивность Vue, чтобы создать небольшое приложение, такое как калькулятор, работающее красиво и просто.

1.1 Обработка событий

События `HTML` — это то, что происходит с DOM-элементами. Когда Vue.js используется на страницах `HTML`, он может **реагировать** на эти события.

События могут представлять всё, от базовых пользовательских взаимодействий до того, что происходит при формировании отрисовки.

Вот некоторые примеры события `HTML`:

- Веб-страница полностью загрузилась
- Поле ввода было изменено
- Кнопка была нажата
- Форма была отправлена

Смысл обработки событий — это то, что вы можете сделать, когда происходит событие.

Во Vue.js для **обработки** DOM-событий вы можете использовать директиву `v-on`.

Директива `v-on` прикрепляет обработчик события к элементу. Тип события обозначается аргументом, например, `v-on:keyup` обрабатывает событие `keyup`.

На заметку

Событие `keyup` происходит, когда пользователь отпускает клавишу. Полный список событий `HTML` вы можете найти здесь[1].

1.1.1 Встроенная обработка событий

Достаточно слов, давайте перейдём к делу и посмотрим на обработку событий в действии. Ниже находится кнопка «Проголосовать!», которая увеличивает количество голосов при каждом нажатии.

[1] https://www.w3schools.com/tags/ref_eventattributes.asp

```
<html>
<head>
  <link
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
    rel="stylesheet"
  >
  <title>Проголосовать</title>
</head>
<body>
  <div class="container">
    <button type="button" v-on:click="upvotes++">
      Проголосовать! {{upvotes}}
    </button>
  </div>
</body>
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.5.1\
7/vue.js"></script>
<script type="text/javascript">
new Vue({
  el: '.container',
  data: {
    upvotes: 0
  }
})
</script>
</html>
```

Счётчик голосов

Внутри наших данных есть свойство **upvotes**. В этом случае мы привязываем обработчик событий нажатия (**click**) с JavaScript-выражением, которое находится рядом с ним. Внутри кавычек всякий раз, когда нажимается кнопка; мы простом увеличиваем счётчик голосов на один, используя оператор инкремента (**upvotes++**).

1.1.2 Обработка событий с использованием методов

Теперь мы будем то же самое, что и в предыдущем примере, но с одним отличием — использовать метод. Метод во Vue.js — это блок кода, предназначенный для выполнения конкретной задачи. Для выполнения метода, вы должны определить и вызвать его.

```
<html>
<head>
  <link
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
    rel="stylesheet"
  >
<title>Проголосовать</title>
</head>
<body>
  <div class="container">
    <button type="button" v-on:click="upvote">
      Проголосовать! {{upvotes}}
    </button>
  </div>
</body>
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.5.1\
7/vue.js"></script>
<script type="text/javascript">
new Vue({
  el: '.container',
  data: {
    upvotes: 0
  },
  // определить методы в объекте **`methods`**
  methods: {
    upvote: function () {
      // **`this`** внутри методов указывает на Vue-экземпляр
      this.upvotes++;
    }
  }
})
</script>
</html>
```

Мы привязываем обработчик события нажатия к методу **upvote**. Это работает так же, как и раньше, но выглядит более чистым и понятным при чтении кода.

Предупреждение

Обработчики событий ограничиваются выполнением **только одного выражения**.

1.1.3 Сокращение для `v-on`

Когда вы всё время используете `v-on` в проекте, то обнаружите что ваша HTML-разметка быстро станет грязной. К счастью, есть сокращение для `v-on` — символ `@`. Символ `@` заменяет весь `v-on:` и при его использовании код выглядит *намного чище*. Использование сокращения абсолютно необязательно.

С использованием `@` кнопка из нашего предыдущего примера будет выглядеть так, сравните сами:

Обработка события 'click', используя `v-on`:

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

Обработка события 'click', используя сокращение `@`

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

1.2 Модификаторы событий

Теперь мы перейдём к созданию приложения калькулятора. Для этого мы будем использовать форму с двумя полями ввода и одним выпадающим списком для выбора желаемой операции.

Несмотря на то, что следующий код выглядит прекрасным, наш калькулятор работает не так, как ожидалось.

```
<html>
<head>
  <title>Калькулятор</title>
  <link
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
    rel="stylesheet"
  >
</head>
<body>
  <div class="container">
    <h1>Введите 2 числа и выберите операцию.</h1>
    <form class="form-inline">
      <!-- Обратите внимание на используемый модификатор 'number'
      который анализирует входные данные в качестве чисел. -->
      <input v-model.number="a" class="form-control">
      <select v-model="operator" class="form-control">
        <option>+</option>
        <option>-</option>
        <option>*</option>
        <option>/</option>
      </select>
      <!-- Обратите внимание на используемый модификатор 'number'
      который анализирует входные данные в качестве чисел. -->
      <input v-model.number="b" class="form-control">
      <button
        type="submit"
        @click="calculate"
        class="btn btn-primary"
      >
        Вычислить
      </button>
   </form>
   <h2>Результат: {{a}}  {{operator}}  {{b}} = {{c}}</h2>
   <pre>{{ $data }}</pre>
</div>
</body>
<script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.5.17/vue.js"></script>
<script type="text/javascript">
  new Vue({
    el: '.container',
    data: {
      a: 1,
      b: 2,
```

```
      c: null,
      operator: '+',
  },
  methods: {
    calculate: function () {
      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;
      }
    }
},
});
</script>
</html>
```

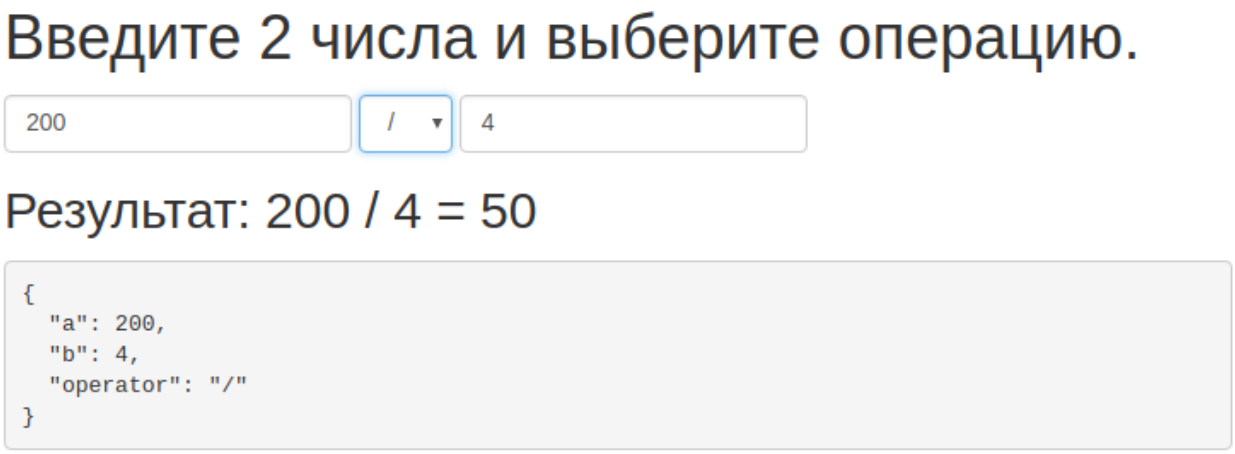
Если вы запустите данный код самостоятельно, то обнаружите, что при нажатии на кнопку «Вычислить», вместо вычисления произойдёт перезагрузка страницы.

Причина такого поведения ясна, потому что когда вы нажимаете на кнопку «Вычислить», то в фоновом режиме отправляете форму, и, таким образом, страница перезагружается.

Для предотвращения отправки формы, мы должны отменить действие по умолчанию для события **onsubmit**. В большинстве случаев вам нужно вызвать **event.preventDefault()** внутри нашего метода обработки. В нашем случае метод обработки событий называется **calculate**.

Итак, наш метод примет следующий вид:

```
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;
  }
}
```



Использование модификаторов событий для создания калькулятора

Хотя мы можем сделать это легко внутри методов, было бы лучше, если эти методы не имели дело с деталями событий DOM.

Vue.js предоставляет четыре модификатора `v-on` для предотвращения поведение по умолчанию события:

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

Итак, используя `.prevent`, наша кнопка отправки изменится с:

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

на:

```
<!-- событие отправки больше не будет перезагружать страницу -->
<button type="submit" @click.prevent="calculate">Вычислить</button>
```

Теперь мы можем безопасно удалить **`event.preventDefault()`** из метода **`calculate`**.

Примечание

`.capture` и **`.self`** редко используются, поэтому мы больше не будем более подробно касаться этой темы в дальнейшем. Если вам интересно больше узнать о *порядке выполнения событий*, посмотрите на эту обучающую статью[2].

1.3 Модификаторы клавиш

При фокусе на одном из полей ввода и нажатии на кнопку ввода (enter), вы заметите, что вызывается метод вычисления. Если кнопка была не внутри формы или вообще не было подобной кнопки, вы могли бы обработать событие клавиатуры вместо этого.

При обработке событий клавиатуры нам часто потребуется проверять коды клавиш. Код кнопки **`Enter`** — 13. Таким образом, мы можем поступить следующим образом для вызова метода вычисления при нажатии кнопки ввода:

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

Учитывая, что все коды клавиш сложно запоминать, Vue предоставляет псевдонимы для наиболее используемых клавиш:

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

Итак, для выполнения метода `calculate` при нажатии на кнопку ввода в нашем примере, поля ввода будут выглядеть так:

[2] https://www.quirksmode.org/js/events_order.html

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

Подсказка

Когда у вас есть форма с большим количеством полей вводов, кнопок и т.д. и вам нужно предотвратить событие отправки формы по умолчанию, вы можете применить модификатор **`submit`** к событию формы.

Например: **`<form @submit.prevent="calculate">`**

Наконец, наш калькулятор готов к работе.

1.4 Вычисляемые свойства

Встроенные выражения Vue.js очень удобны, но для более сложной логики, вы должны использовать вычисляемые свойства. Практически, вычисляемые свойства — это переменные, значение которых зависит от других факторов.

Вычисляемые свойства работают как функции, которые можно использовать в качестве свойств. Но есть существенная разница. Каждый раз, когда изменяется зависимость вычисляемого свойства, значение вычисляемого свойства переоценивается.

Во Vue.js вы определяете вычисляемые свойства внутри объекта **`computed`** экземпляра **`Vue`**.

```
<html>
<head>
  <link
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
    rel="stylesheet"
  >
<title>Привет, 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/2.5.17/vue.js"></script>
<script type="text/javascript">
new Vue({
  el: '.container',
```

```
  data: {
    a: 1,
  },
  computed: {
    // вычисляемый геттер
    b: function () {
      // **`this`** указывает на Vue-экземпляр
      return this.a + 1
    }
  }
});
</script>
</html>
```

Мы установили две переменные: первая, **a**, установлена в 1, а вторая, **b**, будет установлена возвращаемым значением функции внутри объекта вычисляемых свойств. В этом примере значение **b** будет установлено в 2.

```
<html>
<head>
  <link
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
    rel="stylesheet"
  >
  <title>Привет, Vue</title>
</head>
<body>
<div class="container">
  a={{ a }}, b={{ b }}
  <input v-model="a">
  <pre>{{ $data }}</pre>
</div>
</body>
<script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.5.17/vue.js"></script>
<script type="text/javascript">
new Vue({
  el: '.container',
  data: {
    a: 1,
  },
  computed: {
    // вычисляемый геттер
    b: function () {
```

```
26        // **`this`** указывает на Vue-экземпляр
27        return this.a + 1
28      }
29    }
30  });
31  </script>
32  </html>
```

Вышеприведённый пример такой же, как предыдущий, но с одним отличием. Поле ввода связано с переменной **a**. Желательным результатом было бы изменить значение привязанного атрибута и немедленно обновить результат **b**. Но обратите внимание, что он не работает, как мы ожидали.

Если вы запустите данный код и установите переменную **a** на **5**, вы ожидаете, что **b** будет равно **6**. Конечно, это не так, **b** устанавливается значением **51**.

Почему это происходит? Ну, как вы уже могли догадаться, **b** принимает заданное значение из поле ввода **a** в виде строки и добавляет к нему **1** в конец.

Одним из возможных решений — это использовать функцию **parseFloat()**, **которая обрабатывает строку и возвращает число с плавающей точкой**.

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

Другой вариант, который приходит на ум, - использовать **<input type="number">**, как раз для полей ввода, предназначенных содержать числовое значение.

Но есть более аккуратный способ. С Vue.js, когда вы хотите, чтобы пользовательское поле ввода автоматически содержало число, вы можете добавить специальный модификатор **.number**.

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

Модификатор **number** даст нам желаемый результат без каких-либо дополнительных усилий.

Для демонстрации вычисляемых свойств мы будем использовать их в калькуляторе, показанном ранее, но на этот раз используя вычисляемые свойства вместо методов.

Давайте начнём с простого примера, где вычисляемое свойство **c** содержит сумму **a** и **b**.

```
<html>
<head>
  <link
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
    rel="stylesheet"
  >
  <title>Привет, Vue</title>
</head>
<body>
    <div class="container">
      <h1>Введите 2 числа для вычисления их суммы.</h1>
      <form class="form-inline">
        <input v-model.number="a" class="form-control">
        +
        <input v-model.number="b" class="form-control">
      </form>
      <h2>Результат: {{a}} + {{b}} = {{c}}</h2>
      <pre> {{ $data }} </pre>
    </div>
</body>
<script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.5.17/vue.js"></script>
<script type="text/javascript">
new Vue({
  el: '.container',
  data: {
    a: 1,
    b: 2
```

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

Начальный код готов, и в этот момент пользователь может ввести 2 числа и получить их сумму. Цель калькулятора — выполнение четырёх основных операций, поэтому давайте приступим к работе!

Поскольку код `HTML` будет таким же, как в калькуляторе, который мы создавали в предыдущем разделе этой главы (за исключением того, что сейчас нам не потребуется кнопка), я собираюсь показать только блок кода с JavaScript.

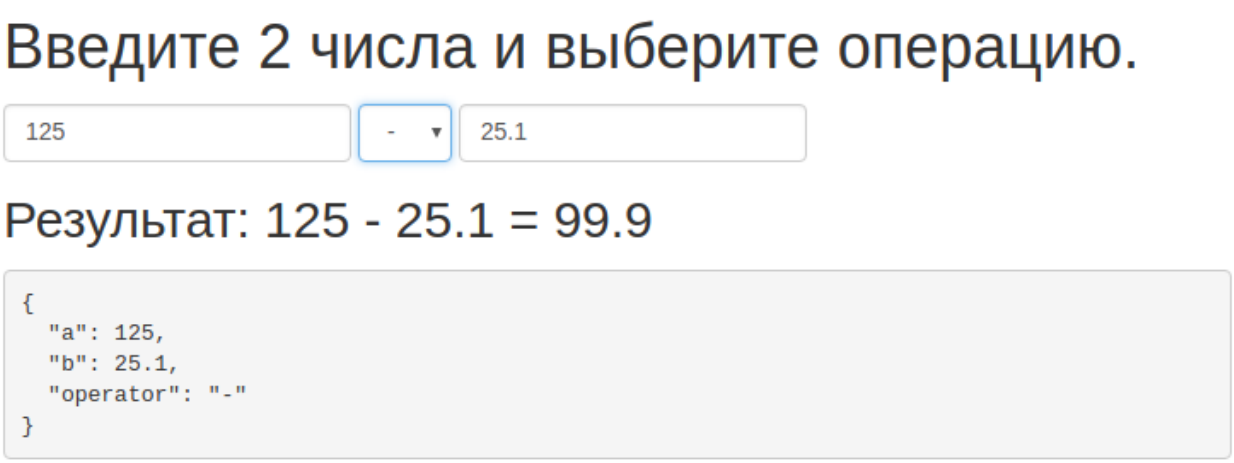
```
new Vue({
  el: '.container',
  data: {
    a: 1,
    b: 2,
    operator: '+',
  },
  computed: {
    c: function () {
      switch (this.operator) {
        case '+':
          return this.a + this.b
        case '-':
          return this.a - this.b
        case '*':
          return this.a * this.b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

Калькулятор готов к использованию. Единственное, что мы должны сделать, - так это переместить всё, что было в методе **`calculate`**, в вычисляемое свойство **`c`**! Всякий раз, когда

вы изменяете `a` или `b`, результаты обновляются в режиме реального времени! Нам не нужны никакие кнопки, события или что-то ещё. **Разве это не удивительно??**

Примечание

Обратите внимание, что обычный подход состоял бы в том, чтобы иметь выражение `if` для предотвращения ошибок деления. Но на случай такой ошибки у нас есть сценарий использования. Если пользователь вводит 1/0, результат автоматически становится бесконечным! Если пользователь вводит текст, в качестве результата отображается “не число”.



Калькулятор, созданный с использованием вычисляемых свойств

Примеры кода

Вы можете найти примеры кода этой главы на GitHub[3].

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

1.5 Домашняя работа

Теперь, когда у вас есть базовое понимание обработки событий, методов, вычисляемых свойств и т.д. во Vue, вы должны попробовать что-то более сложное. Начните с создания массива кандидатов в мэры (`mayor`). У каждого кандидата есть имя (`name`) и количество голосов (`votes`). Используйте кнопку для увеличения количества голосов для каждого кандидата. Используйте вычисляемое свойство для определения, кто сейчас текущий мэр (на основе голосов) и его имя.

Наконец, добавьте поле ввода. При получении фокуса на поле ввода и нажатой клавиши **`delete`** выборы начинаются с самого начала. Это означает, что все голоса обнуляются.

Подсказка

Методы JavaScript **`sort()`** и **`map()`** могут оказаться очень полезными и модификаторы ключей помогут вам в этом.

Пример вывода

Возможное решение

Вы можете найти потенциальное решение этого упражнения здесь[4].

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