VUE.JS 2 MAJESTELERİ (TÜRKÇE)

ALEX KYRIAKIDIS KOSTAS MANIATIS SİNAN ELDEM

The Majesty of Vue.js 2 (Türkçe)

Vue.js 2'yi sıfırdan öğren!

Alex Kyriakidis, Sinan Eldem ve Kostas Maniatis

Bu kitap http://leanpub.com/vuejs2-turkish adresinde satıştadır.

Bu versiyon, 2017-05-28 tarihinde yayınlanmıştır

Bu bir Leanpub kitabıdır. Leanpub, yazar ve yayımcıları Lean Yayımlama sistemi ile destekleyen bir kuruluştur. Lean Yayımlama, henüz çalışma aşamasında olan bir kitabı kullanışlı yollarla destekleyerek, okuyucu geri dönüşünü sağlayan ve prosesi kolaylaştıran bir yöntemdir.

Sevgili oğlum Muhammed Hamza'ya...

İçindekiler

Giriş

Vue.js Hakkında

Vue.js Genel Bakış

Vue (telaffuz edilişi / viyu: /, View 'görünüm' gibi), kullanıcı arayüzlerini oluşturmak için gelişimi sürdürülen bir çerçevedir. Vue, diğer monolitik çerçevelerin aksine, adım adım benimsenmesi için tasarlanmıştır. Çekirdek kütüphane yalnızca **görünüm katmanı** üzerinde yoğunlaşmıştır ve diğer kütüphaneler veya mevcut projelerle entegre edilmesi çok kolaydır. Öte yandan Vue, modern araçlarla ve destekleyici kütüphaneler[1] ile birlikte kullanıldığında gelişmiş Tek Sayfa Uygulamaları için gerekli altyapıyı sağlayabilir.

Deneyimli bir önyüz geliştiricisi iseniz ve Vue.js ürününün diğer kitaplıklar / çerçevelerle nasıl karşılaştırıldığını öğrenmek istiyorsanız, Diğer Çerçevelerle Karşılaştırma bölümüne göz atın.

Vue.js'nin çekirdeği hakkında daha fazla bilgi edinmek isterseniz Vue.js resmi rehberi[2] konusuna bakın.

Kullananlar Vue.js hakkında ne düşünüyorlar

"Vue.js beni JavaScript'e sevk eden şey, kullanımı son derece kolay ve keyifli. Temel hizmetlerini genişleten geniş eklenti ve ekosistemi mevcut. Küçük veya büyük herhangi bir projeye hızlıca ekleyebilir, birkaç satırlık bir kod yazabilir ve ayarlayabilirsiniz. Vue.js hızlı, hafif ve önyüz gelişiminin geleceğidir!"

-Alex Kyriakidis

"Javascript'i seçmeye başlayınca bir ton olasılık öğrenmek zorunda kalacağımdan endişenlendim, ancak arkadaşım Vue.js öğrenmeyi önerdiğinde ve tavsiyelerine uyup ikna oldum, işler harikalaştı. Öğretmenleri okurken ve izlerken şimdiye kadar yaptığım tüm şeyleri düşünmeye devam ettim ve Vue'yu daha önce öğrenmek için yatırım yapsaydım bunun ne kadar kolay olacağını düşündüm. Bence, çalışmalarınızı hızlı, güzel ve kolay yapmak istiyorsanız Vue, ihtiyaç duyduğunuz JS Çerçevesidir."

-Kostas Maniatis

---

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

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

"Sözlerimi not alın: Vue.js 2016'da popülerlikte gökyüzü-roketi olacak. Gerçekten o kadar iyi."

- Jeffrey Way

"Vue, her zaman bir JavaScript çerçevesinde istediğim şey, sizinle geliştirici olarak ölçeklendirilen bir çerçeve. Basit tek sayfa uygulaması veya Vuex ve Vue Router ile gelişmiş tek sayfa uygulaması oluşturabilirsiniz. Gerçekten şimdiye kadar gördüğüm en parlak JavaScript çerçevesi."

- Taylor Otwell

"Vue.js, tam donanımlı bir SPA'da olduğu gibi, sunucu tarafından oluşturulmuş bir uygulamada kullanmak için aynı derecede doğal olduğunu düşündüğüm ilk çerçeve. İster tek bir sayfada küçük bir widgete ihtiyacım olsun, ister karmaşık bir Javascript istemcisi oluşturuyor olayım, hiç bu kadar kolay olmamıştı. Yığınlar içinde kaybolmam gerekmiyor."

- Adam Wathan

"Vue.js, kullanımı basit ve kolay anlaşılır bir çerçevedir ve başkalarının en karmaşık çerçeveyi kimin yapabileceğini görmek için mücadele ettiği bir dünyada tertemiz bir soluk."

- Eric Barnes

"Vue.js'i sevmemin nedeni, melez bir tasarımcı/geliştirici olduğumdan, React, Angular ve diğerlerine baktım ama öğrenme eğrisi ve terminolojisi her zaman beni tercih dışında bıraktı. Vue.js ilk anladığım JS çerçevesi. Ayrıca, sadece benim gibi az güvenenler için JS'ler arasından birini seçmek kolay değil, aynı zamanda Angular and React geliştiricileri deneyimli ve fark ettiklerini gözlemledik ve onlar da Vue.js'yi beğeniyor. Javascript dünyasında bu benzeri görülmemiş bir şey ve bu nedenle Londra Vue.js Buluşması'na başladım."

-Jack Barham

Hoşgeldiniz

Kitap Hakkında

Bu kitap Vue.js adlı hızla yayılmakta olan Javascript Framework yolunda size rehberlik edecektir!

Bir süre önce, Laravel ve Vue.js temelli yeni bir proje başlattık. Vue.js 'kılavuzunu ve birkaç eğitici materyali okuduktan sonra, web'deki Vue.js ile ilgili kaynak eksikliğini keşfettik. Projemizin gelişimi sırasında çok deneyim kazandık, kazandıklarımızı dünyayla paylaşmak için bu kitabı yazma fikri geldi. Vue.js 2 çıktığına göre, tüm örneklerin ve bunların göreli içeriğinin yeniden yazıldığı ikinci bir sürüm yayınlayarak kitabımızı güncelleme zamanının geldiğine karar verdik.

Kitap, tüm örneklerin herkese yeterli rehberlik sağlayacak kadar ayrıntılı biçimde anlatıldığı gayri resmi, sezgisel ve kolay takip edilen bir formatta yazılmıştır.

Temel bilgilerle başlayacağız ve birçok örnekle vue.js 2'nin en önemli özelliklerini ele alacağız. Bu kitabın sonunda, hızlı önyüz uygulamaları oluşturabilir ve Vue.js entegrasyonu ile mevcut projelerinizin performansını arttırabilirsiniz.

Bu kitap kim içindir?

Modern web geliştirme öğrenmek için zaman harcayan herkes Bootstrap, Javascript ve birçok Javascript çerçevesi gördü.

Bu kitap, hafif ve basit bir Javascript çerçevesi öğrenmek isteyen herkes içindir.

Aşırı bilgi gerekmez, ancak HTML ve Javascript'i biliyorsanız daha iyi olur. Bir dize ile bir nesne arasındaki farkın ne olduğunu bilmiyorsanız, belki önce biraz araştırma yapmanız gerekebilir.

Bu kitap, zaten Vue.js'teki yollarını bilen ve bilgilerini genişletmek isteyen her okuyucu için de yararlıdır.

Temasa geçmek

Kitabınızla ilgili olarak bizimle iletişime geçmek, görüşlerinizi bildirmek veya dikkatinizi çekecek diğer konularda bize ulaşmak isterseniz, bizimle iletişime geçmekten çekinmeyin.

İsim	E-posta	Twitter
The Majesty of Vue.js	hello@tmvuejs.com	@tmvuejs
Alex Kyriakidis	alex@tmvuejs.com	@hootlex
Kostas Maniatis	kostas@tmvuejs.com	@kostaskafcas
Sinan Eldem	sinan@sinaneldem.com.tr	@sineld

Ev ödevi

Kod öğrenmenin en iyi yolu kod yazmaktır, bu nedenle bir çok bölümün sonunda sizin için bir şey öğrenip kendiniz öğrendiklerinizi test etmeniz için bir egzersiz hazırladık. Vue 2 için yapılan alıştırmaları ve çözümleri de güncelledik ve Vue.js hakkında daha iyi bir fikir sahibi olmak için bunları mümkün olduğunca çözümlemenizi tavsiye ediyoruz.

Belki birkaç farklı örnek veya yol size doğru fikri verecektir. Tabii ki acımasız değiliz, ipuçları ve potansiyel çözümleri de sunduk!

Yolculuğa başlayabilirsiniz!

Örnek kod

GitHub'da kitapta kullanılan kod örneklerinin çoğunu bulabilirsiniz. Buradan[3], bu kitaptaki kaynak kodlarına ulaşabilirsiniz.

İndirmeyi tercih ederseniz, bir [.zip] dosyasını burada[4] bulabilirsiniz.

Bu, kitaptan birşeyler kopyala/yapıştır yapmanızın önüne geçerek size zaman kazandıracaktır ancak unutmayın, en iyi öğrenme yolu kodlamaktır.

Hata Giderme

İçeriğimizin doğruluğunu sağlamak için her türlü özen gösterilmiş olsa da hatalar olur. Kitabın içinde hata bulur ve bize bildirirseniz minnettar oluruz. Bunu yaparak, diğer okuyucuları hayal kırıklığından koruyabilir ve bu kitabın sonraki sürümlerini iyileştirmemize yardımcı olabilirsiniz. Herhangi bir hata bilgisi bulursanız, lütfen GitHub deposu[5]'nda bir issue gönderin.

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

Sözleşmeler

Kitap boyunca aşağıdaki gösterimsel sözleşmeler kullanılmıştır.

Bir kod bloğu aşağıdaki şekilde ayarlanır:

JavaScript

```
function(x, y){
    // Bu bir yorumdur
}
```

Metin içindeki kod kelimeleri ve veriler şöyle gösterilir: "`.container` değişkenini duyarlı bir sabit genişlikli konteyner için kullanın."

Yeni terim ve **önemli kelimeler** kalın olarak gösterilmiştir.

İpuçları, **notlar ve uyarılar** aşağıdaki gibidir:

Bu bir Uyarı

Bu öğe bir uyarı veya dikkat gösteriyor.

Bu bir İpucu

Bu öğe bir işaret gösterir veya öneride bulunur.

Bu bir bilgi kutusu

Bazı özel bilgiler burada.

Bu bir nottur

Konuyla ilgili bir not.

Bu bir ipucu

Konuyla ilgili bir ipucu.

Bu bir terminal komutudur

Terminalde çalıştırılacak komut.

Bu bir karşılaştırma metni

Konu ile ilgili şeyleri karşılaştıran küçük bir metin.

Bu Github bağlantısıdır.

Bağlantılar, her kitabın kod örneklerini ve olası ödev çözümlerini bulabileceğiniz bu kitabın ambarına gider.

Etkileşimlilik

Bu bölümde önceki örnekleri yeniden oluşturacak ve genişleteceğiz, 'yöntemler (methods)', 'olay işleme (event handling)' ve 'hesaplanmış özellikler (computed properties)' ile ilgili yeni şeyler öğreneceğiz. Farklı yaklaşımları kullanarak birkaç örnek geliştireceğiz. Şimdi, Vue'nun etkileşimini, Hesap Makinesi gibi güzel ve kolay çalışan küçük bir uygulama inşa ederek öğrenmenin tam zamanı.

Olay İşleme

`HTML` olayları, DOM öğelerinin başına gelenlerdir. Vue.js HTML sayfalarında kullanıldığında, bu olaylara **tepki** verebilir.

Olaylar, temel kullanıcı etkileşimlerinden, işleme modelinde olanlara kadar her şeyi temsil edebilir.

`HTML` olaylarına bazı örnekler:

- Bir web sayfasının yüklenmesi tamamlandı
- Bir giriş alanı değiştirildi
- Bir düğmeye tıklandı
- Bir form gönderildi

Olay işleme eylemi, herhangi bir olay gerçekleştiğinde bir şeyler yapabilmenizdir.

Vue.js'de, olayları **dinlemek (listen)** için `v-on` direktifini kullanabilirsiniz.

`v-on` direktifi, bir öğeye bir olay dinleyicisi ekler. Olayın türü argüman ile gösterilir, örneğin `v-on:keyup` olayı `keyup` olayını dinler.

Bilgi

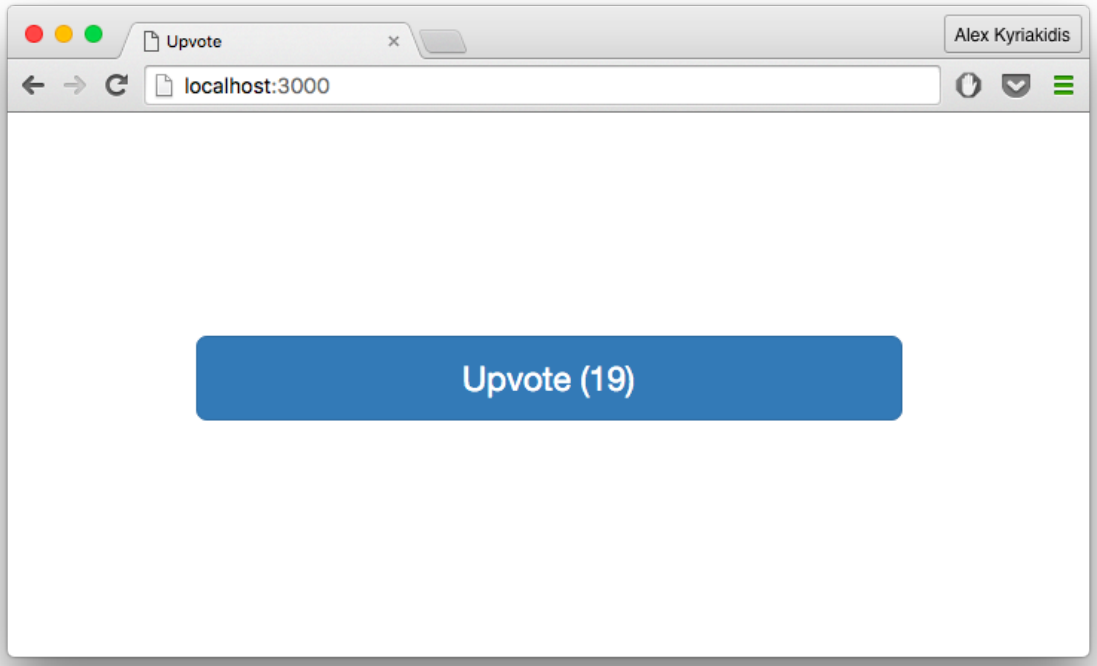
`keyup` olayı, kullanıcı bir tuşu bıraktığında meydana gelir. `HTML` olaylarının tam listesini burada[6] bulabilirsiniz.

Olayları Satır-içi İşleme

Yeterince çene çaldık şimdi ilerleyelim ve olay yönetimini harekete geçirelim. Aşağıda, her tıklandığında upvotes sayısını arttıran bir 'Upvote (Oy Arttır)' düğmesi vardır.

[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>Oy Arttır</title>
</head>
<body>
    <div class="container">
        <button v-on:click="upvotes++">
            Oy Arttır! {{upvotes}}
        </button>
    </div>
</body>
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/vue/2\
.2.2/vue.js"></script>
<script type="text/javascript">
new Vue({
    el: '.container',
    data: {
        upvotes: 0
    }
})
</script>
</html>
```



Oy Arttırma Sayacı

Verilerimiz içinde bir `upvotes` değişkeni var. Bu durumda `click` için bir olay dinleyicisini hemen yanında bulunan ifadeyle bağlarız. Tırnak işaretlerinin içinde, butona her bastığımızda oy artışını (`upvotes++`) kullanarak sadece birer artırım sayısını arttırıyoruz.

Olayları Yöntemleri Kullanarak Ele Alma

Şimdi bu örneğin aynısını yöntem kullanarak inşa edeceğiz. Vue.js'de yöntemler belirli bir görevi gerçekleştirmek için tasarlanmış kod bloklarıdır. Bir yöntemi çalıştırmak için onu tanımlamanız ve daha sonra çağırmanız gerekir.

```
<html>
<head>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.cs\
s" rel="stylesheet">
<title>Oy Arttır</title>
</head>
<body>
    <div class="container">
        <button v-on:click="upvote">
```

```
            Oy Arttır! {{upvotes}}
        </button>
    </div>
</body>
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/vue/2\
.2.2/vue.js"></script>
<script type="text/javascript">
new Vue({
    el: '.container',
    data: {
        upvotes: 0
    },
    // ** `methods` ** nesnesinin altındaki yöntemleri tanımlar
    methods: {
        upvote: function () {
            // **`this`** Vue örneği içindeki yöntemleri işaret eder
            this.upvotes++;
        }
    }
})
</script>
</html>
```

Bir tıklama olayı dinleyicisini '**upvote**' adlı bir yönteme bağladık. Tıpkı daha önce olduğu gibi çalışıyor, ancak kodunuzun okunuşu daha temiz ve anlaşılır.

Uyarı

Olay işleyicileri **yalnızca bir deyimi** yürütmekle sınırlandırılmıştır.

V-on'un Kısaltılmış Hali

Kendinizi bir projedeki `v-on` uygulamasını kullanır bulduğunuzda, HTML'inizin hızla kirlendiğini göreceksiniz. Neyse ki `v-on` için `@` kısa kullanım sembolü vardır. `@` tüm `v-on:` yerine geçer ve onu kullanırken kod *çok daha temiz* görünür. Kısaltmayı kullanmak tamamen isteğe bağlıdır.

`@` kullanımı ile önceki örneğimizin düğmesi şöyle olacaktır:

`v-on:` ile 'click' olayını dinliyoruz

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

`@` kısaltması ile 'click' olayını dinliyoruz

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

Olay Değiştiriciler

Şimdi devam edip bir Hesap Makinesi uygulaması oluşturacağız. Bunu yaparken, istenen işlemi seçmek için iki girişli ve bir açılır menü içeren bir form kullanacağız.

Aşağıdaki kod iyi görünse de hesap makinemiz beklendiği gibi çalışmıyor.

```
<html>
<head>
  <title>Hesap Makinesi</title>
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.\
css" rel="stylesheet">
</head>
<body>
  <div class="container">
    <h1>2 rakam girin ve işlemi seçin.</h1>
    <form class="form-inline">
        <!-- Burada, sayıları sayı olarak ayrıştırmak için
        'sayı' özel değiştiricisinin geçirildiğine dikkat edin.-->
        <input v-model.number="a" class="form-control">
        <select v-model="operator" class="form-control">
           <option>+</option>
           <option>-</option>
           <option>*</option>
           <option>/</option>
        </select>
        <!-- Burada, sayıları sayı olarak ayrıştırmak için
        'sayı' özel değiştiricisinin geçirildiğine dikkat edin.-->
        <input v-model.number="b" class="form-control">
```

```
        <button type="submit" @click="calculate"
        class="btn btn-primary">
            Hesapla
        </button>
    </form>
    <h2>Sonuç: {{a}}  {{operator}}  {{b}} = {{c}}</h2>
    <pre>
        {{ $data }}
    </pre>
</div>
</body>
<script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.2.2/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>
```

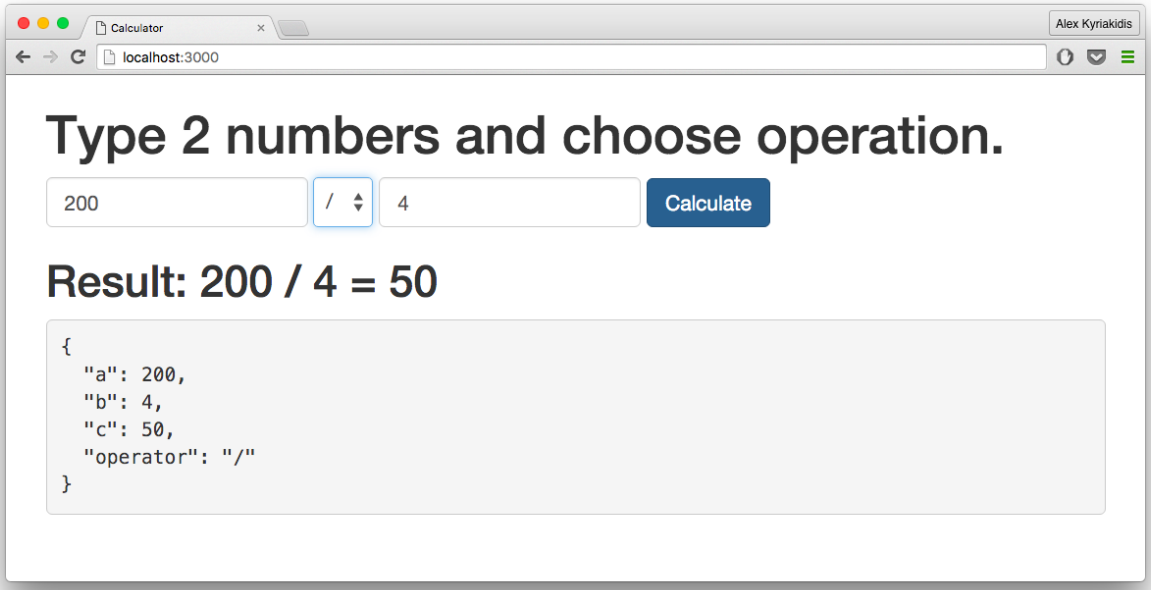
Bu kodu kendiniz çalıştırmayı denerseniz, “hesapla” düğmesi tıklandığında, hesaplamak yerine sayfayı yeniden yükler.

Bu mantıklıdır, çünkü “hesapla”‘yı tıkladığınızda arka planda formu gönderiyorsunuz ve böylece sayfa yeniden yüklenir.

Formun gönderilmesini önlemek için, **`onsubmit`** olayının varsayılan eylemini iptal ettirmeliyiz. Olay işleme yöntemimizde **`event.preventDefault()`**‘u çağırmak çok genel bir kullanımdır. Bu durumda, olay işleme yöntemi **`calculate`** ismini almıştır.

Böylece, yöntemimiz şu şekilde olacak:

```
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;
    }
}
```



Hesap makinesi oluşturmak için Etkinlik Değiştiricileri kullanımı

Bunu yöntemlerde kolayca yapabilsek de, yöntemlerin DOM etkinlik ayrıntılarıyla uğraşmak yerine veri mantığı hakkında cahil olması daha iyi olur.

Vue.js, olay varsayılan davranışını önlemek amacıyla `v-on` için dört olay değiştirici sağlar:

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

Yani, `.prevent`'i kullanarak, gönderme düğmemizi değiştirebiliriz. **bundan**:

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

buna:

```
<!-- gönderme olayı artık sayfayı yeniden yüklemeyecektir -->
<button type ="submit" @click.prevent ="calculate">Hesapla</button>
```

Ve şimdi `calculate` yöntemimizden `event.preventDefault()`'u güvenli bir şekilde kaldırabiliriz.

Not

`.capture` ve `.self` nadiren kullanılmaktadır, bu nedenle daha fazla ayrıntılandırma zahmetine girmeyeceğiz. *Etkinlik Sırası* hakkında daha fazla bilgi edinmek isterseniz öğreticiye[7] bakabilirsiniz.

Anahtar Değiştiriciler

Girişlerden birine odaklanırken enter'a basarsanız, calculate metodunun çağrıldığını göreceksiniz. Düğme formun içinde değilse veya hiç düğme yoksa, onun yerine klavye olayını dinleyebilirsiniz.

Klavye olaylarını dinlerken sıklıkla anahtar kodlarını kontrol etmemiz gerekiyor. `Enter` düğmesinin tuş kodu 13'tür. Dolayısıyla bunu şöyle kullanabiliriz:

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

Tüm önemli kodları hatırlamak güç olduğundan, Vue en yaygın kullanılan tuşlar için takma adlar sağlar:

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

Dolayısıyla, örneğimizde Enter'a basıldığında `calculate` yöntemini uygulamak için girişler şu şekilde olacaktır:

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

İpucu

Bir sürü giriş / düğme / vb. içeren bir formunuz olduğunda ve varsayılan gönderim davranışını önlemeniz gerekiyorsa, formun `submit` olayını değiştirebilirsiniz.

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

** Sonunda, hesap makinemiz hazır ve çalışıyor.**

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

Hesaplanan Özellikler

Vue.js satır-içi ifadeleri çok uygundur, ancak daha karmaşık mantık için hesaplanan özellikleri (computed properties) kullanmalısınız. Pratikte hesaplanan özellikler, değerlerinin diğer faktörlere bağlı olduğu değişkenlerdir.

Hesaplanan özellikler, nitelik olarak kullanabileceğiniz fonksiyonlar gibi çalışır. Fakat önemli bir fark vardır. Hesaplanan özelliğin bağımlılığı her değiştiğinde, hesaplanan özelliğin değeri yeniden değerlendirilir.

Vue.js'de hesaplanan özellikler **`Vue`** örneği içinde **`computed`** nesnesinde tanımlanır.

```
<html>
<head>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.cs\
s" rel="stylesheet">
<title>Merhaba 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/2.2.2/vue.js"></script>
<script type="text/javascript">
new Vue({
    el: '.container',
    data: {
      a: 1,
    },
    computed: {
      // hesaplanmış bir alıcı
      b: function () {
        // **`this`** Vue örneğini işaret eder
        return this.a + 1
      }
    }
});
</script>
</html>
```

İki değişken ayarladık, İlki **a** olarak 1'e, ikincisi **b** ise hesaplanmış nesnedeki fonksiyonun döndürülen sonucu tarafından tanımlanır. Bu örnekte, **b** değeri **2** olarak tanımlanacaktır.

```
<html>
<head>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.cs\
s" rel="stylesheet">
<title>Merhaba 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/2.2.2/vue.js"></script>
<script type="text/javascript">
new Vue({
    el: '.container',
    data: {
      a: 1,
    },
    computed: {
      // hesaplanmış bir alıcı
      B: function () {
        // **`this`** vm örneğini işaret eder
        return this.a + 1
      }
    }
});
</script>
</html>
```

Yukarıdaki örnek önceki ile aynıdır, ancak tek bir farkla. Bir girdi **a** değişkenine bağlıdır. İstenilen sonuç bağlanan özniteliğin değerini değiştirmek ve **b**'nin sonucunu derhal güncellemek olacaktır. Ancak, beklediğimiz gibi çalışmadığına dikkat edin.

Bu kodu çalıştırıp değişken **a**'yı **5**'e ayarlarsanız, **b**'nin **6**'ya eşit olmasını beklersiniz. Tabii, ama öyle değil, **b** burada **51** olarak ayarlanmış oldu.

Bu Neden oluyor? Daha önce de düşündüğünüz gibi, **b** değeri **a**'dan verilen bir değeri dize olarak alır ve numarasının sonuna **1** ekler.

Olası bir çözüm, **parseFloat()** fonksiyonu kullanarak dizeyi ayrıştırmak ve kayan nokta sayılarına döndürmektir.

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

Akla gelen bir başka seçenek, sayısal bir değer içermesi gereken girdi alanları için kullanılan **<input type ="number">**'ı kullanmaktır.

Ancak daha düzgün bir yol var. Vue.js ile, kullanıcı girdisini otomatikman sayı olarak kalıcı olmasını isterseniz **.number** özel değiştiricisini ekleyebilirsiniz.

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

number değiştiricisi bize daha fazla çaba sarf etmeden istenilen sonucu verecektir.

Hesaplanan özelliklerin daha geniş bir resmini göstermek için bunları kullanacağız ve daha önce gösterdiğimiz hesap makinesini inşa edeceğiz fakat bu sefer yöntemler yerine hesaplanan özellikleri kullanacağız.

Basit bir örnek ile başlayalım, burada hesaplanmış bir özellik **c**, **a** artı **b** toplamını içeriyor.

```
<html>
<head>
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.mi\
n.css" rel="stylesheet">
    <title>Merhaba Vue</title>
</head>
<body>
    <div class="container">
        <h1>Toplamlarını hesaplamak için 2 rakam girin.</h1>
        <form class="form-inline">
            <input v-model.number="a" class="form-control">
            +
            <input v-model.number="b" class="form-control">
        </form>
        <h2>Sonuç: {{a}} + {{b}} = {{c}}</h2>
        <pre> {{ $data }} </pre>
    </div>
</body>
<script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.2.2/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>
```

Başlangıç kodu hazır, bu noktada kullanıcı 2 sayı yazıp bunların toplamını alabilir. Amacımız dört temel işlemi gerçekleştirebilen bir hesap makinesi yapmak, bu nedenle inşa etmeye devam edelim!

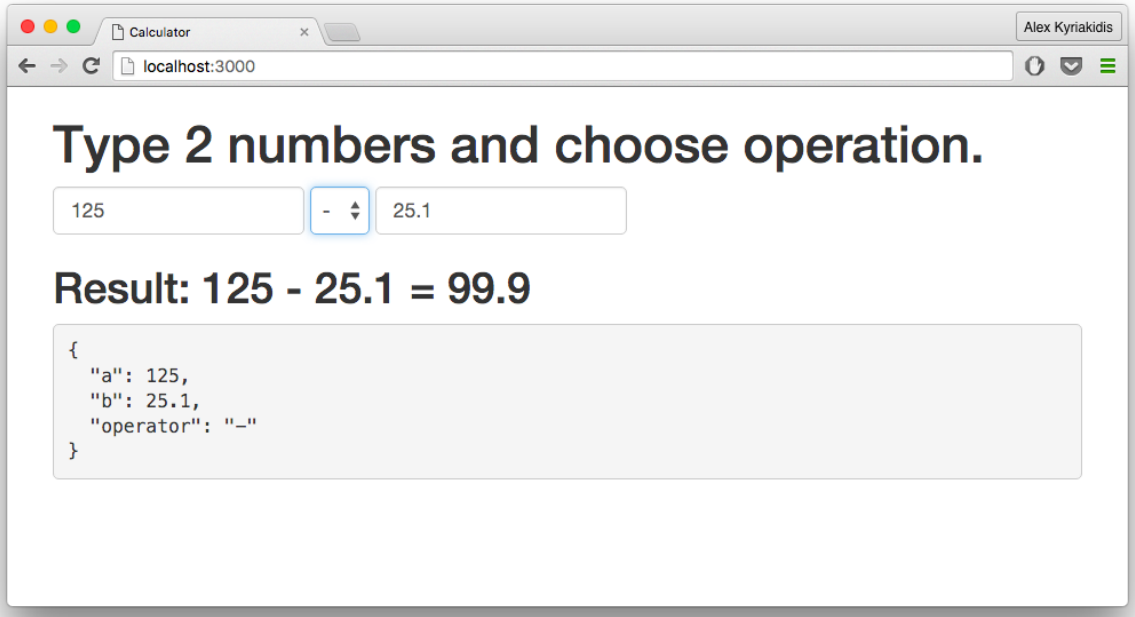
`HTML` kodu, bundan önceki bölümünde hazırladığımız hesap makinesi ile aynı olacağından (tek fark, şimdi bir düğmeye ihtiyacımız yok) yalnızca Javascript kod bloğunu göstereceğim.

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

Hesap makinesi kullanıma hazırdır. Yapmamız gereken tek şey, içinde bulunan her şeyi **`calculate`** yönteminin hesaplanan özelliği olan **`c`**'ye taşımak oldu! **`a`** veya **`b`** değerini değiştirdiğinizde sonuç gerçek zamanlı olarak güncellenir! Herhangi bir düğmeye, olaya veya başka bir şeye ihtiyacımız yok. **Ne kadar harika!**

Not

Buradaki normal yaklaşımın, bölme hatasından kaçınmak için **`if`** ifadesine sahip olacağını unutmayın. Ancak, zaten bu tür kusurlara dair bir tahmin zaten var. Kullanıcı 1/0 yazarsa, sonuç otomatik olarak sonsuza döner! Kullanıcı bir metin yazarsa, görüntülenen sonuç "sayı değildir" şeklindedir.



Hesaplanan özelliklerle oluşturulmuş hesap makinesi

Kod Örnekleri

Bu bölüme ait kod örneklerini GitHub[8] adresinde bulabilirsiniz.

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

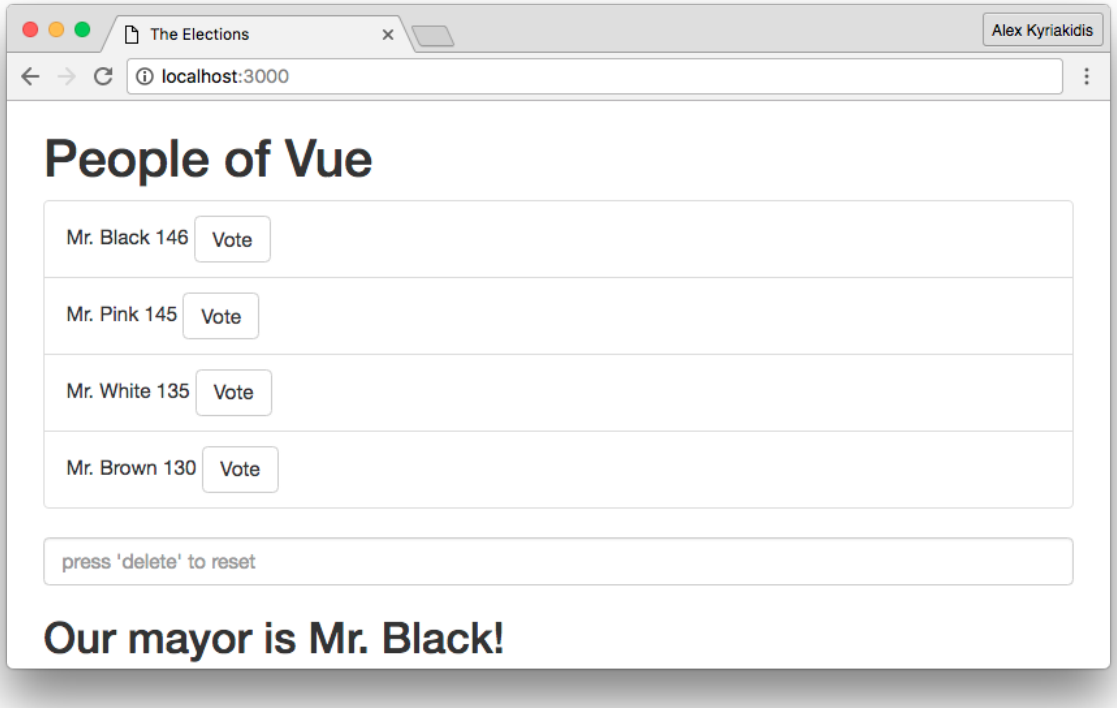
Ev ödevi

Şimdi, Vue'nun olay işleme, yöntemler, hesaplanmış özellikleri vb. hakkında temel bilgiye sahip olduğunuzdan, biraz daha zorlayıcı bir şey denemelisiniz. Bir "Sınıf Başkanı" adayı dizisi oluşturarak başlayın. Her adayın bir "adı" ve bir dizi "oyu" vardır. Her adayın oy sayısını artırmak için bir düğme kullanın. Mevcut "Sınıf Başkanı"'nın kim olduğunu belirlemek için hesaplanmış bir özellik kullanın ve adını gösterin.

Son olarak, bir girdi ekleyin. Bu girdi üzerine odaklanın ve '**sil**' tuşuna basıldığında, seçimler en başından başlasın. Bu, tüm oyların 0 olması anlamına gelir.

İpucu

Javascript'in `sort()` ve `map()` yöntemleri çok faydalı olabilir ve anahtar değiştiriciler sizi oraya götürecektir.



Örnek Çıktı

Potansiyel Çözüm

Bu alıştırma için potansiyel çözümü burada[9] bulabilirsiniz.

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