

Mari Belajar Vue.Js



Vue.js

Reactive Components for Modern Web Interfaces

Teten Nugraha

Tennugraha777@gmail.com

Kata Pengantar

Puji dan syukur kehadiran Allah SWT atas limpahan rahmat dan karunianya sehingga Ebook Mari Mengenal Framework Vue.js tahun 2016 dapat diselesaikan. Buku ini merupakan buku panduan bagi yang sudah atau akan mempelajari framework javascript di *Client Side* dan memberikan petunjuk serta gambaran bagaimana menggunakan framework Vue.js ini.

Terima kasih disampaikan kepada orang tua, partner hidup saya yang selalu memberikan semangat dalam menyelesaikan buku ini. Juga teman-teman dan pihak-pihak yang terkait yang memberikan saran, kritik dan kontribusi dalam penyelesaian buku ini.

Saya menyadari masih banyak terdapat kekurangan dalam buku ini, untuk itu kritik dan saran terhadap penyempurnaan buku ini sangat diharapkan dapat di kirim ke tennugraha777@gmail.com . Semoga Ebook ini dapat memberi manfaat bagi para pembaca dan khususnya untuk para *developer* di Indonesia.

Bandung, 19 Juli 2016

Teten Nugraha S.Kom

Vue JS



1. Pendahuluan

- Apa itu Vue.js ?

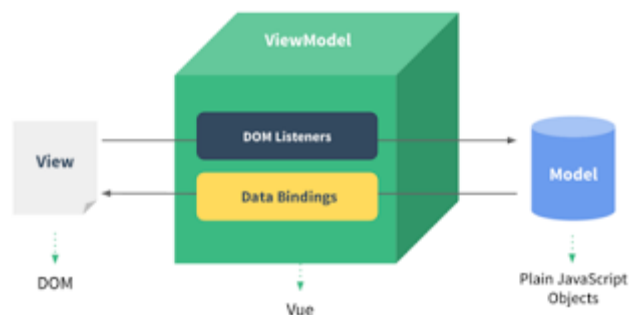
Dilansir dari situs Vue.js, Vue.js adalah sebuah librari javascript untuk membangun antarmuka web yang interaktif. Fokus utama Vue.js adalah menyediakan mekanisme *Reactive Data Binding* dan menyediakan fungsi untuk mengatur komponen *View* dengan API yang sesederhana mungkin.

Vue.js di buat untuk di fokuskan pada komponen *Frontend*. Jika Anda pernah menggunakan framework Angular, akan lebih mudah jika menggunakan Vue.js. dan pengalaman saya pun sangatlah gampang mengintegrasikan Vue.js dengan proyek-proyek yang sedang saya kerjakan.

Vue.js adalah framework javascript yang serupa dengan Angular, React, Polymer, Riot dan lain-lain. Tapi bagaimanapun juga, Vue.js menyediakan kesederhanaan, *performance*, *flexibility*. Dalam situs nya, Anda bisa melihat bagaimana perbandingan Vue.js dengan framework-framework javascript serupa.

Framework ini ditulis oleh Evan You seorang developer profesional. Bisa dilihat disitus nya <http://evanyou.me/> atau diakun github nya <https://github.com/yyx990803> .

- Reactive Data Binding



Salah satu fitur yang ditawarkan oleh Vue.js adalah *System Reactive Data Binding* yang berfungsi agar data dan DOM tetap terikat bersama-sama. Jika Anda biasanya menggunakan jQuery secara manual dalam memanipulasi DOM, kode yang ditulis pastilah berulang-ulang dan rawan kesalahan atau error.

Vue.js juga menganut *data-driven*. Yang berarti kita menggunakan sintaks khusus dalam template HTML yang kita gunakan untuk “mengikat” DOM dengan data yang telah kita buat. Setelah proses binding diciptakan, DOM kemudian akan bersinkronisasi dengan data. Setiap kali mengubah data, DOM juga akan memperbaruinya (*Two Way Data Binding*) dan hal ini membuat kode kita lebih mudah untuk ditulis dan mudah untuk dikembangkan.

- Component System

Component System adalah salah satu konsep yang penting di Vue.js, karena ini adalah sebuah abstraksi yang memungkinkan kita membangun aplikasi yang besar dan konsep penggunaan kembali komponen yang sudah dibangun (*reusable*). Kebanyakan antarmuka dari sebuah aplikasi bisa menjadi sebuah abstraksi ke dalam bentuk *tree* sebagai berikut :



Dengan Vue.js, kita bisa memanipulasi seakan-akan kita membuat tag HTML sendiri. Ini merupakan salah satu fitur Vue.js **Custom Element**. Dengan *Custom Element*, kita bisa membuat elemen-elemen tag HTML sesuai dengan kebutuhan proyek kita.

2. Download dan Instalasi Vue.js

Ada beberapa langkah untuk menginstall Vue.js, diantaranya :

a. Download di situs Vue.js

Anda bisa langsung mengakses situs Vue.js di <https://vuejs.org/> . di situs tersebut, kita diberikan dua pilihan alternatif unduh, untuk versi Development dan versi Production. Khusus untuk versi development yang *minified*, kita harus lebih hati-hati dalam menggunakan versi ini, karena versi ini masih dalam tahap pengembangan, jadi kadang-kadang muncul beberapa bug pada saat kita menggunakan nya.

Development Version

Production Version

File Vue.js yang sudah di download tadi, kemudian simpan dalam proyek Anda (file html dan php). Berikut adalah contoh instalasi Vue.js menggunakan cara mendownload.

lokal

```
<html>
<head>
<body>

<script src="js/vue.min.js"></script>
</script>
</body>
</html>
```

b. Menggunakan CDN

Untuk pilihan ini, kita tidak perlu repot-repot menyimpan file Vue.js dalam proyek kita. Yang kita perlukan hanya memanggil file Vue.js ini di situs **jsDeliver** atau **cdnjs** dengan syarat kita sudah terkoneksi ke internet. Contoh nya <https://cdn.jsdelivr.net/vue/1.0.26/vue.min.js> atau <https://cdnjs.cloudflare.com/ajax/libs/vue/1.0.26/vue.min.js> .

jsDeliver

```
<html>
<head>

<body>

<script src="https://cdn.jsdelivr.net/vue/1.0.26/vue.min.js"></script>
</script>
</body>
</html>
```

CDN

```
<html>
<head>

<body>

<script src="https://cdn.jsdelivr.net/vue/1.0.26/vue.min.js"></script>
</script>
</body>
</html>
```

c. Menggunakan Bower

Jika sering menggunakan bower, maka lebih mudah lagi. Cukup dengan ketikkan sintak berikut, maka Vue.js sudah terinstall dalam proyek Anda.

```
bower install vue
```

bower

```
<html>
<head>

<body>

<script src="bower_components/vue/dist/vue.min.js"></script>
</script>
</body>
</html>
```

!!!! Dalam buku ini tidak akan membahas mengenai Bower, tapi jika Anda tertarik mempelajari mengenai bower dan serius mendalami dunia web developer, akan sangat bermanfaat jika Anda mempelajari mengenai bower khususnya untuk manajemen asset.

3. Requirement

Kebutuhan / *tool* yang sudah harus disediakan diantaranya :

a. Text Editor

Ada banyak text editor yang *free* yang bebas digunakan tanpa harus membeli lisensi terlebih dahulu. Penulis menggunakan text editor Sublime Text Versi 3 yang sudah diinstall Emmet (plugin untuk mempercepat penulisan kode HTML). Bisa juga menggunakan PhpStorm, Aptana, Netbeans, Notepad++ dan lain-lain.



b. Browser

Untuk menampilkan output dari kode yang telah dibuat. Bisa menggunakan Google Chrome, Mozilla Firefox, Opera dan lain-lain.



Diatas adalah *list* yang harus sudah diinstall, adapun daftar dibawah ini adalah beberapa *tool* tambahan yang bersifat *optional* yang bisa Anda install ataupun tidak, diantaranya:

c. Webserver

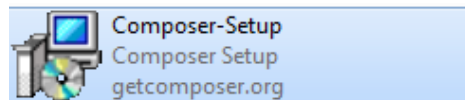
Tool ini sangat dibutuhkan untuk menjalankan aplikasi-aplikasi yang dibuat menggunakan bahasa pemrograman PHP. Anda bisa menggunakan XAMP, WAMP, AppServ dan lain-lain.



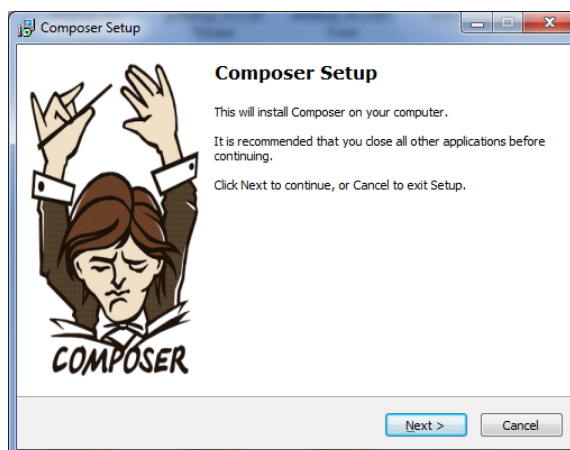
d. Composer

Karena penulis menjalankan kode nantinya menggunakan PHP Server Built in, maka terlebih dahulu install *composer*. Composer adalah sebuah *dependency* 'manager' untuk PHP. Anda dapat menambah library yang dibutuhkan untuk website Anda secara otomatis tanpa perlu mendownload satu persatu. Mirip dengan apt-get install pada sistem operasi linux. Adapun proses instalasi composer adalah sebagai berikut :

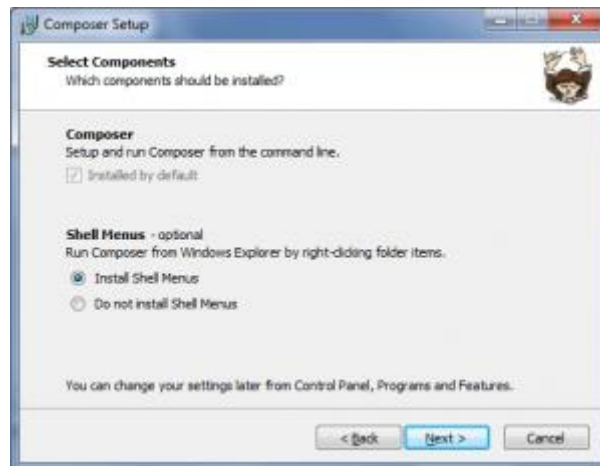
- Unduh composer di <https://getcomposer.org/Composer-Setup.exe>,



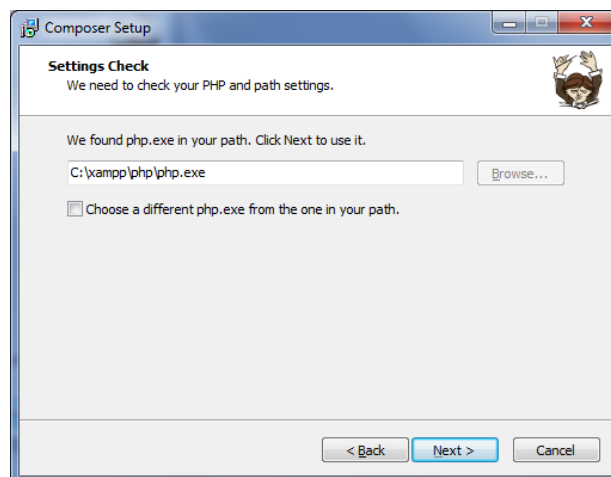
- Klik dua kali file tersebut sehingga muncul dialog setup install composer kemudian klik Next



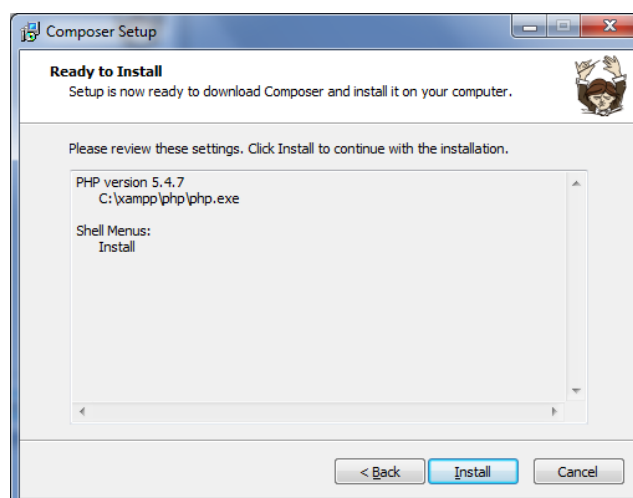
- Memilih komponen yang akan diinstall. Pilih "Install Shell Menus" kemudian klik next



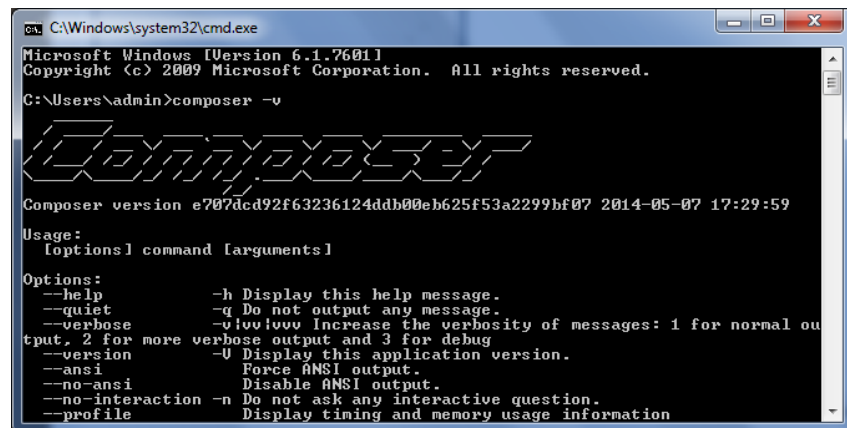
- Check versi php. Pada tombol “browse”, kemudian masukan path php yang sudah diinstall di komputer. Disini dicontohkan path php nya yaitu di “C:/xampp/php/php.exe” kemudian klik next. (Penulis Menggunakan XAMPP)



- Jika versi php sudah memenuhi standar instalasi Laravel, maka akan muncul tampilan sebagai berikut. Kemudian klik Install



- Jika sudah berhasil instal composer, untuk mengecek apakah composer sudah berjalan dengan baik, buka Command Prompt kemudian tuliskan “*composer -v*”. Jika berhasil composer akan tampil sebagai berikut .



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

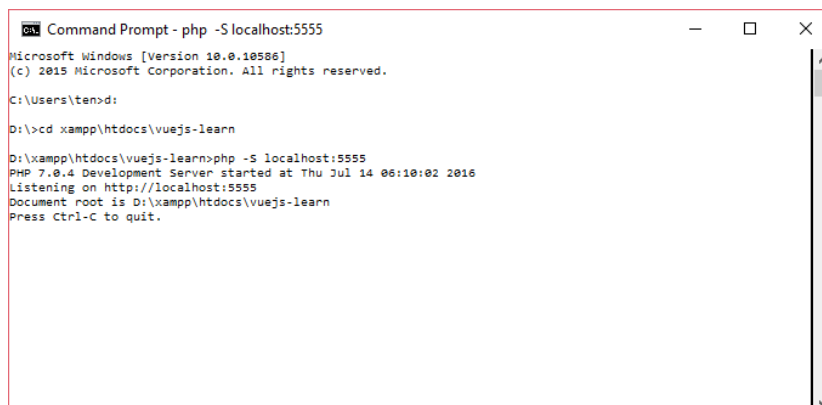
C:\Users\admin>composer -v

Composer version e707dcd92f63236124ddb00eb625f53a2299bf07 2014-05-07 17:29:59

Usage:
[options] command [arguments]

Options:
--help           -h Display this help message.
--quiet          -q Do not output any message.
--verbose        -vvvvvvv Increase the verbosity of messages: 1 for normal ou
tput, 2 for more verbose output and 3 for debug
--version        -V Display this application version.
--ansi           Force ANSI output.
--no-ansi        Disable ANSI output.
--no-interaction -n Do not ask any interactive question.
--profile        Display timing and memory usage information
  
```

Untuk dapat menggunakan PHP Built in Server, terlebih dahulu buka *command prompt* dan masuk ke direktori masuk ke C:/xampp/htdocs/<nama_projek> misalkan nama proyeknya **vuejs-learn**. Setelah itu ketik sintak **php -S localhost:5555** jika berhasil maka tampilan seperti dibawah ini (penulis menginstall xampp nya di partisi D).



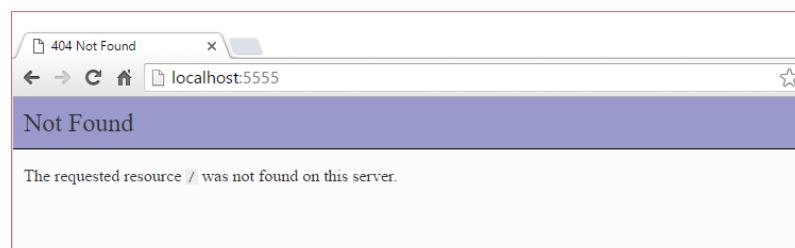
```

Command Prompt - php -S localhost:5555
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\ten>
D:\>cd xampp\htdocs\vuejs-learn

D:\xampp\htdocs\vuejs-learn>php -S localhost:5555
PHP 7.0.4 Development Server started at Thu Jul 14 06:10:02 2016
Listening on http://localhost:5555
Document root is D:\xampp\htdocs\vuejs-learn
Press Ctrl-C to quit.
  
```

Dan jika dijalankan **localhost:5555** maka akan muncul error seperti dibawah ini. Jika error ini sudah muncul, berarti built in server nya sudah berhasil berjalan.



4. Hello World

Seperti bahasa pemrograman lainnya, kita akan awali dengan bagaimana cara menampilkan tulisan “Hello World” menggunakan framework Vue.js. Buat file .html atau .php, kemudian buat template HTML dan panggil file Vue.js. berikut adalah contoh menampilkan Hello World.

Div dengan id “vueElement”

```
<div id="vueElement">
  {{ says }}
</div>
```

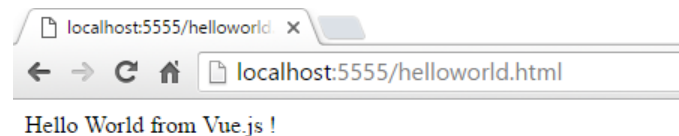
buat file atau taruh script javascript ini di bawah script pemanggilan file Vue.js

object “Obj”

```
<script>

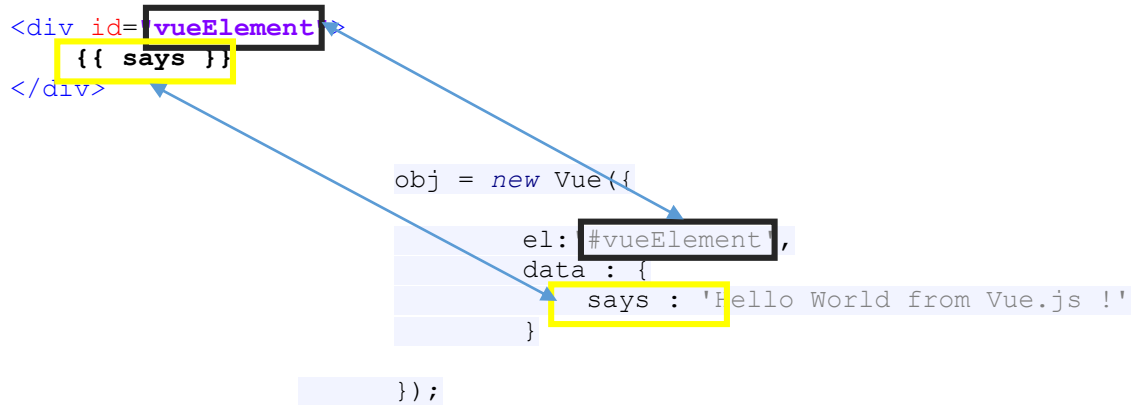
  obj = new Vue({
    el: '#vueElement',
    data : {
      says : 'Hello World from Vue.js !'
    }
  });
</script>
```

Kemudian jika file index.html ini di jalankan di browser maka akan muncul tulisan sebagai berikut :



Penjelasan

Pertama-tama kita tentukan dulu elemen mana yang akan kita gunakan dalam Vue.js. dalam hal ini kita menggunakan elemen div dengan ID “divElement”. Nanti ID “divElement” ini juga kita deklarasikan dalam object Vue.js. sesudah kita deklarasikan element, selanjutnya kita masukan data binding sederhana, dalam contoh diatas kita menggunakan data dengan nama “says”, data ini kita deklarasikan di view menggunakan tanda “{{ }}”, tapi untuk catatan, jika anda menggunakan framework PHP Laravel, maka ditambahkan tanda “@” di awalnya jadi “ @{{ }} “, karena di laravel tanda tersebut juga dipakai di blade sistem nya. Sesudah kita isi data “says”, maka apa yang kita set di object Vue.js maka akan tampil di komponen view nya.



5. Membuat Objek Vue.js

Ketika kita membuat instant menggunakan konstruktor `Vue()`, sangat perlu untuk menspesifikasikan elemen mana yang akan kita gunakan nantinya. Dalam kode diatas, menggunakan `el` untuk menspesifikasin elemen yang akan dimanipulasi menggunakan Vue.js menggunakan CSS Selector.

Deklarasi membuat objek Vue.js

```
vueObj = new Vue({
  ...
});
```

Sebuah instant atau object dari Vue.js menggunakan pola (*pattern*) MV-VM yaitu Model View ViewModel, dimana jika data di model berubah maka data di view juga berubah dan sebaliknya jika data di model berubah maka data di view juga akan berubah.

Itulah salah satu fitur penting jika kita ingin menggunakan framework javascript yaitu *Data Binding*. Dengan Vue.js, mekanisme data binding akan menjadi sangat mudah digunakan. Vue konstruktor dapat di turunkan untuk membuat *custom component* dengan kode :

Vue extended sintak

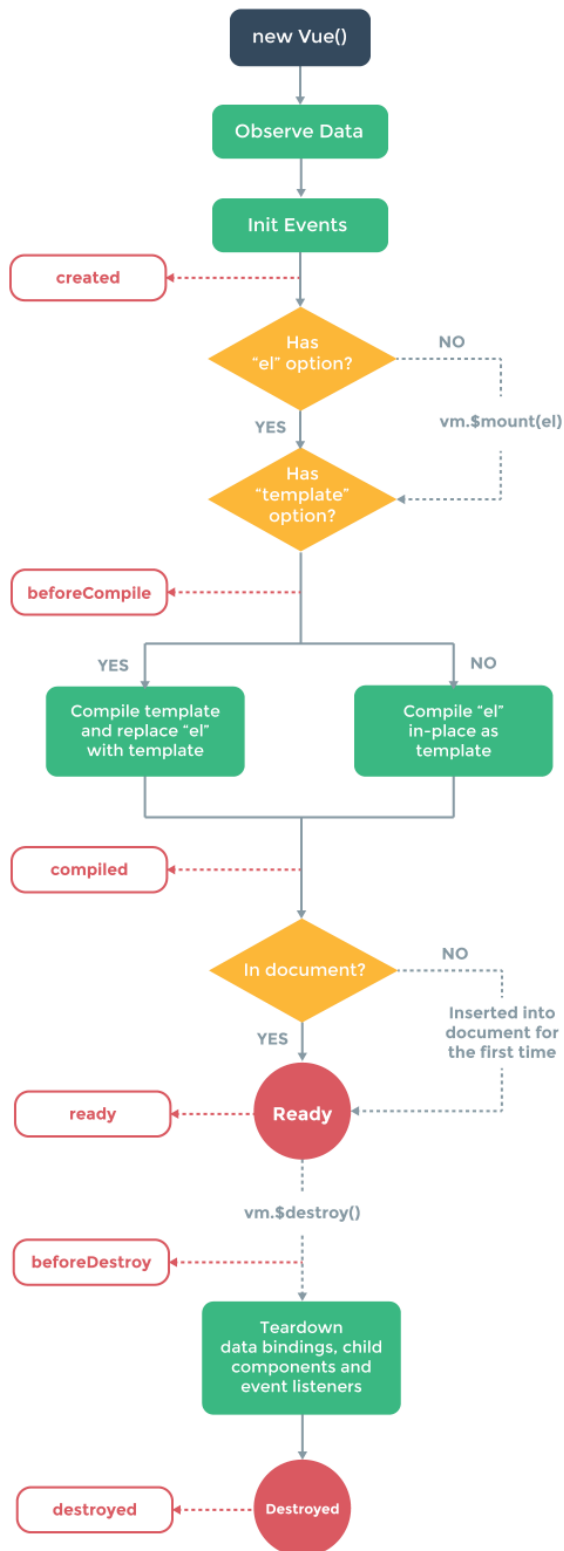
```
var customComponent = Vue.extend({
  ...
})

var myCustomComponent = new customComponent()
```

meskipun kamu membuat proses pewarisan, dalam banyak kasus kamu harus mendaftarkan sebuah konstruktor komponent sebagai sebuah elemen dan template.

Diagram Lifecycle

Di bawah ini adalah diagram dari siklus penciptaan objek Vue.js, kamu tidak perlu untuk memahami semuanya sekarang, tapi untuk kedepannya, kamu dianjurkan untuk memahami diagram dibawah ini.



6. Two Way Data Binding

Data binding merupakan salah satu fitur penting dalam framework javascript khususnya Vue.js. Dengan konsep MVVM, Vue.js memberikan kemudahan dalam memanipulasi data di model dan di view. Untuk lebih memahami konsep ini, kita akan coba langsung membuat kode program dibawah ini.

Buatlah file **databinding.html** dan panggil file Vue.js.

View databinding.html

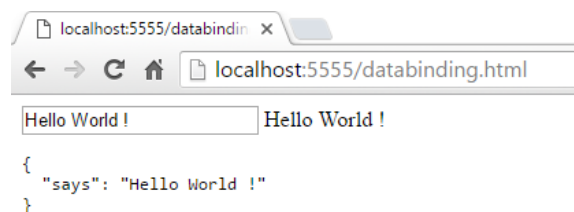
```
<div id="vueElement">
  <input type="text" v-model="says"> {{ says }}

  <pre>{{ $data | json }}</pre>
</div>
```

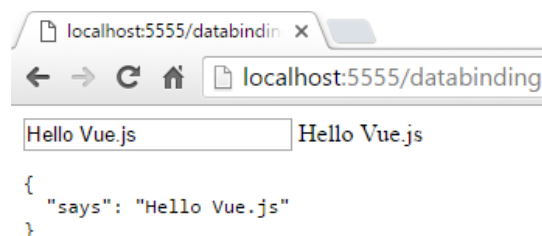
Vue.js objek

```
obj = new Vue({
  el: '#vueElement',
  data : {
    says : 'Hello World !'
  }
});
```

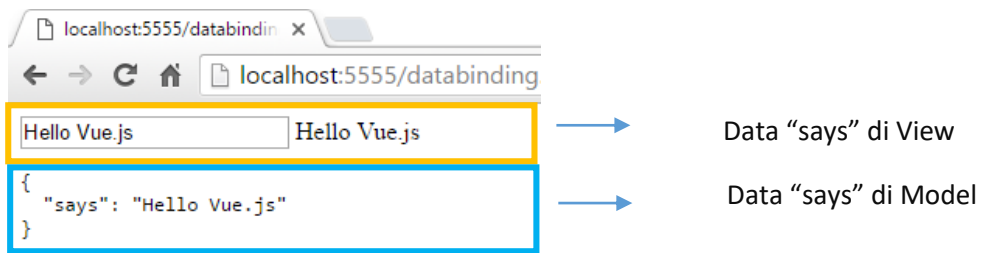
Jalankan **databinding.html** di browser dan akan muncul tulisan Hello World.



Di textbox tersebut, kita coba ganti hello world menjadi “hello Vue.js”, maka tulisan di sebelah kanan dan dibawahnya pun akan otomatis terganti sesuai dengan apa yang kita inputan di textbox tersebut.



Dari contoh diatas, kita dapat melihat apa yang kita inputan tadi, otomatis merubah teks dibagian kanan dan dibawahnya, seperti ini lah contoh penerapan *two way data binding*. Model bisa merubah data ke View, dan sebaliknya View bisa merubah data Ke Model.



Penjelasan

Berikut kita akan coba bedah kode **databinding.html** diatas, dimulai dengan pendeklarasikan elemen div dengan ID **"divElement"** dan begitu pun di pendeklarasian objek Vue.js, **el** nya kita isi dengan **#divElement**.

```
<div id="vueElement">
  ...
</div>
```

```
obj = new Vue({
  el: '#vueElement',
  ...
});
```

Setelah itu kita, deklarasikan data binding dengan nama **"says"** dalam objek Vue.js dan isi dengan nilai **"Hello World !"**.

```
obj = new Vue({
  el: '#vueElement',
  data: {
    says: 'Hello World !'
  }
});
```

Di view **databinding.html**, buat input text dan kita refer ke data model **"says"** menggunakan *directive* **v-model** yang tadi di buat di Model. Disamping itu kita coba deklarasikan output view nya.

```
<input type="text" v-model="says" > {{ says }}
```

Ada banyak *directive* yang akan kita pelajari kedepan, jika ingin melihat *directive* yang lengkap, Anda bisa mengunjungi situs nya langsung dan masuk ke bagian API nya.

Untuk melihat representasi data **"says"** yang ada dimodel, kita coba buat sebuah output data menggunakan format JSON.

```
<pre>{{ $data | json }}</pre>
```

```
{
  "says": "Hello Vue.js"
}
```

7. Computed Properties

Karena di Vue.js ini tidak mempunyai *controller* seperti pada angular, maka alternatif lain yaitu fitur *computed propertie*. Fitur ini akan sangat bermanfaat bila kita membutuhkan operasi logik atau operasi aritmatika ke dalam aplikasi.

Untuk lebih jelasnya, buat file **compprop.html** dan panggil file Vue.js. dalam file ini kita mengambil contoh kasus, dimana nilai yang kita inputkan memiliki rentang antara 1 – 10 maka bernilai C, 11-20 bernilai B, 21-30 bernilai A.

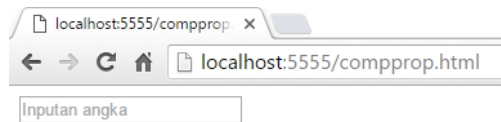
compprop.html

```
<div id="vueElement">
  <input type="text" v-model="poin">
  <h2>Nilai Anda Adalah : {{ nilai }}</h2>
</div>
```

Objek Vue.js

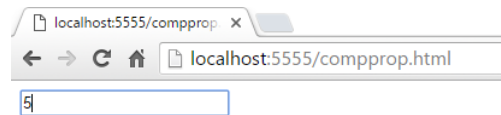
```
obj = new Vue({
  el: '#vueElement',
  data : {
    poin: '',
  },
  computed : {
    nilai: function() {
      if(this.poin >=1 && this.poin <= 10){
        return 'C';
      }else if(this.poin >=11 && this.poin <= 20){
        return 'B';
      }else if(this.poin >=21 && this.poin <= 30){
        return 'A';
      }else{
        return '';
      }
    }
  }
});
```

Jalankan kode **compprop.html** dibrowser, kemudian output nya akan akan sebagai berikut :



Nilai Anda Adalah :

Kemudian kita coba inputan angka 5 pada input teks tersebut, maka Nilainya adalah C



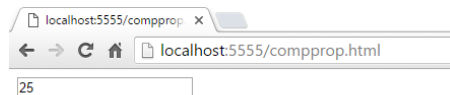
Nilai Anda Adalah : C

Kita coba input angka 15, maka akan keluar nilai B



Nilai Anda Adalah : B

Dan apabila kita coba inputkan nilai 25, maka akan keluar nilai A



Nilai Anda Adalah : A

Penjelasan

Pertama kita buat elemen div dengan ID **divElement** dan panggil di objek Vue.js

```
<div id="vueElement">
  ...
</div>
```

```
obj = new Vue({
  el: '#vueElement',
  ...
});
```


Sesudah itu, karena kita akan menginputkan angka yang maka kita sediakan input teks dengan *directive v-model* yang mengacu ke data binding “**point**” dan inisialisasi nilai point ini dengan nilai kosong “

```
<div id="vueElement">
  <input type="text" v-model="poin">
  ...
</div>
```

```
obj = new Vue({
  el: '#vueElement',
  data : {
    poin: '',
  },
  ...
});
```

Setelah itu, kita coba buat label untuk menghasilkan output nilai dari angka yang telah kita inputkan. Untuk menampilkan output ini, kita akan memanggil fungsi “**nilai**”. Fungsi **nilai** inilah yang akan ditaruh di properti komputasi (*computed*), karena di dalam fungsi ini terdapat operasi logika.

```
<div id="vueElement">
  <input type="text" v-model="poin">
  <h2>Nilai Anda Adalah : {{ nilai }}</h2>
</div>
```

```
obj = new Vue({
  el: '#vueElement',
  data : {
    poin: '',
  },
  computed : {
    nilai: function() {
      ...
    }
  }
});
```

Berdasarkan studi kasus, jika angka yang diinputkan antara 1-10 maka akan muncul nilai C. Jika angka antara 11-20 akan muncul nilai C dan jika angka antara 21-30 maka akan muncul A. Operasi ini kita simpan di method nilai pada *computed properties*.

```

obj = new Vue({
  el: '#vueElement',
  data : {
    poin: '',
  },
  computed : {
    nilai: function() {
      if(this.poin >= 1 && this.poin <= 10){
        return 'C';
      } else if(this.poin >= 11 && this.poin <= 20){
        return 'B';
      } else if(this.poin >= 21 && this.poin <= 30){
        return 'A';
      } else{
        return '';
      }
    }
  }
});

```

8. Operasi Logika

Ada situasi dimana kita ingin ada operasi logika sederhana (*if .. else*) dalam komponen View seperti di bawah ini .

```

{{ #if display }}
<p>Tampilkan kan</p>
{{ /if }}

```

Dalam Vue.js, sudah terdapat fitur untuk menangani kondisi seperti diatas. Terdapat *directive* untuk mengatur operasi logika (*if .. else*) yang bisa disimpan dalam View. Contoh sederhananya seperti dibawah ini.

```

<p v-if="display">Tampilkan</p>

```

Kita akan coba membuat file **conditional1.html** untuk melihat kondisi menggunakan *directive* v-if dan jangan lupa untuk memanggil file Vue.js nya.

conditional1.html

```

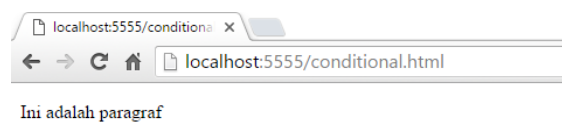
<div id="vueElement">
  <p v-if="display">Ini adalah paragraf</p>
</div>

```

Object Vue.js

```
obj = new Vue({  
  
  el: '#vueElement',  
  
  data : {  
    display: true,  
  },  
  
});
```

Dan jika ditampilkan di browser, maka akan uncul tulisan “Ini adalah paragraf”.



Penjelasan

Dalam *directive v-if* diatas, `<p v-if="display">` kita *data binding Display* yang kita set di object Vue.js dengan tipe boolean yaitu “true”. Jika kita set **display** dengan nilai “false” maka tulisan “Ini adalah paragraf” tidak akan muncul.

Selain *directive v-if*, Vue.js juga menyediakan untuk menangani kondisi yang lainnya yaitu menggunakan *directive v-else*. Untuk mempermudah pemahaman, kita coba membuat file **conditional2.html** dan jangan lupa panggil file Vue.js.

conditional2.html

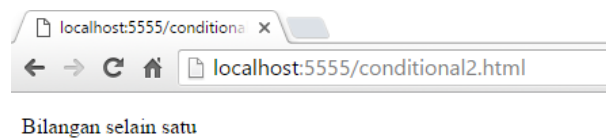
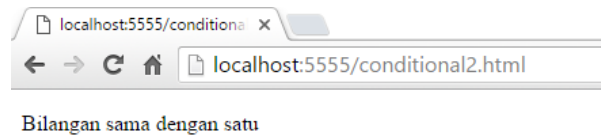
```
<div id="vueElement">  
  <p v-if="display == 1">Bilangan lebih dari satu</p>  
  <p v-else>Bilangan selain satu</p>  
</div>
```

Objek Vue.js

```
obj = new Vue({  
  
  el: '#vueElement',  
  
  data : {  
    display: 1,  
  },  
  
});
```

Penjelasan

Dalam contoh diatas, *data binding* kita set dengan nilai 1, dan dalam file view **conditional2.html** kita atur menggunakan *directive v-if .. v-else*. Dimana jika data display nya berisi 1 maka akan tereksekusi baris `<p v-if="display == 1">Bilangan lebih dari satu</p>` dan jika data angka berisi selain angka 1, maka baris ini `<p v-else>Bilangan selain satu</p>`.



9. List Rendering

Fitur ini menggunakan *directive v-for*, kalau di PHP ada fitur `Foreach()`, *directive v-for* ini mirip dengan fitur `foreach` tersebut. Biasanya untuk mengeluarkan data yang berbentuk array dalam bentuk perulangan.

Untuk lebih jelasnya, kita buat file **listrender1.html** dan jangan lupa panggil file `Vue.js`.

listrender1.html

```
<div id="vueElement">
  <ul>
    <li v-for="agenda in agendas">{{ agenda }}</li>
  </ul>
</div>
```

object vue.js

```
obj = new Vue({
```

```
  el: '#vueElement',
```

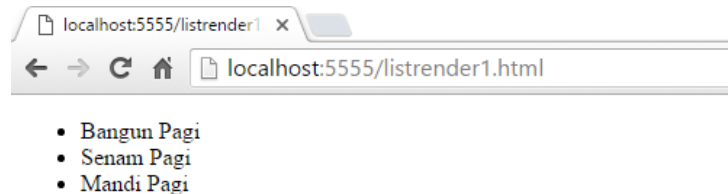
```
  data : {
    agendas: [
      'Bangun Pagi',
      'Senam Pagi',
      'Mandi Pagi'
```

```

    ]
  },
});

```

Dan jika dijalankan di browser, maka akan terlihat list daftar dari nilai *data binding* agendas.



Penjelasan

Karena nilai dari *data binding* nya berulang, maka kita menggunakan elemen `` untuk mengeluarkan nilai-nilai tersebut.

```

data : {
  agendas : [
    'Bangun Pagi',
    'Senam Pagi',
    'Mandi Pagi'
  ]
},

```

Dan dalam elemen yang berulang ini `` kita simpan sisipkan *directive v-for* yang mengacu ke *data binding* agendas.

```

<ul>
  <li v-for="agenda in agendas">{{ agenda }}</li>
</ul>

```

Directive v-for selain untuk mengeluarkan *data binding* yang bertipe *array*, juga dapat mengeluarkan data informasi (*properties*) dari suatu object menggunakan *\$key* dan *value* dari object itu sendiri. Setiap *property* dia mempunyai nilai *\$key* yang berbeda-beda. Untuk lebih jelasnya buat file **listrender2.html** dan panggil file **Vue.js**.

listrender2.html

```

<ul id="vueElement">
  <li v-for="value in mahasiswa">
    {{ $key }} : {{value}}
  </li>
</ul>

```

Object Vue.js

```

obj = new Vue({

```

```

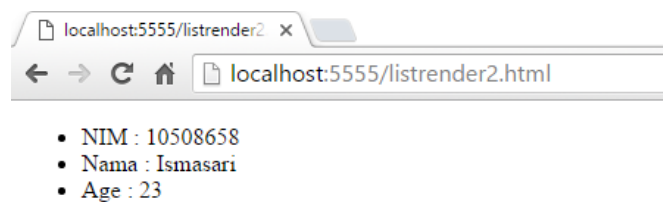
    el: '#vueElement',

    data : {
      mahasiswa: {
        NIM : '10508658',
        Nama : 'Ismasari',
        Age: 23
      }
    },

  });

```

Dan jika kita jalankan di browser, akan muncul keterangan dari object Mahasiswa.



10. Event Handling

Event Handling adalah konsep penanganan suatu aksi yang terjadi. Dalam beberapa kondisi, sering kita membuat aplikasi yang mengharuskan pengguna untuk menekan tombol maka akan terjadi suatu aksi misalnya menampilkan sebuah popup pesan.

Event merupakan respon dari program ketika user melakukan tindakan terhadap element tertentu dalam aplikasi.

Untuk lebih jelasnya, kita akan coba menjelaskan *event handling* dalam sebuah contoh. Contoh yang akan kita ambil yaitu kita coba membuat sebuah button, yang setiap kita tekan button itu, ada label dengan nilai yang terus naik / *increment*.

Buat file **eventHandling.html** dan panggil file **vue.min.js**.

eventHandling.html

```

<div id="eventHandlingID">

  <button type="submit" v-on:click="updateCount">

    Incremental Counter : {{ count }}

  </button>

</div>

```

Objek Vue.js

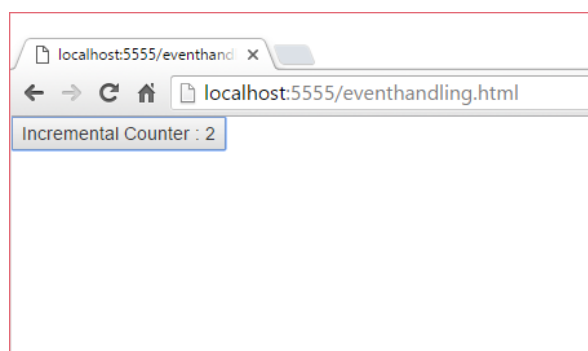
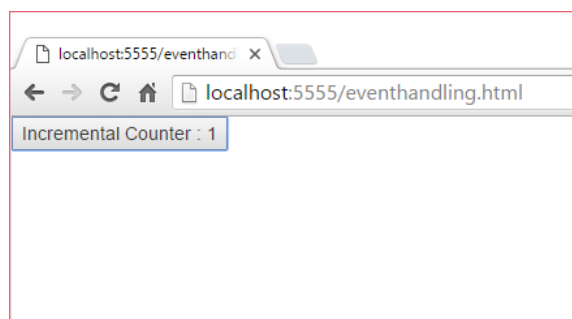
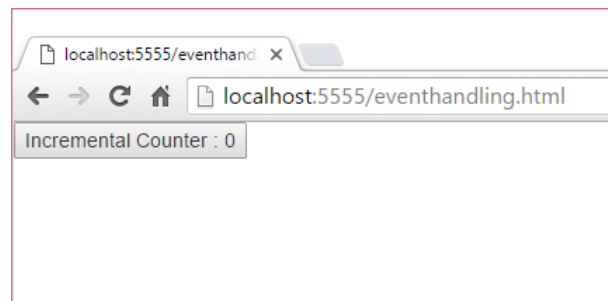
```

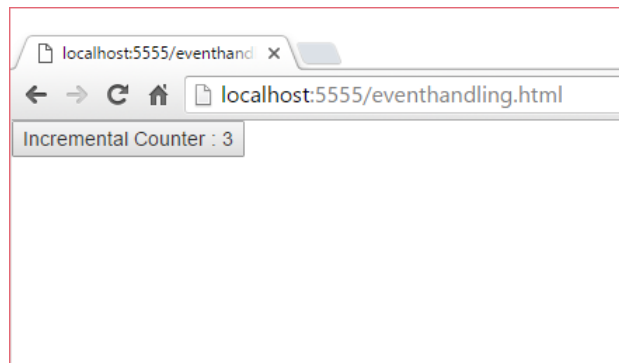
var obj = new Vue({

  el: '#eventHandlingID',

```

```
    data : {  
      count:0  
    },  
  
    methods: {  
  
      //methods untuk update counter  
      updateCount:function() {  
        this.count += 1  
      }  
    }  
  }  
});
```





Buat element **div** dengan ID **eventHandlingID** dan object Vue.js dengan **el eventHandlingID**.

eventHandling.html

```
<div id="eventHandlingID">
  ...
</div>
```

Objek Vue.js

```
var obj = new Vue({
  el: '#eventHandlingID',
  ...
});
```

Setelah itu membuat *data binding* **count** dan membuat methods **updateCount**. Method ini kita gunakan untuk mengupdate *data binding* secara *increment* 1. Semua methods dalam Vue.js harus berada pada lingkup (*scope*) **methods:{ ... }**.

Objek Vue.js

```
var obj = new Vue({
  el: '#eventHandlingID',
  data : {
    count: 0
  },
  methods: {
    //methods untuk update counter
    updateCount: function() {
      this.count += 1
    }
  }
});
```


Di View, kita buat `<button>` dengan *directive* **`v-on:click`** yang mengacu ke methods **`updateCount`**. Di dalam `<button>` ini, lalu buat label untuk menampilkan nilai counter dari *data binding* `count`.

```
<div id="eventHandlingID">

  <button type="submit" v-on:click="updateCount">

    Incremental Counter : {{ count }}

  </button>

</div>
```

11. Bekerja dengan Components

Components adalah salah satu fitur penting dalam framework Vue.js, bisa dibilang fitur ini memungkinkan *programmer* memperluas elemen HTML dan membungkus (*encapsulation*) dalam suatu kode.

a. Registration

Untuk menggunakan fitur ini, kita diharuskan untuk mendaftarkan dulu *component* apa yang akan kita gunakan. Terdapat dua cara untuk mendaftarkan *component* yang akan kita buat. Untuk lebih jelasnya perhatikan kedua langkah ini.

- Cara Pertama

Buat file **`customcomp1.html`** dan panggil file `vue.min.js`. dan ketikan kode berikut.

`customcomp1.html`

```
<div id="learnComponent">
  <my-custom-component></my-custom-component>
</div>
```

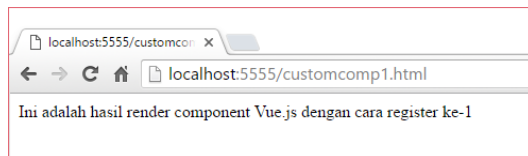
Script js

```
// define
var myCustomComponent = Vue.extend({
  template: '<div>Ini adalah hasil render component Vue.js dengan  
cara register ke-1</div>'
});

// register
Vue.component('my-custom-component', myCustomComponent)

// object
new Vue({
  el: '#learnComponent'
})
```

Dan jika kita jalankan di browser, maka akan muncul tulisan dibawah ini.



Pertama kita membuat object dari **Vue.extend({ ... })** dan menentukan template dari *component* yang akan kita buat.

```
// define
var myCustomComponent = Vue.extend({
  template: '<div> Ini adalah hasil render component Vue.js dengan
cara register ke-1</div>'
});
```

Setelah itu, dari *object* yang sudah dibuat diatas kemudian kita daftarkan menggunakan **Vue.component({ ... })** dan menentukan nama dari *component* yang akan kita buat. (seolah-olah kita membuat elemen baru di HTML).

```
// register
Vue.component('my-custom-component', myCustomComponent)
```

Setelah proses registrasi selesai, selanjutnya membuat *root instance Vue.js*.

```
new Vue({
  el: '#learnComponent'
})
```

- Cara Kedua
Dalam cara kedua ini, kita akan coba menyederankan proses registrasi yang pertama. Buat file **customcomp2.html** dan panggil file **vue.min.js**. dan ketikkan kode berikut.

customcomp1.html

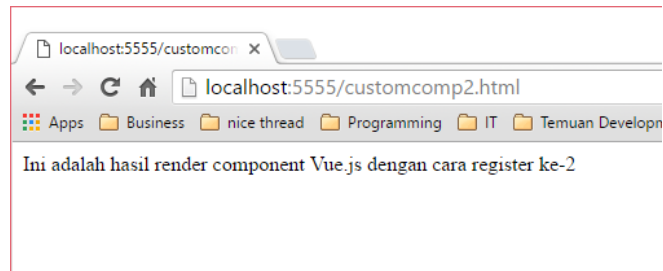
```
<div id="learnComponent">
  <my-custom-component></my-custom-component>
</div>
```

Script js

```
// define
Vue.component('my-custom-component', {
  template: '<div> Ini adalah hasil render component Vue.js
dengan cara register ke-2</div>'
});

// object
new Vue({
  el: '#learnComponent'
});
```

Dan jika dijalankan di browser, maka output akan sebagai berikut.



Dalam cara yang kedua ini, kita menyederhanakan proses registrasi menggunakan **Vue.component({ ... })** dan langsung mendefinisikan nama component dan template nya.

```
// define
Vue.component('my-custom-component', {
  template: '<div>Ini adalah hasil render component Vue.js
dengan cara register ke-2</div>' });
```

b. Components Props

Setiap object dari *components* berada pada lingkung nya masing-masing, itu artinya kamu tidak bisa secara langsung me-referensi data yang ada di *parent* dalam sebuah template milik *child*-nya. Data dapat dilewatkan ke *child component* menggunakan **Props**.

Untuk lebih jelas dalam mengenal apa itu **props**, kita buat file html **customcompProp.html** dan panggil file **vue.min.js**. Dalam contoh ini kita menggunakan framework css bootstrap 3 (**bootstrap.min.css**) menggunakan CDN.

customcompProp.html

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
">

<div id="learnComponentProp" class="container col-md-4
style="padding-top:10px;">

  <alert type="success" bold="Success" msg="Alert ini untuk pesan sukses"></alert>
  <alert type="warning" bold="Warning." msg="Alert ini untuk peringatan."></alert>
  <alert type="danger" bold="Danger" msg="Pesan ini untuk peringatan keras"></alert>
  <alert type="info" bold="Info." msg="Alert ini untuk menampilkan pesan info."></alert>

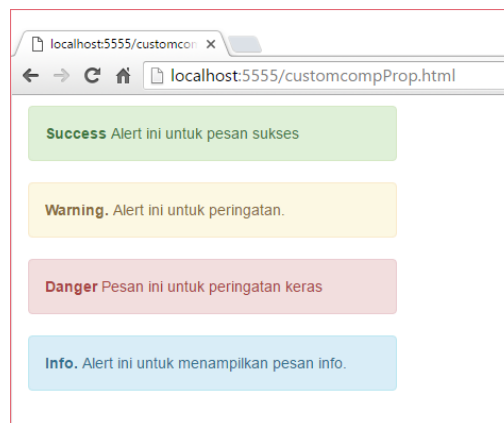
</div>
```

Vue.js object

```
// define
Vue.component('alert', {
  props: ['type', 'bold', 'msg'],
  template: '<div class="alert alert-{{ type }}" role="alert"><b>{{
bold }}</b> {{ msg }}</div>'
});
```

```
// object
new Vue({
  el: '#learnComponentProp'
});
```

Dan jika kita jalankan di browser, maka akan muncul tipe-tipe alert.



Pada contoh diatas, kita membuat *component* dengan nama **alert** yang nantinya akan dikombinasikan dengan bootstrap css.

```
Vue.component('alert', {
  ...
});
```

```
<alert type="success" bold="Success" msg="Alert ini untuk pesan sukses"></alert>
<alert type="warning" bold="Warning." msg="Alert ini untuk peringatan."></alert>
<alert type="danger" bold="Danger" msg="Pesan ini untuk peringatan keras"></alert>
<alert type="info" bold="Info." msg="Alert ini untuk menampilkan pesan info."></alert>
```

Setelah itu, kita buat **props** untuk *component alert*. Dalam contoh, kita membuat tiga yaitu **type**, **bold**, dan **msg**.

```
Vue.component('alert', {
  props: ['type', 'bold', 'msg'],
  ...
});
```

Setelah mendefinisikan **props** nya, kita coba buat template untuk kita masukan tiap-tiap **props**nya.

```
Vue.component('alert', {
  props: ['type', 'bold', 'msg'],
```

```
template: '<div class="alert alert-{{ type }}" role="alert"><b>{{  
bold }}</b> {{ msg }}</div>'
```

```
});
```

Untuk memasukan data **props** nya di template, gunakan tanda `{{ ... }}`. Untuk *component* nya sudah beres sekarang buat instance vue.js nya.

```
// object  
new Vue({  
  el: '#learnComponentProp'  
});
```

Untuk pengaturan object Vue.js sudah sekarang membuat file view untuk menampilkan dalam bentuk HTML nya. Dalam elemen HTML inilah, **props** yang sudah dibuat masing-masing diberi nilai.

12. Filter

Dalam kumpulan data, kita biasanya mengurutkan data dari Abjad A sampai Z misalnya, atau mengelompokkan data mahasiswa yang mempunyai nilai diatas 3.5, atau mencari data nama mahasiswa yang namanya diawali dengan inisial kata A.

Dalam Vue.js, *filter* adalah sebuah fungsi yang menerima nilai / *value*, dan memfilter sesuai kebutuhan. Untuk *filter ini*, Vue.js sudah menyediakan fungsi-fungsi bawaannya seperti **capitalize**, **uppercase**, **lowercase**, **currency**, **pluralize**, **json**, **key**, **filterBy** dan **orderBy**. Tapi disini, penulis akan memberikan contoh filter menggunakan fungsi bawaan yaitu **uppercase** yang berfungsi untuk membentuk hurup kapital.

Buat file **basicFilter.html** dan panggil file vue.min.js

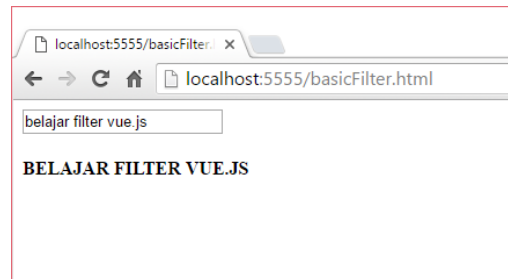
basicFilter.html

```
<div id="filterUppercase">  
  <input type="text" v-model="message" />  
  <h4>{{ message | uppercase }}</h4>  
</div>
```

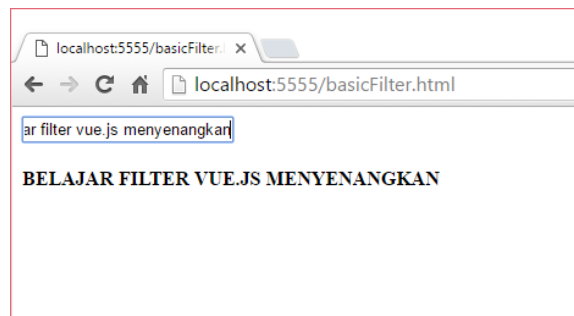
Objek vue.js

```
var obj = new Vue({  
  el: '#filterUppercase',  
  data: {  
    message: 'belajar filter vue.js'  
  }  
});
```

Dan jika kita jalankan di browser, hasilnya akan seperti dibawah ini.



Dan jika kita menambahkan kata di textbox diatas, maka otomatis kata yang dibawahnya juga akan bertambah dan berbentuk kapital.



Pertama kita buat div dengan ID “filterUppercase” dan buat objek Vue.js dengan el mengacu ke div “filterUppercase”.

```
<div id="filterUppercase">
  ...
</div>
```

```
var obj = new Vue({
  el: '#filterUppercase',
  ...
});
```

Kemudian kita buat *data binding* “message” dengan isi “belajar filter vue.js” dan *data binding* ini kita tempatkan juga di textbox dalam div ID “filterUppercase”.

```
<div id="filterUppercase">
  <input type="text" v-model="message" />
  ...
</div>
```

```
var obj = new Vue({
  el: '#filterUppercase',
  data: {
    message: 'belajar filter vue.js'
  }
});
```

Sesudah membuat *data binding*, selanjutnya yaitu kita coba menampilkan *data binding* tersebut dengan memberikan filter **uppercase**. Untuk menggunakan filter yaitu sesudah kita mendefinisikan *data binding* di View juga ditambahkan simbol ' | ' dan fungsi filter yang akan dipakai, dalam hal ini kita menggunakan fungsi **uppercase**.

```
<div id="filterUppercase">
  <input type="text" v-model="message" />
  <h4>{{ message | uppercase }}</h4>
</div>
```

13. Custom Filter

Disamping menyediakan fungsi filter bawaan, Vue.js juga memungkinkan developer untuk membuat filter yang sesuai dengan kebutuhan masing-masing. Salah satunya dalam objek Vue.js. Untuk lebih jelasnya mari kita buat file **customFilter.html** dan jangan lupa panggil file Vue.js nya.

customFilter.html

```
<div id="customVilter">
  <h2>{{ kata | upper }}</h2>
  <h2>{{ kata | lower }}</h2>
</div>
```

Objek Vue.js

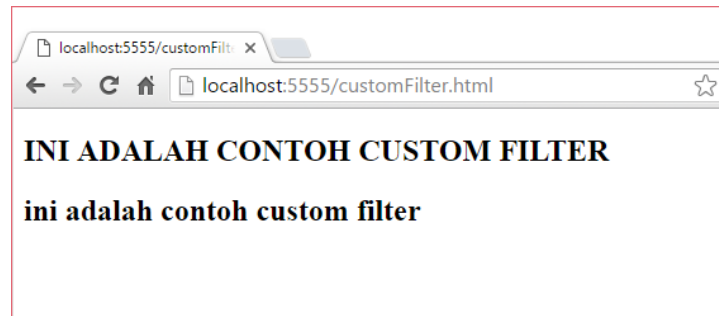
```
var obj = new Vue({
  el: '#customVilter',
  data : {
    kata: 'Ini adalah contoh custom filter'
  },
  filters : {
    //filter untuk upper
    upper: function(value) {
      return value.toUpperCase();
    },
    //filter untuk fungsi lower
    lower: function(value) {
      return value.toLowerCase();
    }
  }
});
```

```

    }
  });

```

Dan jika dijalankan di browser maka akan muncul dua tulisan. Yang satu menggunakan *uppercase* dan yang satu menggunakan *lowercase*.



Penjelasan

Pertama-tama kita membuat div dengan ID **customFilter** dalam ID ini kita akan memanggil data binding “kata” yang menggunakan dua filter yaitu *upper* dan *lower*.

```

<div id="customVilter">
  <h2>{{ kata | upper }}</h2>
  <h2>{{ kata | lower }}</h2>
</div>

```

Selanjutnya, membuat objek Vue.js dengan el **customFilter**. Dan data binding “kata” dengan isi “Ini adalah contoh custom filter”.

```

var obj = new Vue({
  el: '#customVilter',
  data : {
    kata: 'Ini adalah contoh custom filter'
  },
});

```

Sekarang dalam objek Vue.js diatas, kita deklarasikan filte yang akan kita buat di dalam *scope filters: { ... }*. Untuk masing-masing filter *upper* dan *lower* kita bentuk menjadi sebuah fungsi yang memanipulasi data menjadi dalam bentuk *uppercase* dan *lowercase*.

```

filters : {
  //filter untuk upper
  upper: function(value) {
    return value.toUpperCase();
  },
  //filter untuk fungsi lower
  lower: function(value) {

```



```
        return value.toLowerCase();
    }
}
```

14. Demo Simple CRUD dengan Vue.js

Dalam bab terakhir ini, kita akan coba membuat operasi *Cread, Retrieve, Update* dan *Delete* atau yang lebih dikenal dengan istilah CRUD. Dalam kasus ini, data disimpan dalam bentuk array. Pertama kita buat file **basicCrud.html** dengan id element body nya yaitu “**belajar**”, dalam file ini kita menggunakan CDN untuk memanggil file asset CSS (bootstrap.min.css) dan JS (jquery, bootstrap.min.js, vue.min.js)

basicCrud.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Belajar Basic CRUD dengan Vue.js</title>

  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
">

</head>
<body id="belajar">

<tempat kita menulis container dan modal>

<script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"><
/script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/vue/1.0.26/vue.min.js"></script>
>
<script src="crud.js"></script>
</body>
</html>
```

Dalam elemen body file html diatas, buat container dengan isi button dan tabel. Tabel untuk memunculkan modal dialog untuk tambah barang dan edit barang. Tabel untuk melihat data yang sudah masuk.

basicCrud.html - container

```
<div class="container">
  <div class="row">
    <div class="col-lg-12">
      <p></p>

      <button @click="tambahBarang" type="button" class="btn btn-primary"
data-toggle="modal" data-target="#modal">
        Tambah Barang
      </button>
```

```

<p></p>

<table class="table table-striped table-bordered table-hover">
  <thead>
    <tr>
      <th class="text-center">ID Barang</th>
      <th class="text-center">Nama Barang</th>
      <th class="text-center">Jenis Barang</th>
      <th class="text-center">Tanggal Kadaluarsa</th>
      <th class="text-center">Aksi</th>
    </tr>
  </thead>
  <tbody>
    <tr v-for="b in dataBarang">
      <td>{{ b.idBarang }}</td>
      <td>{{ b.namaBarang }}</td>
      <td>{{ b.jenisBarang }}</td>
      <td>{{ b.tanggalKadaluarsa }}</td>
      <td class="text-center">
        <button type="button" class="btn btn-success" data-
toggle="modal" data-target="#modal" @click="editBarang(b)">
          <i class="glyphicon glyphicon-pencil"></i>
        </button>
        <button type="button" class="btn btn-danger"
@click="hapusBarang(b)">
          <i class="glyphicon glyphicon-trash"></i>
        </button>
      </td>
    </tr>
  </tbody>
</table>
</div>
</div>
</div>

```

basicCrud.html - modal

```

<!-- Modal -->
<div class="modal fade" id="modal" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel">
  <div class="modal-dialog" role="document">
    <div class="modal-content">

      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
        <h4 v-if="enable" class="modal-title">Tambah Barang</h4>
        <h4 v-else="enable" class="modal-title">Edit Barang</h4>
      </div>

      <div class="modal-body">
        <form v-if="enable">

          <div class="form-group">
            <label>Nama Barang</label>
            <input type="text" class="form-control" placeholder="Masukkan
Nama Barang" v-model="inputDataBarang.namaBarang"/>
          </div>

```

```

        <div class="form-group">
            <label>Jenis Barang</label>
            <select class="form-control" v-
model="inputDataBarang.jenisBarang">
                <option value="" disabled>Pilih Jenis Barang</option>
                <option value="gas">Gas</option>
                <option value="padat">Padat</option>
                <option value="cair">Cair</option>
            </select>
        </div>

        <div class="form-group">
            <label>Tanggal Kadaluarsa</label>
            <input type="date" class="form-control" placeholder="Masukkan
Tanggal Kadaluarsa" v-model="inputDataBarang.tanggalKadaluarsa"/>
        </div>
    </form>

    <form v-else="enable">

        <div class="form-group">
            <label>ID Barang</label>
            <input type="text" class="form-control" v-
model="inputDataBarang.idBarang" disabled/>
        </div>

        <div class="form-group">
            <label>Nama Barang</label>
            <input type="text" class="form-control" placeholder="Masukkan
Nama Barang" v-model="inputDataBarang.namaBarang"/>
        </div>

        <div class="form-group">
            <label>Jenis Barang</label>
            <select class="form-control" placeholder="Pilih Jenis Barang"
v-model="inputDataBarang.jenisBarang">
                <option value="" disabled>Pilih Jenis Barang</option>
                <option value="gas">Gas</option>
                <option value="padat">Padat</option>
                <option value="cair">Cair</option>
            </select>
        </div>

        <div class="form-group">
            <label>Tanggal Kadaluarsa</label>
            <input type="date" class="form-control" placeholder="Masukkan
Tanggal Kadaluarsa" v-model="inputDataBarang.tanggalKadaluarsa"/>
        </div>
    </form>

    <div class="modal-footer">
        <button type="button" class="btn btn-warning" data-
dismiss="modal">Batal</button>
        <button v-if="enable" type="button" class="btn btn-primary" data-
dismiss="modal" @click="simpanBarang(inputDataBarang)">Simpan</button>
        <button v-else="enable" type="button" class="btn btn-success" data-
dismiss="modal" @click="updateBarang(inputDataBarang)">Update</button>
    </div>
</div>
</div>

```

</div>

Setelah selesai membuat file **basicCrud.html** selanjutnya membuat object Vue.js. kita simpan dalam file **crud.js** sehingga terpisah penulisannya dengan file html nya.

crud.js

```
'use strict';

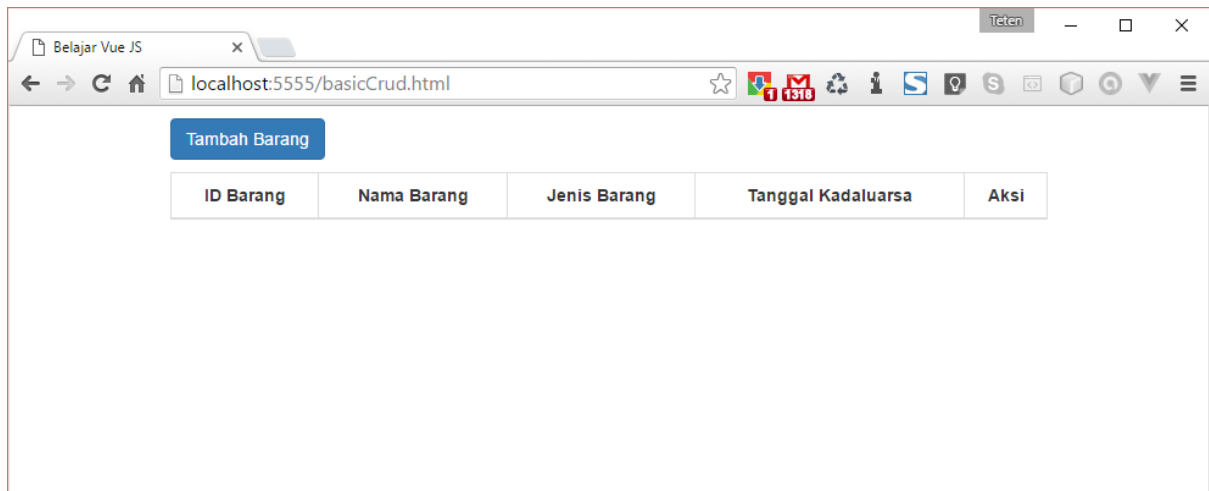
new Vue({
  el: '#belajar',
  data: function() {
    return {
      dataBarang: [],
      inputDataBarang: {},
      enable: false
    }
  },
  methods: {
    generateUUID: function() {
      var d = new Date().getTime();
      if (window.performance && typeof window.performance.now ===
"function") {
        d += performance.now();
      }
      var uuid = 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g,
function(c) {
        var r = (d + Math.random() * 16) % 16 | 0;
        d = Math.floor(d / 16);
        return (c == 'x' ? r : (r & 0x3 | 0x8)).toString(16);
      });
      return uuid;
    },
    tambahBarang: function() {
      this.enable = true;
      this.inputDataBarang = {};
    },
    simpanBarang: function(barang) {
      this.dataBarang.push({
        'idBarang': this.generateUUID(),
        'namaBarang': barang.namaBarang,
        'jenisBarang': barang.jenisBarang,
        'tanggalKadaluarsa': barang.tanggalKadaluarsa
      });
    },
    editBarang: function(barang) {
      this.enable = false;
      this.index = this.dataBarang.indexOf(barang);
      this.inputDataBarang.idBarang = barang.idBarang;
      this.inputDataBarang.namaBarang = barang.namaBarang;
      this.inputDataBarang.jenisBarang = barang.jenisBarang;
      this.inputDataBarang.tanggalKadaluarsa = barang.tanggalKadaluarsa;
    },
    updateBarang: function(barang) {
      this.dataBarang[this.index].idBarang = barang.idBarang;
      this.dataBarang[this.index].namaBarang = barang.namaBarang;
      this.dataBarang[this.index].jenisBarang = barang.jenisBarang;
      this.dataBarang[this.index].tanggalKadaluarsa =
barang.tanggalKadaluarsa;
      this.inputDataBarang = {};
    },
  },
}
```

```

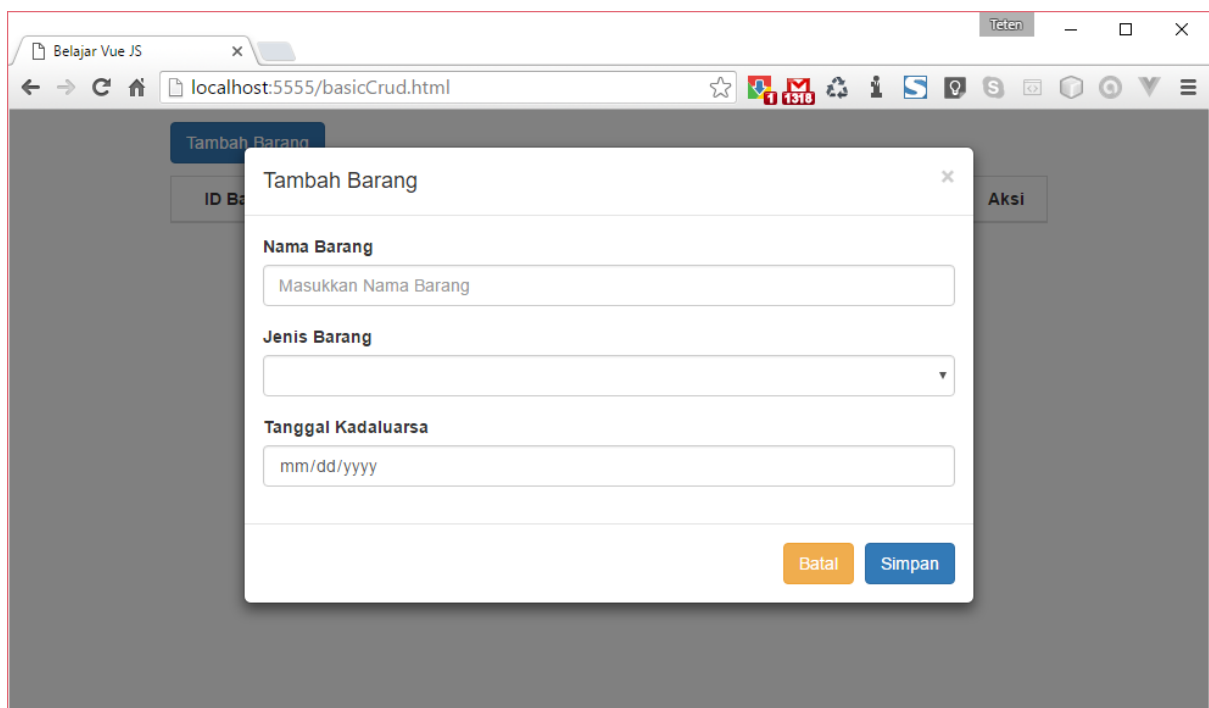
hapusBarang: function(barang) {
  var result = confirm('Anda ingin menghapus data barang ?');
  if (result) {
    this.index = this.dataBarang.indexOf(barang);
    this.dataBarang.splice(this.index, 1);
  }
}
}
});

```

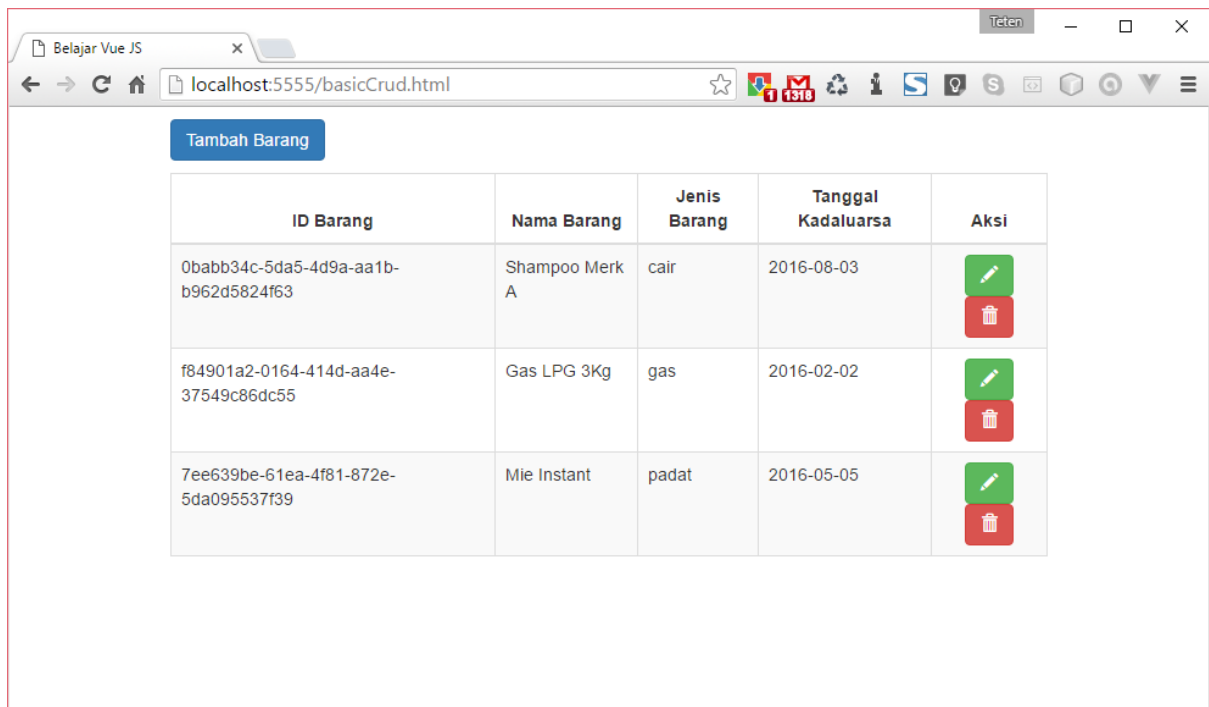
Jalankan file **basicCrud.html** di browser, maka tampilan pertama akan sebagai berikut.



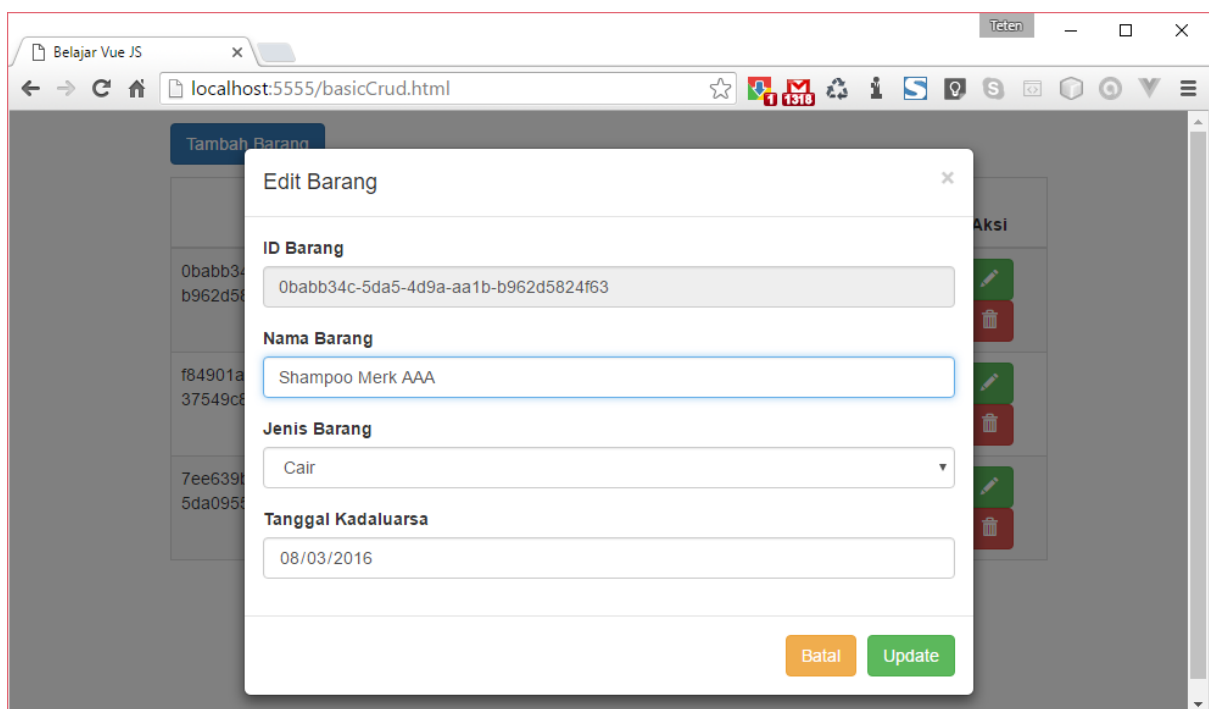
Data diatas masih kosong sehingga data dalam tabel diatas tidak menampilkan data apa-apa. Untuk dapat menambah data, klik tombol 'Tombol Barang' dan akan muncul modal dialog yng berisi form untuk mengisi data barang. Sesudah kita memasukan data kemudian klik tombol 'Simpan'.



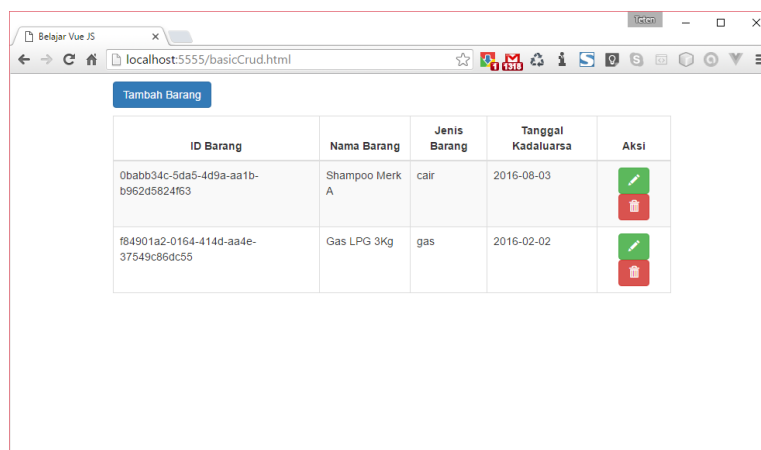
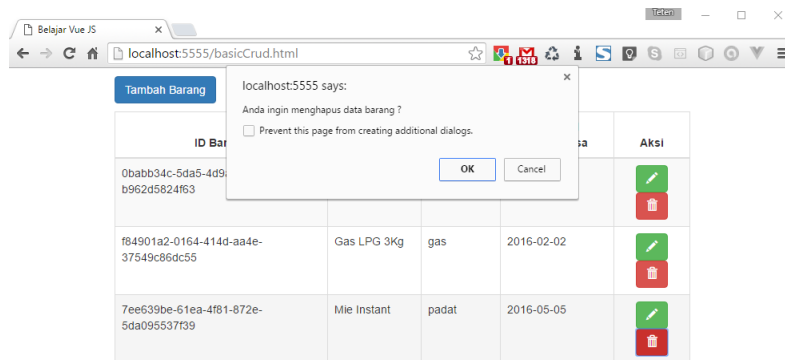
Pada proses penyimpanan data, di object Vue.js ini kita sudah sediakan generasi UUID atau kode unik yang menyediakan kode istimewa disetiap datanya.



Jika ada data yang akan diedit, klik tombol warna hijau dsan akan muncul kotak dialog untuk mengedit data yang sudah masuk dan klik tombol 'Update' untuk menyimpan perubahan.



Dalam data yang masuk, kita coba menghapus data dengan cara klik tombol warna merah dan akan muncul konfirmasi untuk menghapus data.



Daftar Pustaka

<https://vuejs.org/>

<https://coligo.io/>

<http://vegibit.com/>

<https://rizkimufrizal.github.io/belajar-vue-js/>

<https://laracasts.com/series/learning-vue-step-by-step>