The Majesty of Vue.js 2

Alex Kyriakidis

Kostas Maniatis

The Majesty of Vue.js 2 (Deutsch)

Alex Kyriakidis, Kostas Maniatis und Michael Seeger

Dieses Buch wird verkauft, unter http://leanpub.com/vuejs2-deutsch

Diese Version wurde veröffentlicht am 2017-08-08

Inhaltsverzeichnis

Über Vue.js

Vue.js Übersicht

Vue (ausgesprochen wie "view" aus dem Englischen) ist ein fortschrittliches Framework zur Erstellung von Benutzerschnittstellen. Im Gegensatz zu anderen monolithischen Frameworks ist Vue so konzipiert, dass es aufbauend verwendet werden kann. Die Kernbibliothek ist **auf die Darstellungsebene** fokussiert und ist sehr einfach, zusammen mit anderen Bibliotheken oder in existierenden Projekten, einzusetzen. Vue ist ebenso perfekt dazu geeignet, ausgeklügelte Single-Page-Applikationen in Kombination mit modernen Werkzeugen und Hilfsbibliotheken[1] zu implementieren.

Wenn Sie ein erfahrener Frontend-Entwickler sind und wissen wollen, wie man Vue.js mit anderen Bibliotheken/Frameworks vergleichen kann, dann schauen Sie sich das Kapitel Vergleich mit anderen Frameworks an.

Wenn Sie daran interessiert sind, mehr über den Kern von Vue.js zu erfahren, dann riskieren Sie einen Blick in das Vue.js Handbuch[2].

Was man über Vue.js sagt

"Vue.js brachte mich dazu, JavaScript zu lieben. Man kann es extrem einfach und mit Freude benutzen. Es hat ein großes Plugin-Ökosystem und Werkzeuge zur Erweiterung der Basis. Man kann es schnell in jedes Projekt - klein oder groß - einbinden, schreibt ein paar Zeilen Code und man ist fertig. Vue.js ist schnell, leichtgewichtig und es ist die Zukunft der Frontend-Entwicklung!"

—Alex Kyriakidis

"Als ich mit JavaScript anfing, war ich sehr enthusiastisch: Was es hier alles an Möglichkeiten und zu lernen gibt. Als mein Freund mir aber vorschlug, Vue.js zu lernen und ich diesem Rat folgte, nahmen die Dinge überhand. Während ich las und Video-Tutorials schaute, dachte ich an all das, was ich zuvor programmiert hatte und um wieviel einfacher es gewesen wäre, wenn ich Vue früher gelernt hätte. Meine Meinung ist, dass Vue das JS Framework ist, welches man braucht, wenn man seine Arbeit schnell, einfach und gut erledigen möchte."

—Kostas Maniatis

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

"Merken Sie sich meine Worte: Der Bekanntheitsgrad von Vue.js wird 2016 in den Himmel schießen. Es ist so gut."

—Jeffrey Way

"Vue bietet das, was ich immer in einem JavaScript Framework haben wollte: Es ist ein Framework, das mit Dir als Entwickler skaliert. Man kann es in eine Seite einbauen oder eine fortgeschrittene Single-Page-Applikation mit Vuex und dem Vue Router erstellen. Es ist wahrhaftig das vollkommenste JavaScript Framework, das ich je gesehen habe."

—Taylor Otwell

"Vue.js ist das erste Framework, das ich gefunden habe, welches sich in der Benutzung in einer servergerenderten Applikation genauso "natürlich" anfühlt, wie in einer großen SPA. Ob ich nun eine kleine Komponente auf einer einzelnen Seite benötige oder einen komplexen JavaScript Client erstelle, es fühlt sich niemals als "nicht genug" oder wie ein Overkill an."

—Adam Wathan

"Vue.js gelang es, ein Framework zu erstellen, das beides ist: Einfach zu benutzen und einfach zu verstehen. Es ist wie ein Atemzug frischer Luft in einer Welt, in der andere darum kämpfen, herauszufinden, wer das komplexeste Framework erstellt."

—Eric Barnes

"Der Grund, warum ich Vue.js mag, ist, weil ich eine Kombination aus Designer und Entwickler bin. Ich habe mir React, Angular und eine Menge Anderer angeschaut, aber Lernkurven und die Fachsprache haben mich immer aus dem Konzept gebracht. Vue.js ist das erste JS Framework, das ich verstehe. Nicht nur, dass es für diejenigen, die wenig Zuversicht in JS haben (wie mich selbst), sehr einfach zu erlernen ist, ich habe auch festgestellt, dass erfahrene Angular- und React-Entwickler auf Vue.js aufmerksam wurden, und es mögen. Das ist ziemlich beispiellos in der JS-Welt und der Grund, warum ich das London Vue.js Meetup gegründet habe."

—Jack Barham

"Auf der Suche nach einer Alterntive zu Angular und jQuery erinnerte ich mich, dass ich vor längerer Zeit auf Vue.js stieß, es damals aber nur am Rande wargenommen habe. Doch als ich es mir jetzt etwas näher angeschaut hatte, fand ich, dass es ideal für meine primären Zwecke war: Die Steierung der Reaktivität meiner ASP.NET MVC Views. Und so setzte ich Vue.js anstelle von Angular und jQuery ein. Ich lernte Vue.js schätzen und lieben, weil es so einfach zu erlernen und ebenso einfach einzusetzen ist. Mittlerweile bin ich zum Schluss gekommen, dass ich jQuery oder Angular garnicht mehr benötige, denn es gibt Vue.js und dieses geniale Framework kann für beides eingesetzt werden: zur Steigerung der Reaktivität von Views oder als Werkzeug zur Entwicklung ausgewachsener SPAs und sogar PWAs."

—Michael Seeger

The Majesty of Vue.js 2 (Deutsch)

Willkommen

Über dieses Buch

Dieses Buch ist ein Begleiter auf Ihrem Weg mit Vue.js, einem JavaScript Framework, das sich aktuell sehr schnell verbreitet.

Vor einiger Zeit haben wir ein neues Projekt auf Basis von Laravel und Vue.js begonnen. Nach gründlichem Lesen der Vue.js Dokumentation und einiger Tutorials, stellten wir fest, dass die Ressourcen zu Vue.js im Web fehlten. Während der Entwicklung unseres Projekts bauten wir viel Erfahrung auf, und so kamen wir auf die Idee, dieses Buch zu schreiben, um so unser erlangtes Wissen mit der Welt zu teilen. Nun da Vue.js in Version 2 veröffentlicht wurde, entschieden wir, dass es Zeit wird, unser Buch zu aktualisieren und eine zweite Version herauszubringen, in der alle Beispiele und deren zugehöriger Inhalt neu geschrieben wurden.

Dieses Buch ist in informeller, intuitiver und leicht nachzuvollziehender Weise geschrieben. Alle Beispiele sind angemessen detailliert, und dienen als adäquate Anleitung.

Wir beginnen mit den einfachen Grundlagen und beleuchten anhand zahlreicher Beispiele die bedeutungsvollen Charakteristika von Vue.js. Am Ende dieses Buches werden Sie schnelle Frontend-Applikationen erstellen oder die Performance Ihrer existierenden Projekte durch Einbinden von Vue.js 2 erhöhen können.

Für wen ist dieses Buch?

Jeder, der die Zeit dazu aufgebracht hat, moderne Webentwicklung zu lernen, hat Bootstrap, JavaScript und viele JavaScript Frameworks gesehen. Dieses Buch ist für diejenigen die daran interessiert sind, ein leichtgewichtiges und einfaches JavaScript Framework zu erlernen. Dazu wird kein umfassendes Wissen verlangt, obwohl es gut wäre, wenn man mit HTML und JavaScript vertraut ist. Wenn Sie nicht wissen, was der Unterschied zwischen einem String und einem Objekt ist, dann sollten Sie doch noch etwas tiefer in die Materie eintauchen.

Dieses Buch hilft Entwicklern, für die Vue.js neu ist, und auch denen, die Vue.js bereits benutzen und ihr Wissen erweitern und vertiefen wollen. Auch Entwicklern, die einen Umstieg nach Vue.js 2 planen, wird es eine Hilfe sein.

Erstes Herantasten

Wenn Sie uns in Bezug auf das Buch ansprechen, uns Feedback zukommen lassen oder uns auf andere Dinge aufmerksam machen wollen, scheuen Sie sich bitte nicht, uns zu kontaktieren.

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

Hausaufgaben

Der beste Weg, Code zu erlernen, ist Code zu schreiben! Also haben wir am Ende fast jedes Kapitels eine Aufgabe für Sie vorbereitet, mit der Sie überprüfen können was Sie gelernt haben. Wir legen ihnen sehr ans Herz, diese Aufgabe(n) zu lösen, um so Ihr Wissen über Vue.js nachhaltig aufzubauen. Die unterschiedlichen Beispiele können dazu beitragen, bei der Lösung auf den richtigen Weg zu gelangen. Selbstverständlich erhalten Sie auch Hinweise und eventuelle Lösungen von uns!

Sie können also mit Ihrer Reise beginnen!

Beispielcode

Sie finden viele der Codebeispiele, die in diesem Buch stehen, auf GitHub. Hier[3] können Sie im Code stöbern.

Wenn Sie es bevorzugen, sich den Code herunterzuladen, steht ihnen hier[4] eine `.zip`-Datei zur Verfügung.

Das spart Ihnen Code mit Copy & Paste aus dem Buch zu übertragen, was schätzungsweise nicht so gut wäre.

Errata

Auch wenn unsere Inhalte mit größter Sorgfalt auf Richtigkeit geprüft werden, kann es zu Fehlern kommen. Wenn Sie einen Fehler im Buch finden, wären wir dankbar, wenn Sie ihn uns melden. Damit können Sie unsere Leser vor Frust bewahren und uns dabei behilflich sein, die Folgeversionen dieses Buches zu verbessern. Wenn Sie also einen Fehler finden, dann senden Sie ihn bitte an unser GitHub Repository[5].

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

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

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

Konventionen

Die folgenden Konventionen für die Anmerkungen gelten für das ganze Buch.

Ein Codeblock wird wie folgt dargestellt:

JavaScript

```
function(x, y){
    // Das ist ein Kommentar
}
```

Codeworte im Text und Daten werden wie folgt dargestellt: “Benutzen Sie `.container` für einen responsiven Container mit fester Breite.”

Neue Ausdrücke und **wichtige Worte** werden fettgedruckt dargestellt.

Tipps, **Anmerkungen**, **und Warnungen** werden wie folgt dargestellt:

Dies ist eine Warnung

Dieses Element weist eine Warnung aus, oder mahnt zur Vorsicht.

Dies ist ein Tipp

Dieses Element zeigt einen Tipp oder einen Vorschlag an.

Dies ist eine Info-Box

Hier gibt es spezielle Informationen.

Dies ist eine Anmerkung

Eine Anmerkung zum Thema.

Dies ist ein Hinweis

Ein Hinweis zum Thema.

Dies ist eine Anweisung im Terminal

Anweisungen, die im Terminal ausgeführt werden.

Dies ist ein Text mit einem Vergleich

Ein kleiner Text, in dem ein Vergleich angestellt wird.

Dies ist ein Link zu GitHub.

Links führen zum Repository dieses Buches, wo Sie die Codebeispiele und mögliche Lösungen zu den Hausaufgaben jedes Kapitels finden.

Vue.js Fundamentals

Interaktivität

In diesem Kapitel erweitern wir die vorhergehenden Beispiele, um neues Wissen über ‘Methoden’, ‘Ereignisbehandlung’ (Event Handling) und ‘berechnete Properties’ (Computed Properties) zu erlangen. Wir entwickeln ein paar Beispiele und werden dabei unterschiedlich vorgehen. Es ist nun Zeit die Interaktivität von Vue dazu zu nutzen, um eine kleine Applikation, wie z. B. einen Taschenrechner, auf einfache Weise zu implementieren.

Ereignisbehandlung

`HTML`-Ereignisse (die sog. Events) “passieren” einem DOM-Element. Wird Vue.js in `HTML`-Seiten eingesetzt, kann auf diese Ereignisse **reagiert** werden.

Dabei können Ereignisse alles sein: von einer einfachen Interaktion eines Benutzers bis hin zu Ereignissen, die im auszugebenden Model auftreten.

Hier sind einige Beispiele für `HTML`-Ereignisse:

- Eine Seite wurde fertig geladen.
- Der Inhalt eines Eingabefeldes wurde geändert.
- Ein Button wurde angeklickt.
- Ein Formular wurde übertragen.

Sinn und Zweck der Ereignisbehandlung ist, dass man beim Eintreten eines Ereignisses etwas auslösen kann.

Die `v-on`-Direktive wird eingesetzt, um in Vue.js auf DOM-Ereignisse zu **hören**.

Die `v-on`-Direktive verbindet einen sog. **Eventlistener** mit einem Element. Die Art des Ereignisse wird von einem Argument spezifiziert, z. B. hört `v-on:keyup` das `keyup`-Ereignis ab.

Info

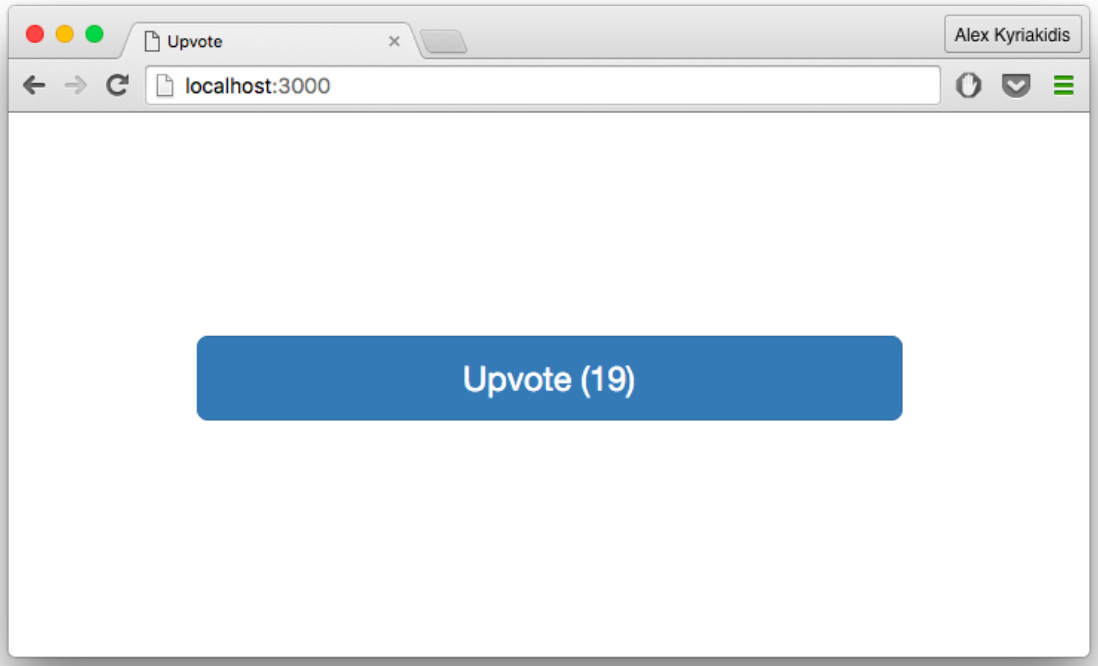
Das `keyup`-Ereignis tritt ein, wenn eine Taste losgelassen wird. Hier[6] finden Sie eine komplette Liste der `HTML`-Ereignisse.

Ereignisse Inline behandeln

Nun schauen wir uns die Ereignisbehandlung in Aktion an. Im Folgenden wird ein ‘Upvote’-Button eingeführt, der die Anzahl der positiven Rückmeldungen bei jedem Klicken hochzählt.

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

```
<html>
<head>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.cs\
s" rel="stylesheet">
<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.js"></script>
<script type="text/javascript">
new Vue({
    el: '.container',
    data: {
        upvotes: 0
    }
})
</script>
</html>
```



Upvotes-Zähler

In unseren Daten existiert eine `upvotes`-Variable. Ein **Eventlistener** für `click` wird mit dem Befehl rechts vom Gleichzeichen verbunden: Innerhalb der Anführungszeichen steht der Befehl, der die Anzahl der positiven Rückmeldung bei jedem Klicken um eins hochzählt (`upvotes++`).

Ereignisse mit Methoden behandeln

Nun werden wir das gleiche Ereignis wie zuvor behandeln. Dazu werden wir aber eine Methode verwenden. In Vue.js ist eine Methode ein Codeblock, der dazu erstellt wird, um eine bestimmte Aufgabe zu erfüllen. Um eine Methode auszuführen, wird sie zunächst definiert und dann ausgeführt.

```
<html>
<head>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.cs\
s" rel="stylesheet">
<title>Upvote</title>
</head>
<body>
    <div class="container">
        <button v-on:click="upvote">
            Upvote! {{upvotes}}
        </button>
    </div>
</body>
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/vue/2\
.3.4/vue.js"></script>
<script type="text/javascript">
new Vue({
    el: '.container',
    data: {
        upvotes: 0
    },
    // Methoden werden innerhalb des **`methods`**-Objekt definiert
    methods: {
        upvote: function(){
            // **`this`** innerhalb von Methoden zeigt auf die Vue-Instanz
            this.upvotes++;
        }
    }
})
</script>
</html>
```

Wir binden einen Click-Eventlistener an eine Methode, die wir '**upvote**' genannte haben. Das Ganze funktioniert genauso wie im vorherigen Beispiel, ist aber klarer strukturiert, einfacher zu lesen und zu verstehen.

Warnung

Eventhandler können **nur einen Befehl oder eine Funktion** ausführen.

Kurzform für `v-on`

Wenn Sie **`v-on`** überall und jedes Mal im Projekt einsetzen, merken Sie, dass der HTML-Code schnell anfängt unleserlich zu werden. Zum Glück gibt es eine Kurzform für **`v-on`**, nämlich das `@`-Symbol. Das `@` ersetzt dabei das komplette **`v-on:`** und so wird Ihr Code *viel sauberer und leserlicher*. Der Einsatz dieser Kurzform ist optional.

Unter Verwendung von `@` ändert sich der Code des Buttons dann wie folgt:

Mit `v-on:` auf 'click' hören

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

Mit der Abkürzung `@` auf 'click' hören

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

Event Modifier

Wir entwickeln nun eine Taschenrechner-Anwendung. Hierzu verwenden wir eine Form mit zwei Input-Elementen und einem Dropdown, um die gewünschte Operation auszuwählen.

Auch wenn der folgende Code in Ordnung aussieht, arbeitet unser Taschenrechner nicht wie erwartet.

```
1  <html>
2  <head>
3    <title>Calculator</title>
4    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.\
5  css" rel="stylesheet">
6  </head>
7  <body>
8    <div class="container">
9      <h1>Type 2 numbers and choose operation.</h1>
10     <form class="form-inline">
11         <!-- Beachten Sie, dass 'number' hier angegeben
12         wird, um die Eingaben in eine Zahl zu wandeln. -->
13         <input v-model.number="a" class="form-control">
```

```
14          <select v-model="operator" class="form-control">
15             <option>+</option>
16             <option>-</option>
17             <option>*</option>
18             <option>/</option>
19          </select>
20          <!-- Beachten Sie, dass 'number' hier angegeben
21          wird, um die Eingaben in eine Zahl zu wandeln. -->
22          <input v-model.number="b" class="form-control">
23          <button type="submit" @click="calculate"
24          class="btn btn-primary">
25              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>
```

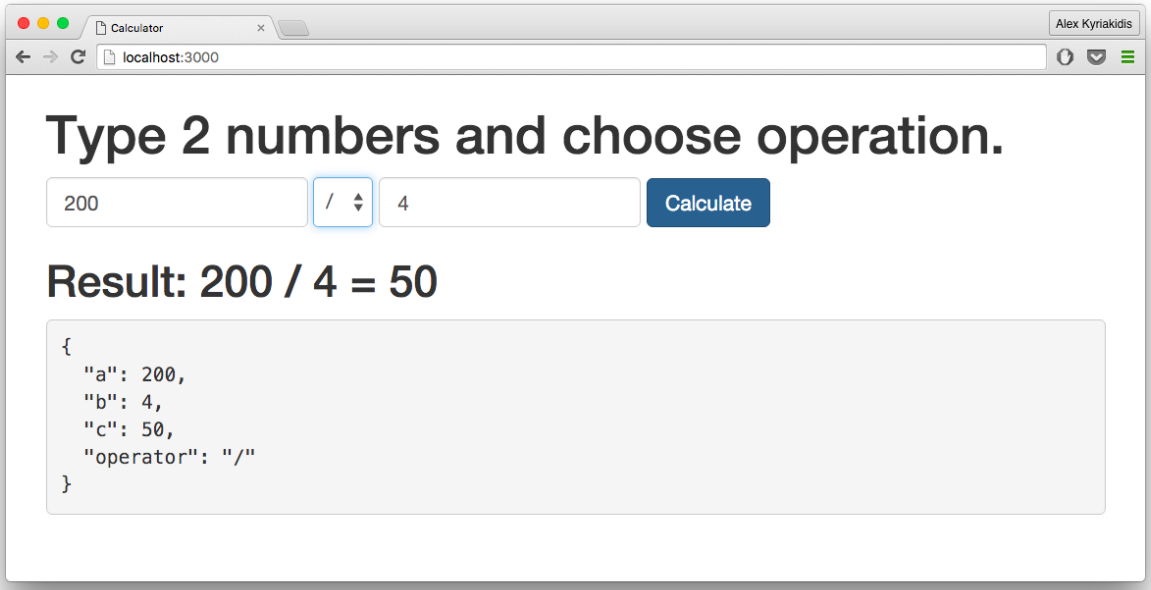
Beim Ausprobieren haben Sie sicherlich bemerkt, dass beim Klicken des “Calculate”-Buttons nicht das Ergebnis errechnet, sondern die Seite neu geladen wird.

Das macht absolut Sinn, denn wenn “Calculate” angeklickt wird, übertragen (“submitten”) Sie die Form im Hintergrund und deshalb wird die Seite neu geladen.

Um diese Übertragung der Form zu verhindern, muss das Standardverhalten des **`onsubmit`**-Ereignisses unterbunden werden. Das geschieht durch den Aufruf von **`event.preventDefault()`** innerhalb der Eventhandler-Methode. In unserem Fall heißt die Eventhandler-Methode **`calculate`**.

Also wird unsere Methode nun so aussehen:

```
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;
    }
}
```



Einsatz von Event-Modifiern in einem Taschenrechner

Auch wenn wir dieses Verhalten einfach innerhalb von Methoden unterbinden können, wäre es doch eigentlich besser, wenn diese Methoden die Notwendigkeit des Unterbindens des Standardverhaltens komplett ignorieren könnten und man sich auf die Logik konzentrieren kann, und nicht das Standardverhalten von Ereignissen im DOM berücksichtigen muss.

Vue.js hält für `v-on` vier sog. 'Event-Modifier' bereit, um das Standardverhalten von Ereignissen zu unterbinden:

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

Wenn wir also `.prevent` einsetzen, ändert sich der Submit-Button **von**:

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

in:

```
1  <!-- Das Submit-Ereignis führt nicht mehr zu einem Reload -->
2  <button type="submit" @click.prevent="calculate">Calculate</button>
```

Und wir können jetzt **`event.preventDefault()`** sicher aus dem Code unserer **`calculate`**-Methode entfernen.

Bemerkung

`.capture` und **`.self`** werden nur selten genutzt, also gehen wir hier nicht weiter darauf ein. Sollten Sie sich doch für die sog. *Event Order* interessieren, so finden sie in diesem Tutorial[7] weitere Informationen.

Key Modifier

Wenn Sie sich eines der Input-Felder ansehen, stellen Sie fest, dass die calculate-Methode ausgeführt wird, sobald Sie hier die Enter-Taste drücken. Wäre der Button nicht innerhalb des Formulars oder gäbe es keinen Button, dann könnten Sie stattdessen auf ein Tastatur-Ereignis warten und dies behandeln.

Beim Behandeln von Tastatur-Ereignissen wird oft auf die Tastencodes geprüft. Der Tastencode für **`Enter`** ist 13. Wir könnten ihn also wie folgt abfragen:

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

Sich alle Tastencodes zu merken, ist nicht unbedingt zielführend. Deshalb hält Vue Aliase für die gebräuchlichsten Tasten bereit:

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

Um also die `calculate`-Methode beim Drücken von Enter auszuführen, müssen die Inputs wie folgt geändert werden:

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

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

Tipp

Befinden sich in einem Formular eine große Anzahl von Inputs, Buttons, etc. und Sie müssen das Standard-Submit-Verhalten unterbinden, dann ändern Sie einfach das **`submit`**-Ereignis des Formulars.

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

Und schlussendlich funtioniert Ihr einfacher Taschenrechner.

Berechnete (“computed”) Properties

Vue.js Inline-Ausdrücke sind sehr bequem, aber um komplexere Logik zu implementieren sollte man berechnete Properties einsetzen. Im Grunde genommen sind berechnete Properties nichts anderes als Variablen, deren Wert von bestimmten Faktoren abhängt.

Berechnete Properties sind wie Funktionen, die man als Property einsetzen kann. Aber es gibt einen wesentlichen Unterschied: Jedes Mal, wenn sich eine Abhängigkeit einer solchen berechneten Property ändert, wird der Wert dieser berechneten Property neu ermittelt.

In Vue.js werden berechnete Properties im **`computed`**-Objekt innerhalb der **`Vue`**-Instanz definiert.

```
<html>
<head>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.cs\
s" rel="stylesheet">
<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: {
      // ein berechneter "Getter"
      b: function () {
        // **`this`** weist auf die Vue-Instanz
        return this.a + 1
      }
    }
});
</script>
</html>
```

Wir haben die Werte zweier Variablen gesetzt: Die Erste, **a**, wurde auf 1 gesetzt und die Zweite, **b**, wird auf das zurückgelieferte Ergebnis der Funktion im **computed**-Objekt gesetzt. In diesem Beispiel wird der Wert von **b** auf **2** gesetzt.

```
<html>
<head>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.cs\
s" rel="stylesheet">
<title>Hello 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/2.3.4/vue.js"></script>
<script type="text/javascript">
new Vue({
    el: '.container',
    data: {
      a: 1,
    },
    computed: {
```

```
24        // ein berechneter Getter
25        b: function () {
26          // **`this`** weist auf die vm-Instanz
27          return this.a + 1
28        }
29      }
30  });
31  </script>
32  </html>
```

Das obenstehende Beispiel gleicht dem Vorhergehenden bis auf eine Sache: Es gibt ein Input-Feld, das an die **a**-Variable gebunden ist. Das erwünschte Ergebnis wäre, den Wert der gebundenen Variablen zu ändern und sofort das Ergebnis in **b** zu aktualisieren. Aber das funktioniert hier nicht wie erwartet.

Wir der Code ausgeführt und die Variable **a** auf **5** gesetzt, erwartet man, dass **b** den Wert **6** erhält. Das sollte klar sein, aber **b** wird auf **51** gesetzt.

Wieso? Sie wissen es sicher schon: **b** übernimmt den Wert aus dem Input-Feld über die Variable **a** als String, und hängt einfach die **1** (in einen String gewandelt) an.

Eine mögliche Lösung des Problems wäre der Einsatz der **parseFloat()**-Funktion, die einen String ein einen numerischen Wert wandelt.

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

Eine weitere Möglichkeit wäre **<input type="number">** zu nutzen, um die Eingabe gleich auf numerische Werte zu beschränken.

Aber es gibt noch einen schöneren Weg: Über den Spezial-Modifier **.number** kann eine Benutzereingabe in Vue.js automatisch in einen numerischen Wert gewandelt werden.

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

Mit dem **.number**-Modifier erzielen wir das gewünschte Ergebnis ohne weiteren Aufwand.

Um die erweiterten Einsatzmöglichkeiten von berechneten Properties zu zeigen, werden wir in unserem Taschenrechner berechnete Properties anstelle von Methoden einsetzen.

Beginnen wir mit einem einfachen Beispiel, in dem eine berechnete Property **c** die Summe von **a** plus **b** erhält.

```
<html>
<head>
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.mi\
n.css" rel="stylesheet">
    <title>Hello Vue</title>
</head>
<body>
    <div class="container">
        <h1>Enter 2 numbers to calculate their sum.</h1>
        <form class="form-inline">
            <input v-model.number="a" class="form-control">
            +
            <input v-model.number="b" class="form-control">
        </form>
        <h2>Result: {{a}} + {{b}} = {{c}}</h2>
        <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,
    b: 2
```

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

Somit ist der Ausgangscode fertig. Der Benutzer tippt zwei Zahlen ein und bekommt deren Summe zurück. Das Ziel ist nun ein Taschenrechner, der vier Rechenoperationen ausführen kann.

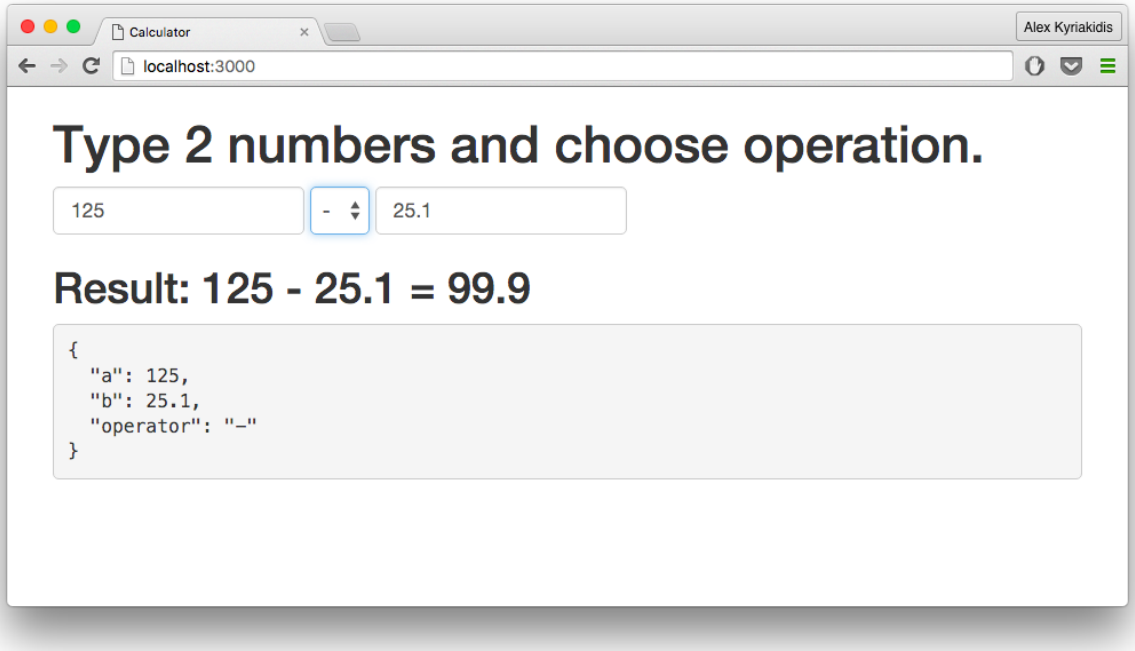
Da der `HTML`-Code der gleiche ist, wie beim Taschenrechner, den wir im vorhergehenden Abschnitt dieses Kapitels implementiert haben (außer dass wir jetzt keinen Button mehr benötigen), zeigen wir hier lediglich den JavaScript-Code.

```
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;
      }
    }
  },
});
```

Somit ist der Taschenrechner bereit für den Einsatz. Das Einzige was zu tun war, war den Code innerhalb der `calculate`-Method in die berechnete Property `c` zu übertragen! Wenn nun der Wert von `a` oder `b` geändert wird, ändert sich das Ergebnis in Echtzeit! Dazu brauchten wir weder Buttons noch Events noch andere Mittel. **Wie genial ist das??**

Bemerkung

Selbstverständlich müsste man hier noch mit einem `if`-Statement Divisionsfehler verhindern. Hierfür gibt es aber bereits Abhilfe: Wenn der Benutzer 1/0 eingibt, wird das Ergebnis automatisch "infinite"! Gibt der Benutzer einen Text ein, lautet das Ergebnis "NaN" ("Not a Number").



Ein Taschenrechner, erstellt mit berechneten Properties.

Codebeispiele

Sie finden Sie die Codebeispiele dieses Kapitels auf GitHub[8].

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

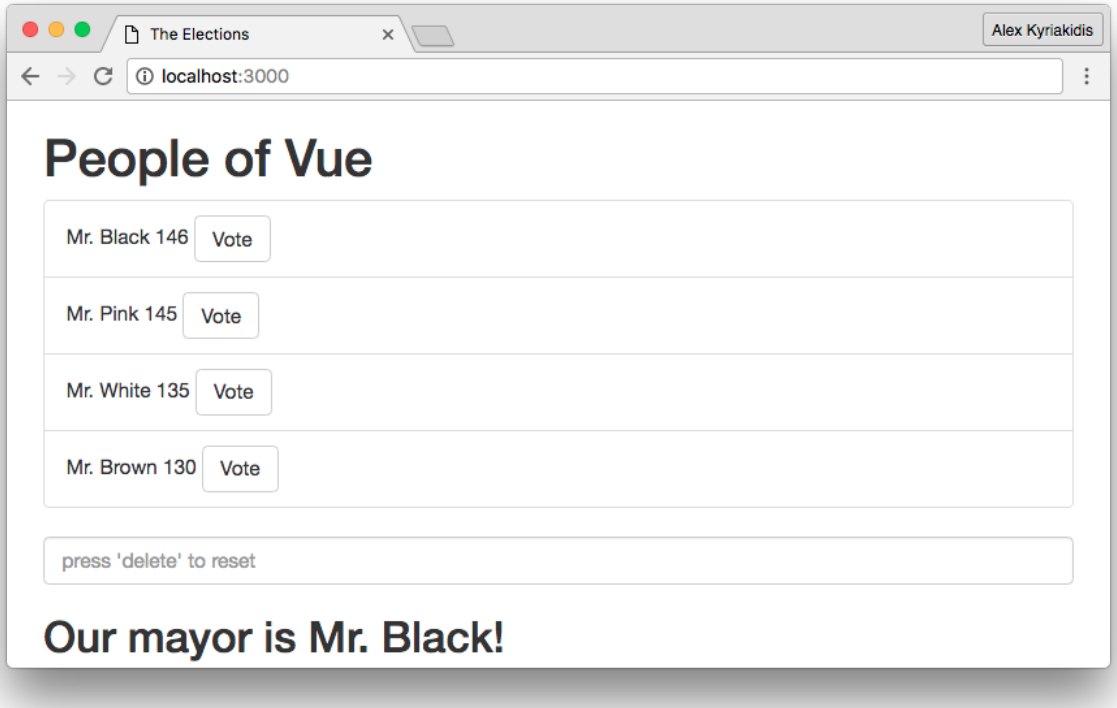
Hausaufgabe

Nun haben Sie ein grundlegendes Verständnis der Ereignisbehandlung von Vue, von Methoden, berechnete Properties, usw. Sie sollten also etwas mehr Herausforderndes ausprobieren. Beginnen Sie damit, ein Array von Bürgermeister-Kandidaten zu erstellen. Jeder Kandidat hat einen Namen und die Anzahl der Stimmen, die für ihn abgegeben wurden. Setzen Sie einen Button dazu ein, die Anzahl der Stimmen für die Kandidaten zu erhöhen. Setzen Sie zudem eine berechnete Property dazu ein, um den aktuellen Bürgermeister zu ermitteln und seinen Namen darzustellen.

Zum Abschluss fügen Sie ein Input-Feld hinzu. Erhält dieses Feld den Fokus und die '**delete**'-Taste wird gedrückt, beginnt die "Wahl" erneut. Das bedeutet: Die Anzahl aller Stimmen werden auf 0 gesetzt.

Hinweis

Die `sort()`- und `map()`-Methoden von JavaScript könnten sich als sehr nützlich erweisen und Key-Modifier bringen Sie der Lösung ebenso schnell nahe.



Beispiel-Ausgabe

Mögliche Lösung

Sie finden hier[9] eine mögliche Lösung zu dieser Übung.

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