



Hafid Mukhlisin
Muhammad Azamuddin

2.5

Vue.js

The Progressive JavaScript Framework

Daftar Isi

- [Daftar Isi](#)
- [Mengenal Vue](#)
 - [Apa itu Vue?](#)
 - [Sejarah Vue](#)
 - [Mengapa Memilih Vue?](#)
 - [Framework Javascript Populer](#)
 - [Didukung Banyak Pustaka](#)
 - [Bukan One Man Show](#)
 - [Digunakan Perusahaan Besar](#)
 - [Mudah Dipelajari](#)
 - [Mudah Diintegrasikan dengan Pustaka Lain](#)
 - [Dukungan Official untuk Pengembangan Aplikasi Enterprise](#)
 - [Fitur Utama](#)
 - [Virtual DOM](#)
 - [Component Base](#)
 - [Template](#)
 - [Modularity](#)
 - [Reactivity](#)
 - [Routing](#)
 - [State Management](#)
 - [Development Tools](#)
 - [Instalasi & Konfigurasi](#)
 - [Hello World](#)
 - [Menguji Reaktifitas](#)
 - [Kesimpulan](#)
- [Dasar-Dasar Vue](#)
 - [Objek Vue](#)
 - [Inisiasi Objek Vue](#)
 - [Properti el](#)
 - [Properti Data](#)
 - [Siklus Objek Vue](#)
 - [create](#)
 - [mount](#)
 - [update](#)
 - [destroy](#)
 - [Penulisan Template](#)
 - [Data Teks](#)
 - [Data Raw HTML](#)
 - [Data Attribute](#)
 - [JavaScript Expression](#)
 - [Properti Template](#)
 - [Properti Methods, Computed, & Filters](#)
 - [Properti Methods](#)

- Properti Computed
 - Properti Filters
 - Argumen Pada Filters
 - Chaining Filters
 - Deklarasi Filters Secara Terpisah
 - Kesimpulan
- Directive
 - Mengenal Directive
 - v-html
 - v-once
 - v-text
 - v-show
 - v-if
 - v-on
 - v-bind
 - Kesimpulan
- List
 - Menampilkan Data Array
 - v-for Menggunakan Tag Template
 - v-for Menggunakan Index
 - Menampilkan Data Objek
 - Menampilkan Data Collection
 - Atribut Key
 - Membatasi v-for menggunakan v-if
 - Perubahan (mutation) Data Pada Array
 - push() & pop()
 - unshift() & shift()
 - sort() & reverse()
 - splice()
 - fungsi set pada Vue
 - Perubahan Data Pada Objek
 - Kesimpulan
- Form
 - Input Binding
 - Text
 - Boolean
 - Array
 - Filtering Data List
 - Handling Submit Form & Validation
 - Validasi Data
 - Prepare Data Submit
 - Send Data To Server
 - Handling File Upload
 - Kesimpulan
- Component
 - *Component* Dasar

- Component Naming
 - Component Registration
 - Global Component
 - Local Component
 - Deklarasi Properti Data
 - Reusable Component
- Component Lanjutan
 - Passing Data To Component
 - Directive Pada Component
 - Update Data Parent From Component
 - Two Way Data Binding on Component
 - Content Distribution with Slots
 - Single File Component
 - Dynamic Components
- Transition Effect
- Mixins
- Plugins
 - Deklarasi Plugins
 - Menggunakan Plugin
- Kesimpulan
- Routing
 - Features
 - Installation
 - Getting Started
 - Dynamic Routing
 - Component BooksComponent
 - Component BookComponent
 - Programmatic Navigation
 - Penamaan Routes
 - Passing Props To Route Component
 - Transitions Effect
 - Navigation Guards
 - Global
 - Per Route
 - Dalam Component
 - Prevent Leave Accident
 - Authentication Route
 - Kesimpulan
- State Management
 - Mengenal State Management
 - Pustaka State Management
 - Instalasi
 - Dev Tools
 - Getting Started
 - Mengakses Store Via Component
 - Getters

- Mutations
- Actions
 - Asynchronous Actions
- Menangani Two Way Data Binding
- Kesimpulan
- Scaffolding Application
 - Briefing Projek
 - Fitur Utama Aplikasi
 - Unified Model Language
 - Use Case Diagram
 - Activity Diagram
 - Class Diagram
 - Desain Database
 - User Interface Specification
 - Tampilan Awal
 - Tampilan Pencarian Buku
 - Tampilan Detail Buku
 - Tampilan Keranjang Belanja
 - Tampilan Login
 - Tampilan Register
 - Tampilan Checkout
 - Preparing Project
 - Command Line Tools
 - Package Manager for Javascript
 - Instalasi NodeJS & NPM
 - Instalasi Vue melalui NPM
 - Apa itu Bundler
 - Instalasi Vue menggunakan Javascript Bundler
 - Browserify
 - Webpack
 - Single File Component
 - Web Server Development
 - Hot Reload
 - Vue Command Line Interface (CLI)
 - Create New Project
 - Create New Project on Web Base
 - Menambahkan Plugin Baru
 - Plugin Axios
 - Plugin Vuetify
 - Kesimpulan
 - Referensi
 - Mengenal Web Service
 - Definisi Web Service
 - Standard Web Service
 - Method Web Service
 - Cara Kerja Web Service

- HTTP Response Code
- Stateless pada Web Service
- Persiapan Tools Pengembangan
 - Bahasa Pemrograman: PHP
 - Database Server: MySQL, MariaDB
 - Web Server: Nginx, Apache
 - Git
 - Package Tools
 - Docker
 - Instalasi Docker
 - Instalasi Laradock
 - Menjalankan Container
 - Mengatasi Bug Pada MySQL
 - Masuk Ke Workspace Container
 - Uji Coba Membuat Projek Baru
 - XAMPP
 - Homestead
 - Composer
 - Instalasi Composer
 - Postman
 - Penggunaan
 - Generate Dokumentasi
- Laravel
 - Mengetahui Laravel
 - Instalasi
 - Konfigurasi
 - Variabel Konfigurasi
 - Virtual Domain & Pretty URL
 - Struktur Direktori Aplikasi
 - Routing
 - Routing Web
 - Routing API (Web Service)
 - HTTP Verbs Method
 - Routing Parameter
 - Routing Redirect
 - Routing Name
 - Routing Group
 - Route Prefixes
 - Route Sub-Domain
 - Controller
 - Middleware
 - Rate Limiting
 - CORS
 - Multiple Middleware
 - Database
 - Konfigurasi

- Migration
 - Generate Migration
 - Tabel Books
 - Tabel Users
 - Tabel Categories
 - Tabel Book Category
 - Tabel Orders
 - Tabel Book Order
 - Tabel Provinces
 - Tabel Cities
 - Execute Migration
- Seeding
 - Generate Seeder
 - Tabel Users
 - Tabel Books
 - Tabel Categories
 - Tabel Provinces & Cities
 - Register Seeder
 - Execute Seeder
- Interact with Database
 - Raw Query
 - Query Builder
 - ORM Eloquent
 - Membuat Model Eloquent
 - Konvensi Model Eloquent
 - Query Pada Model Eloquent
 - Batch Insert Pada Model Eloquent
 - Soft Delete
 - Pagination
- Resources
- Authentication
 - Token Field
 - JWT Token
 - OAuth2
- Endpoint Project Studi Kasus
 - Endpoint Books
 - Endpoint Categories
 - Endpoint Users
 - Endpoint Orders
- Kesimpulan
- Layout Aplikasi
- Halaman Home / Utama
 - Kode Backend
 - Kode Frontend
- Halaman Kategori Buku
 - Kode Backend

- Kode Frontend
- Halaman Buku
 - Kode Backend
 - Kode Frontend
- Halaman Detail Buku
 - Kode Backend
 - Kode Frontend
- Halaman Pencarian Buku
 - Kode Backend
 - Kode Frontend
- Halaman Keranjang Belanja
 - Kode Backend
 - Kode Frontend
- Halaman Formulir Pengiriman Buku
- Halaman Konfirmasi Belanja
 - Kode Backend
 - Kode Frontend
- Halaman Login
- Halaman Register
- Halaman Profile
- Halaman Histori Belanja
- Halaman Detail Belanja
- Kesimpulan

List

Data dalam bentuk list atau daftar yang bisa berupa array , objek atau collection (array dari objek) bisa kita tampilkan dengan mudah menggunakan Vue.

Menampilkan Data Array

Sebagai contoh, misalnya kita mempunyai data judul buku dalam bentuk array.

```
books : [  
  'C++ High Performance',  
  'Mastering Linux Security and Hardening', 'Python Programming  
Blueprints',  
  'Mastering PostgreSQL 10'  
]
```

Atau jika kita masukkan dalam struktur data pada objek Vue, kira-kira seperti berikut.

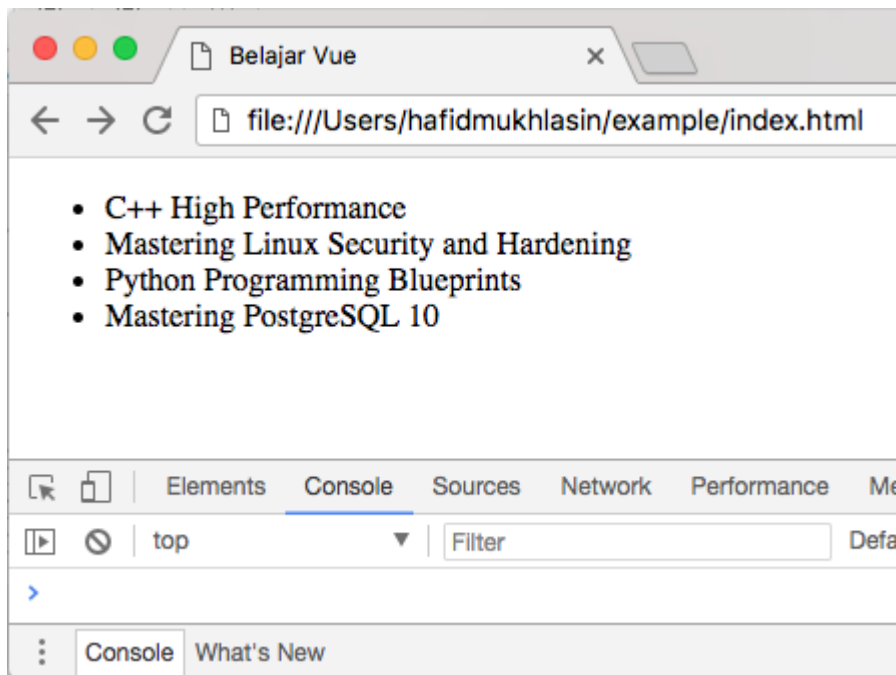
```
var vm = new Vue({  
  el: '#app',  
  data: {  
    books : [  
      'C++ High Performance',  
      'Mastering Linux Security and Hardening',  
      'Python Programming Blueprints',  
      'Mastering PostgreSQL 10'  
    ]  
  }  
})
```

Data tersebut ingin kita tampilkan menggunakan tag HTML list (li). Maka pada template Vue kita cukup mendefinisikannya sebagai berikut.

```
<div id="app">  
  <ul>  
    <li v-for="book in books">  
      {{ book }}  
    </li>  
  </ul>  
</div>
```

Vue mempunyai directive **v-for** yang berfungsi untuk melakukan perulangan sebanyak elemen data yang ada pada variabel **books**. Sedangkan **book** (tanpa s) merupakan elemen (item satuan) dari array **books** yang bisa langsung ditampilkan tentunya dengan menggunakan **mustache** **{{ }}**

Mari kita lihat hasilnya.



v-for Menggunakan Tag Template

Kode di atas bisa kita tulis dengan menggunakan tag template sebagai berikut.

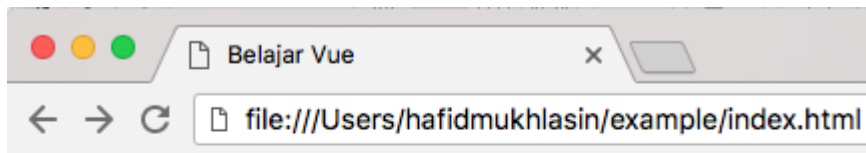
```
<div id="app">
  <ul>
    <template v-for="book in books">
      <li> {{ book }} </li>
    </template>
  </ul>
</div>
```

v-for Menggunakan Index

Index dari suatu array yang kita tampilkan melalui v-for bisa kita gunakan dengan menambahkan argumen kedua sebagai berikut

```
<li v-for="(book, index) in books">
  {{ index+1 }}. {{ book }}
</li>
```

Parameter index didefinisikan untuk menampung key dari array books. Oleh karena index dari suatu array dimulai dari 0 maka kita juga bisa menggunakan ekspresi matematika untuk memanipulasinya supaya index dimulai dari angka 1. Berikut hasilnya.



- 1. C++ High Performance
- 2. Mastering Linux Security and Hardening
- 3. Python Programming Blueprints
- 4. Mastering PostgreSQL 10

Selain `in`, kita bisa juga menggunakan delimiter `of` pada directive `v-for`.

```
<li v-for="(food, index) of foods">
```

Menampilkan Data Objek

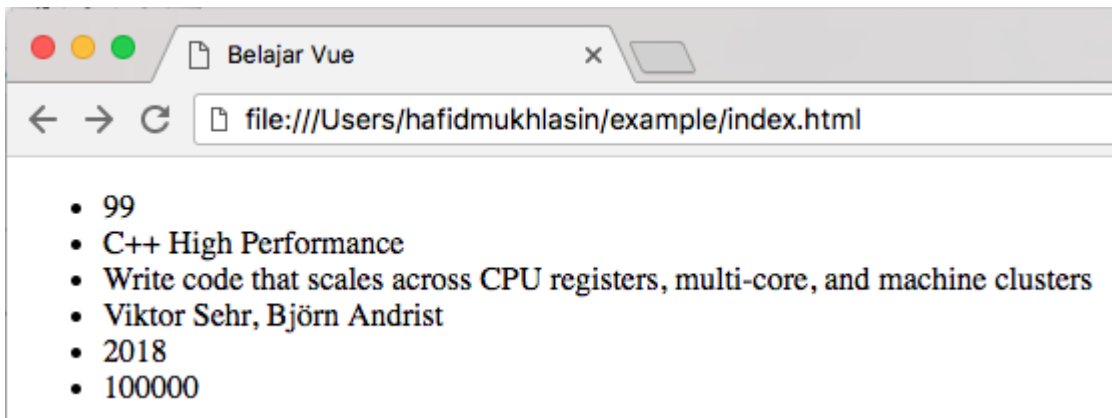
Sebagaimana array, data objek juga bisa kita tampilkan menggunakan directive `v-for`

```
book: {  
  id: 99,  
  title: 'C++ High Performance',  
  description: 'Write code that scales across CPU registers, multi-core,  
and machine clusters',  
  authors: 'Viktor Sehr, Björn Andrist',  
  publish_year: 2018,  
  price: 100000,  
}
```

Adapun kode untuk templatanya sebagai berikut.

```
<li v-for="value of book">  
  {{ value }}  
</li>
```

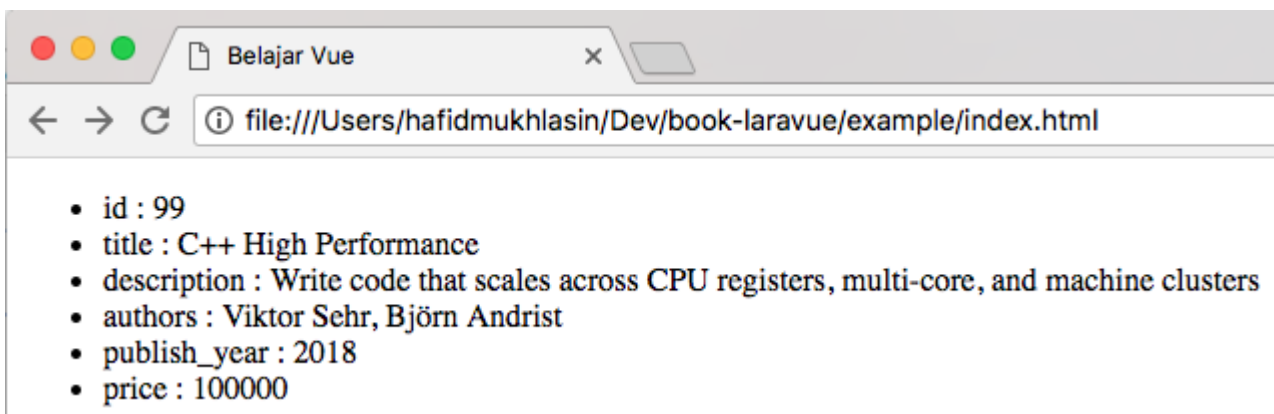
Berikut ini hasilnya.



Kita juga bisa menambahkan argumen kedua untuk key, seperti berikut.

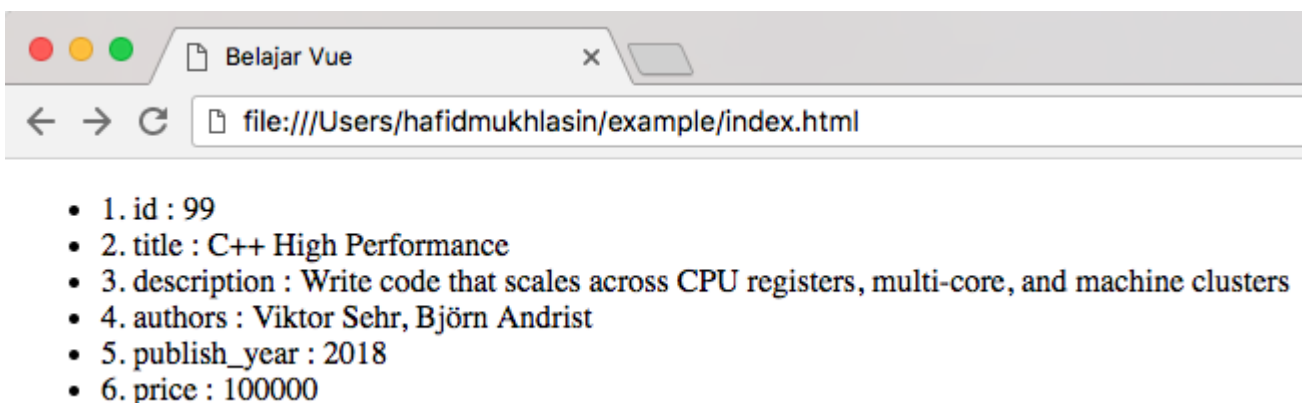
```
<li v-for="(value, key) of book">
  {{ key }} : {{ value }}
</li>
```

Berikut ini hasilnya.



Adapun argumen ketiga yang bisa kita tambahkan akan menjadi index dari objek tersebut.

```
<li v-for="(value, key, index) of book">
  {{ index+1 }}. {{ key }} : {{ value }}
</li>
```



Menampilkan Data Collection

Pada kasus nyata, seringkali kita dapati data tidak dalam bentuk array sederhana ataupun objek, melainkan dalam bentuk yang lebih kompleks semisal array dari objek atau dalam format JSON (Javascript Object Notation).

Perhatikan contoh data list berikut.

```
books : [  
  {  
    id: 99,  
    title: 'C++ High Performance',  
    description: 'Write code that scales across CPU registers, multi-  
core, and machine clusters',  
    authors: 'Viktor Sehr, Björn Andrist',  
    publish_year: 2018,  
    price: 100000,  
    image: 'c++-high-performance.png'  
  },  
  {  
    id: 100,  
    title: 'Mastering Linux Security and Hardening',  
    description: 'A comprehensive guide to mastering the art of  
preventing your Linux system from getting compromised',  
    authors: 'Donald A. Tevault',  
    publish_year: 2018,  
    price: 125000,  
    image: 'mastering-linux-security-and-hardening.png'  
  },  
  {  
    id: 101,  
    title: 'Mastering PostgreSQL 10',  
    description: 'Master the capabilities of PostgreSQL 10 to  
efficiently manage and maintain your database',  
    authors: 'Hans-Jürgen Schönig',  
    publish_year: 2016,  
    price: 90000,  
    image: 'mastering-postgresql-10.png'  
  },  
  {  
    id: 102,  
    title: 'Python Programming Blueprints',  
    description: 'How to build useful, real-world applications in the  
Python programming language',  
    authors: 'Daniel Furtado, Marcus Pennington',  
    publish_year: 2017,  
    price: 75000,  
    image: 'python-programming-blueprints.png'  
  },  
]
```

Misalnya data tersebut ingin kita tampilkan dalam bentuk HTML tabel, maka melalui pendekatan yang sama dengan sebelumnya kita bisa menyusun templatnya sebagai berikut.

```
<div id="app">
  <table border=1>
    <tr v-for="book of books">
      <td>
        
      </td>
      <td>
        title: {{ book.title }} <br>
        description: {{ book.description }} <br>
        authors: {{ book.authors }} <br>
        price: {{ book.price }}
      </td>
    </tr>
  </table>
</div>
```

Mari kita bedah satu persatu kode di atas.

Pertama, directive v-for kita letakkan di elemen tr pada table karena elemen itulah yang akan di-looping. Pilihan lain kita bisa juga menggunakan elemen `<template>`

```
<template v-for="book of books">
  <tr>
    ...
  </tr>
</template>
```

Kedua, pada kolom pertama tabel ini kita akan tampilkan cover buku menggunakan kode ``. Supaya nilai dari atribut src dari elemen image menjadi dinamis sesuai dengan data books, maka (sebagaimana yang telah kita bahas pada bab terdahulu) kita perlu menambahkan directive `v-bind` pada atribut tersebut. `v-bind:src` atau disingkat menjadi `:src`.

Oleh karena variabel `book` yang dihasilkan dari perulangan variabel `books` berbentuk objek, maka kita bisa panggil setiap item didalamnya dengan menggunakan titik diikuti nama keynya.

```
book.title // judul buku
book.image // nama file cover buku
```

Karena lokasi cover buku pada tutorial ini ada dalam direktori `images/vue/books` maka kita bisa tambahkan definisi direktori tersebut pada atribut `src`.

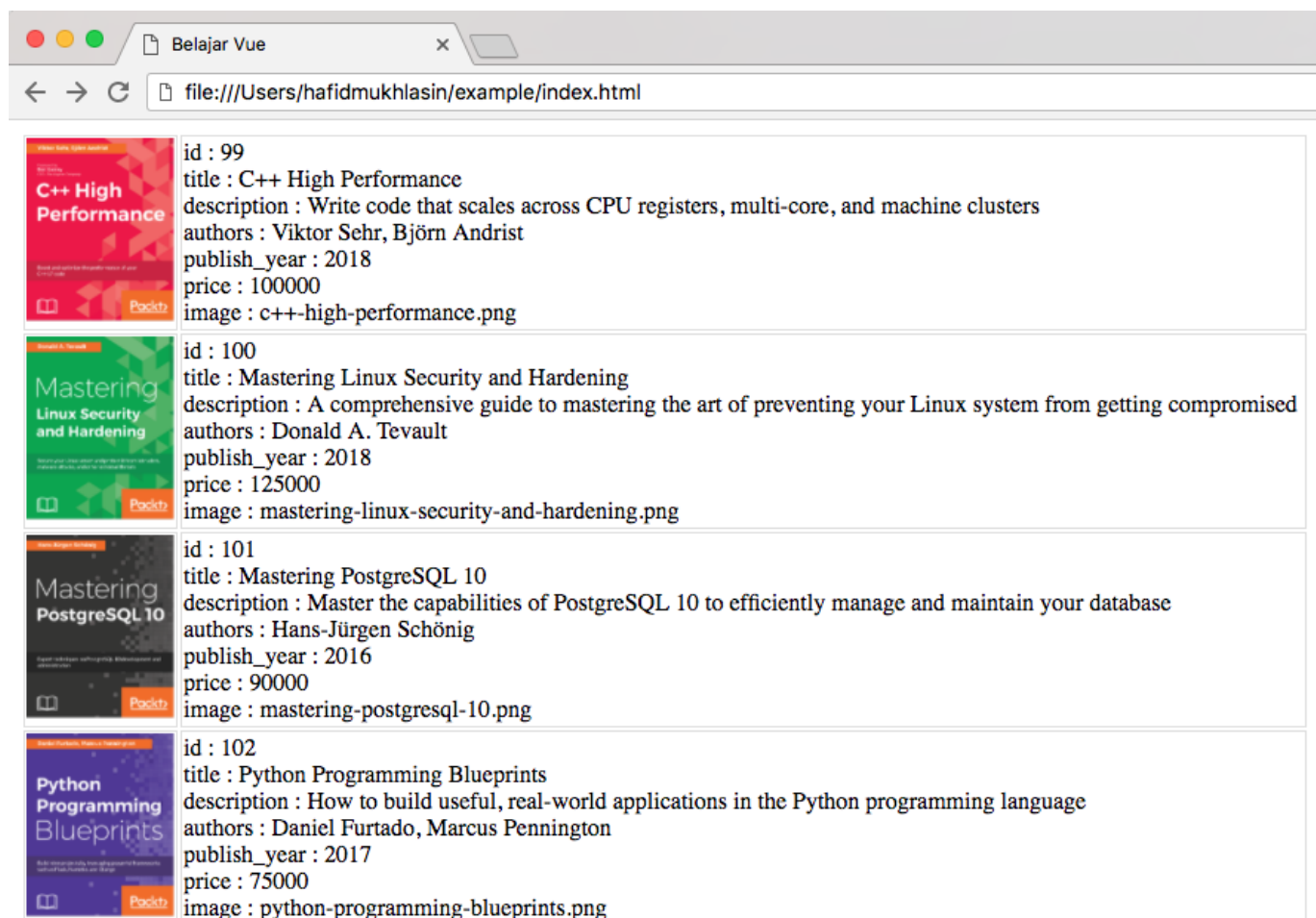
Ketiga, sebagaimana poin kedua, pada kolom kedua dari tabel, bisa kita tampilkan detail bukunya.

```
<td>
  title: {{ book.title }} <br>
  description: {{ book.description }} <br>
  authors: {{ book.authors }} <br>
  price: {{ book.price }}
</td>
```

Boleh juga kita gunakan directive `v-for` lagi sebab variabel `book` berbentuk objek.

```
<td>
  <template v-for="(value, key) of book">
    {{ key }} : {{ value }} <br>
  </template>
</td>
```

Berikut ini hasilnya.



Atribut Key

Terkait dengan metode menampilkan list, Vue menyarankan agar sebisa mungkin menggunakan atribut `key` pada tag HTML yang ikut dalam perulangan `v-for`. Key tersebut berperan sebagai penanda unik, sehingga Vue bisa melakukan tracking perubahan atas setiap tag HTML dari elemen list yang di-render.

Nilai atribut key sebenarnya bisa kita dapat darimana saja asal unik, misal:

- index dari array
- key atau properti dari objek

Berikut ini contohnya:

```
<li v-for="(book, index) of books" v-bind:key="index">
  {{ index+1 }}. {{ book }}
</li>
```

Pada contoh di atas, nilai atribut key berasal dari argumen index. Kita perlu menambahkan directive v-bind untuk mem-binding nilai dari atribut key yang dinamis sesuai dengan hasil perulangan v-for.

Tips: penulisan dari **v-bind:key** bisa disingkat menjadi **:key** saja (shorthand).

Membatasi v-for menggunakan v-if

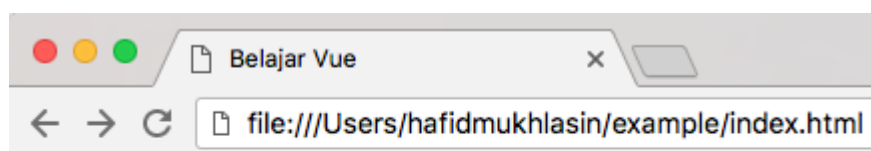
Adakalanya kita hanya ingin menampilkan data dengan kriteria tertentu saja, misalnya menampilkan data buku yang harganya lebih besar sama dengan 100 ribu.

Berdasarkan data variabel **books**, yang memenuhi kriteria tersebut hanya dua buku yaitu buku dengan judul "C++ High Performance" dan "Mastering Linux Security and Hardening".

Penerapannya di Vue sebagai berikut.

```
<ul>
  <li v-for="(book, index) of books" :key="index" v-
if="book.price>=100000">
    {{ book.title }}
  </li>
</ul>
```

Kode v-if melakukan pengujian apakah harga buku lebih besar sama dengan 100000, jika benar maka data buku ditampilkan, demikian sebaliknya.



- **C++ High Performance**
- **Mastering Linux Security and Hardening**

Fungsi v-if pada v-for ini mirip dengan filter data, "kelemahan"-nya adalah index dari array jadi tidak berurutan.

Perubahan (mutation) Data Pada Array

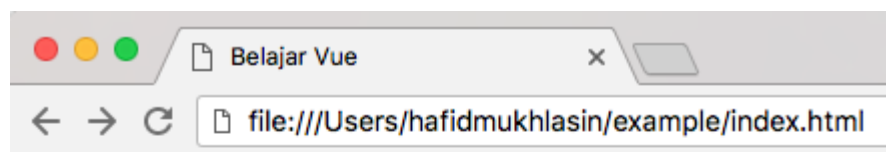
Data pada contoh sebelumnya merupakan data statis yang tidak atau belum ada perubahan. Namun sebagaimana yang telah dijelaskan diawal bahwa data pada Vue bersifat observer artinya perubahannya senantiasa dipantau dan bisa menjadi pemicu untuk perubahan yang lain. Sebagai contoh, apabila data dihubungkan dengan template maka perubahan data akan menyebabkan perubahan pada tampilan atau DOM melalui rendering ulang.

Catatan: Kaidah pada Vue ini tetap mengikuti aturan main pada Javascript. Sebagai contoh, perubahan data berbentuk array harus menggunakan fungsi-fungsi yang tersedia di Javascript sehingga kita tidak bisa langsung mengeset data array secara langsung menggunakan index-nya.

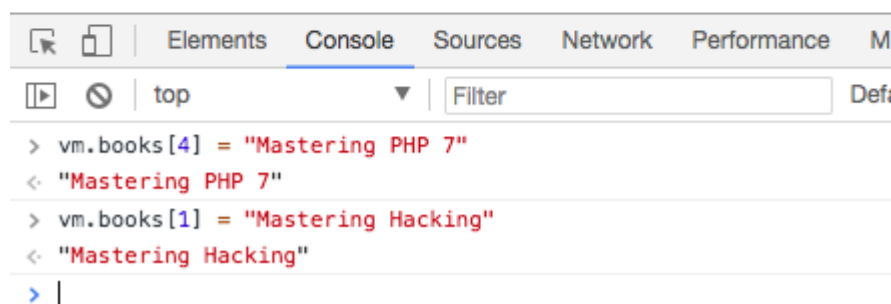
Asumsinya, kita menggunakan data array books.

```
books : [  
  'C++ High Performance',  
  'Mastering Linux Security and Hardening', 'Python Programming  
Blueprints',  
  'Mastering PostgreSQL 10'  
]
```

Kita akan coba melakukan manipulasi melalui console.



- C++ High Performance
- Mastering Linux Security and Hardening
- Python Programming Blueprints
- Mastering PostgreSQL 10



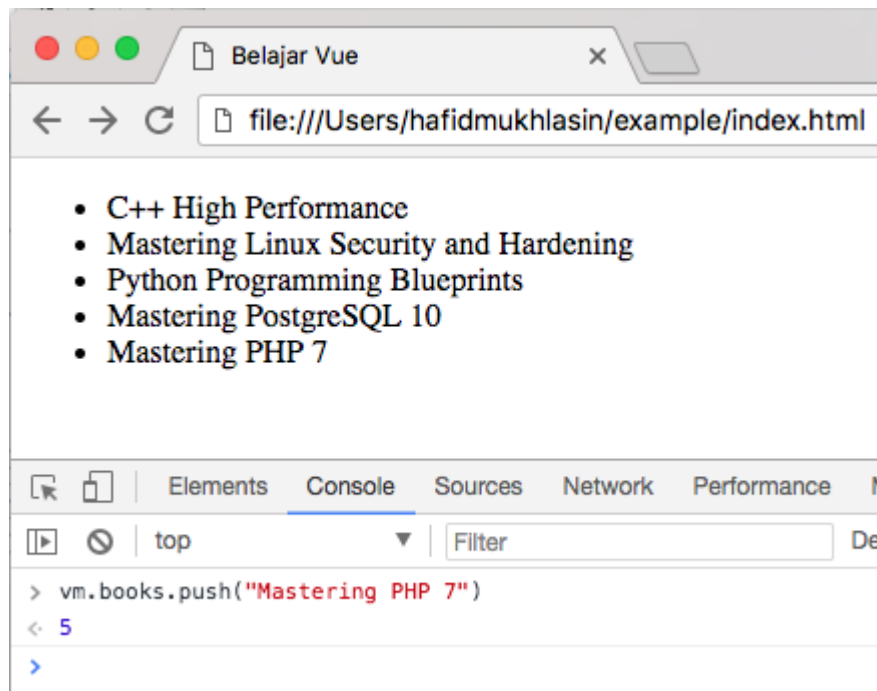
Pada contoh di atas, meski kita menambahkan data elemen baru pada variabel books (Mastering PHP 7), namun tampilan daftar buku tetap tidak berubah atau tidak dirender ulang. Demikian juga perubahan data judul buku pada index pertama dari "Mastering Linux Security and Hardening" menjadi "Masterng Hacking" pun juga tidak bersifat observe atau tidak reaktif.

Untuk mengatasi hal ini, kita perlu menggunakan fungsi-fungsi built-in Javascript untuk memanipulasi data berbentuk array. Fungsi-fungsi tersebut yaitu push(), pop(), shift(), unshift(), sort(), reverse(), dan splice().

push() & pop()

Fungsi push digunakan untuk menambahkan data elemen baru pada suatu array pada posisi index terakhir. Sebaliknya fungsi pop untuk menghapus elemen terakhir dari suatu array. Misalnya:

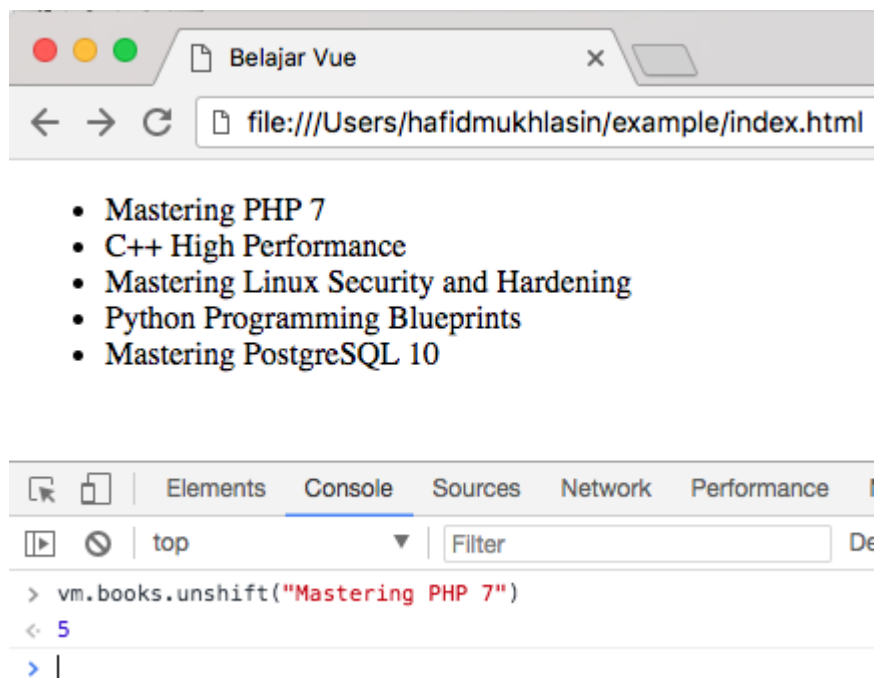
```
vm.books.push('Mastering PHP 7')
```



unshift() & shift()

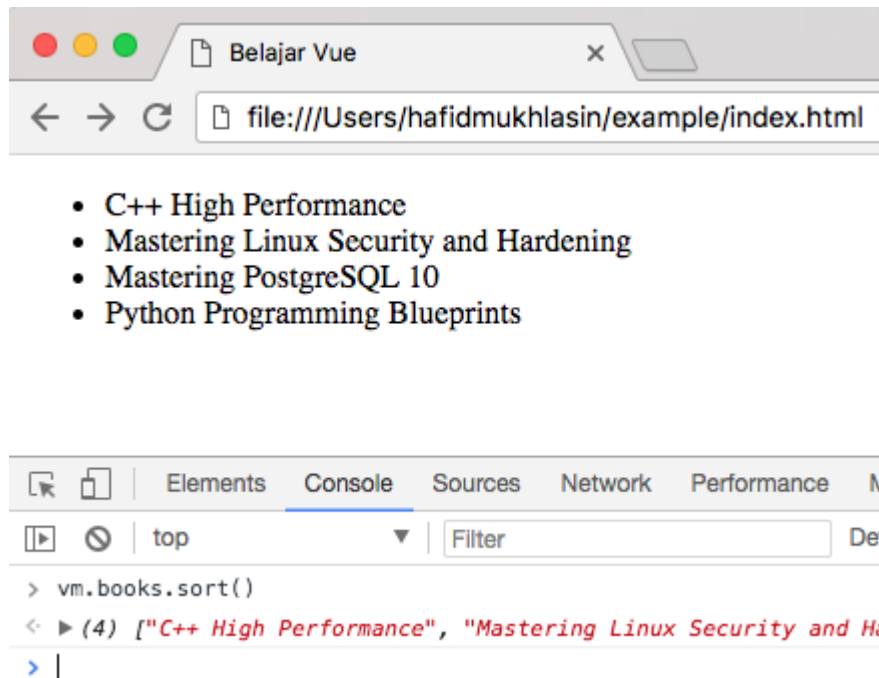
Fungsi unshift digunakan untuk menambahkan data elemen baru pada suatu array pada posisi index pertama (0). Sebaliknya fungsi shift untuk menghapus elemen pertama dari suatu array. Misalnya:

```
vm.books.unshift('Mastering PHP 7')
```



sort() & reverse()

Fungsi sort digunakan untuk mengurutkan data elemen pada suatu array secara ascending, sedangkan fungsi reverse melakukan sebaliknya. `vm.books.sort()`

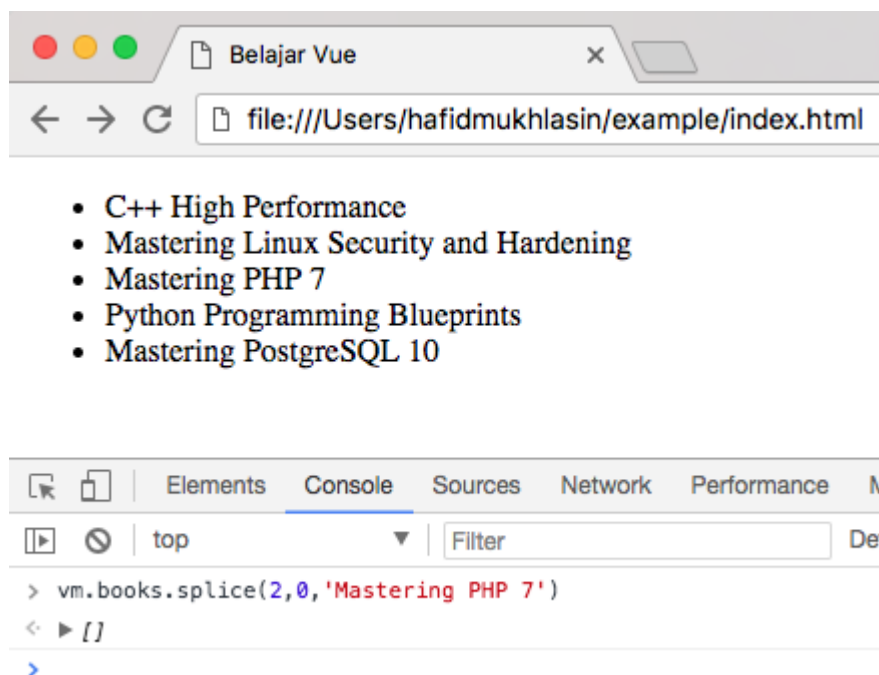


`splice()`

Fungsi splice ini multi fungsi, bisa digunakan untuk menambahkan data elemen baru pada suatu array. Misalnya: `vm.books.splice(2,0,'Mastering PHP 7')`

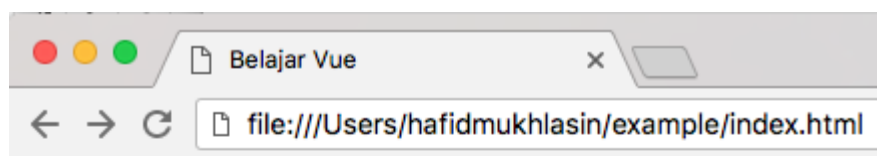
Perintah tersebut akan menambahkan data elemen baru yaitu 'Mastering PHP 7' pada index ke dua dari array.

- parameter pertama menunjukkan posisi index dari data yang akan ditambahkan.
- adapun parameter kedua menunjukkan jumlah elemen pada array yang akan dihapus.

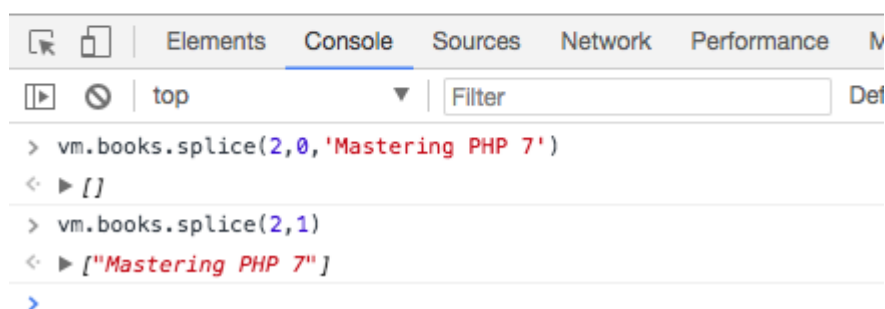


Fungsi splice juga bisa digunakan untuk menghapus elemen dari suatu array pada index tertentu. Misalnya: `vm.books.splice(2,1)`

Perintah tersebut akan menghapus elemen array pada index ke-dua sebanyak satu elemen.



- C++ High Performance
- Mastering Linux Security and Hardening
- Python Programming Blueprints
- Mastering PostgreSQL 10



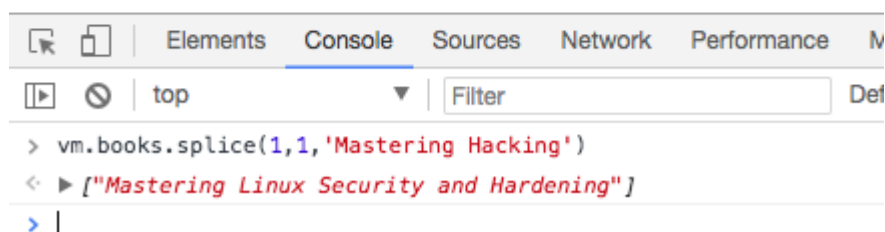
Perintah splice pun bisa digunakan untuk mengubah elemen dari suatu array pada index tertentu. Misalnya:

```
vm.books.splice(1,1,'Mastering Hacking')
```

Perintah tersebut akan menghapus elemen array index ke-satu sekaligus menambahkan elemen baru pada index ke-satu juga. (ingat: index array dimulai dari 0)



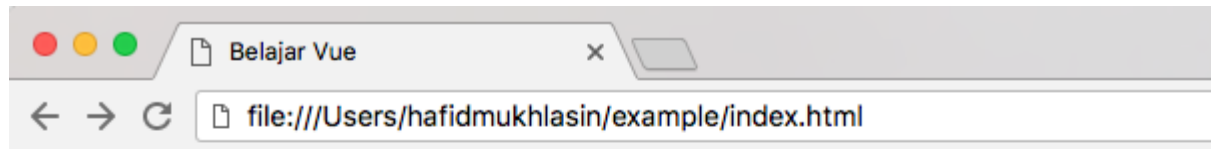
- C++ High Performance
- Mastering Hacking
- Python Programming Blueprints
- Mastering PostgreSQL 10



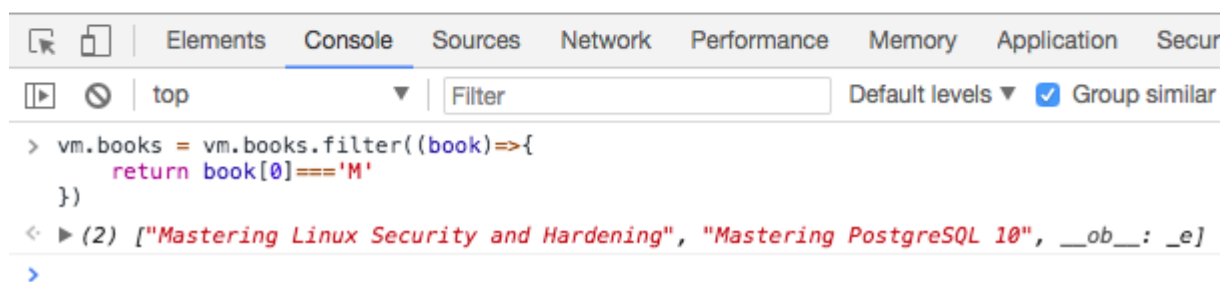
Disamping fungsi-fungsi di atas, terdapat beberapa fungsi built-in lagi terkait array yang bisa kita gunakan untuk memanipulasi elemen pada array namun sifatnya tidak melakukan perubahan langsung pada current array melainkan mengembalikan data array baru. Diantaranya: `filter()`, `concat()` dan `slice()`.

Satu contoh, fungsi filter digunakan untuk mem-filter elemen dari array berdasarkan kriteria tertentu. Misalnya mengambil judul buku (dari books) yang diawali dengan huruf **M**.

```
vm.books = vm.books.filter((book)=>{
  return book[0]=== 'M'
})
```



- Mastering Linux Security and Hardening
- Mastering PostgreSQL 10



Implementasi pada Vue biasanya dengan menggunakan properti **computed**. Properti ini berupa fungsi-fungsi yang nilainya senantiasa dipantau atau observe sesuai dengan perubahan data.

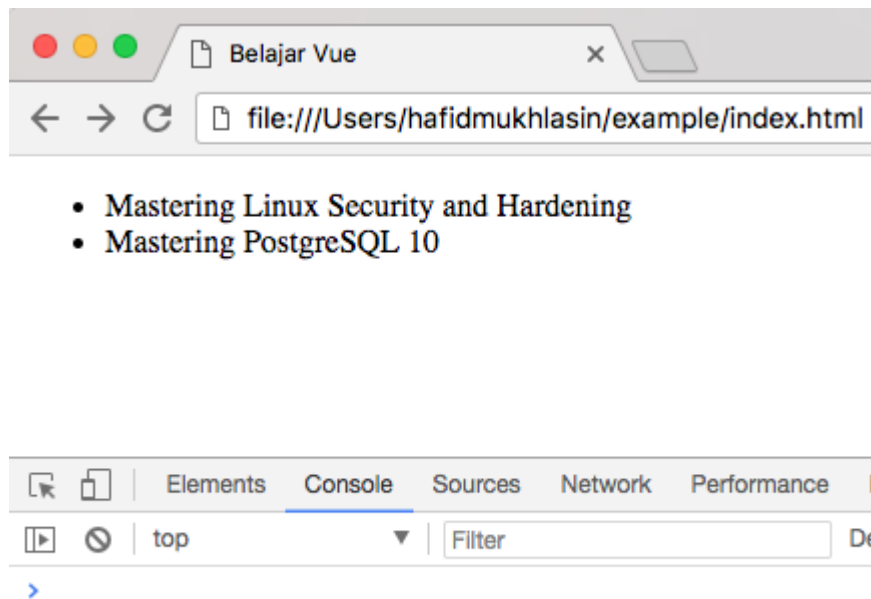
Berikut ini contoh kodenya

```
<div id="app">
  <ul>
    <li v-for="(book, index) of booksPrefixM" :key="index">
      {{ book }}
    </li>
  </ul>
</div>

<script type="text/javascript">
var vm = new Vue({
  el: '#app',
  data: {
    books : [
      'C++ High Performance',
      'Mastering Linux Security and Hardening', 'Python Programming
Blueprints',
      'Mastering PostgreSQL 10'
    ]
  }
})
```

```
    },
    computed: {
      booksPrefixM() {
        return this.books.filter((book)=>{
          return book[0]=== 'M'
        })
      }
    }
  })
</script>
```

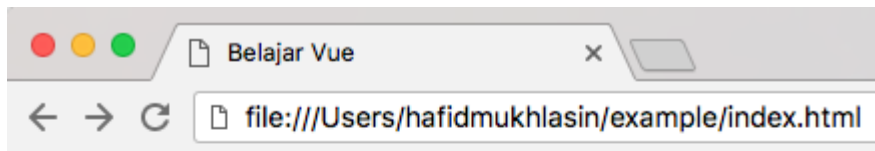
Berikut ini hasilnya



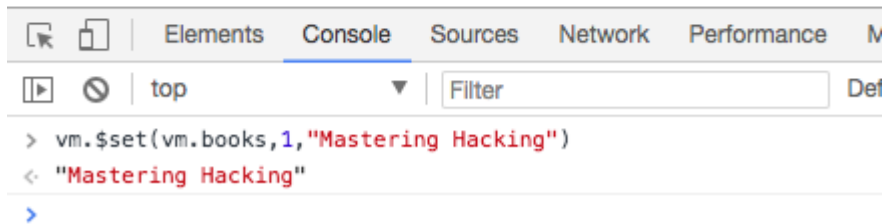
fungsi set pada Vue

Sebenarnya Vue juga menyediakan fungsi built-in untuk mengatasi masalah keterbatasan pada Javascript ini, jika pada contoh sebelumnya untuk mengubah data pada suatu elemen array menggunakan fungsi splice maka sebenarnya kita bisa juga melakukannya dengan fungsi set Vue.

Vue.set(data_array, index_yang_akan_diganti, nilai_baru)



- C++ High Performance
- Mastering Hacking
- Python Programming Blueprints
- Mastering PostgreSQL 10



Perubahan Data Pada Objek

Seperti halnya array, objek pun juga perlu fungsi khusus untuk melakukan perubahan data.

Asumsi, kita gunakan data objek berikut.

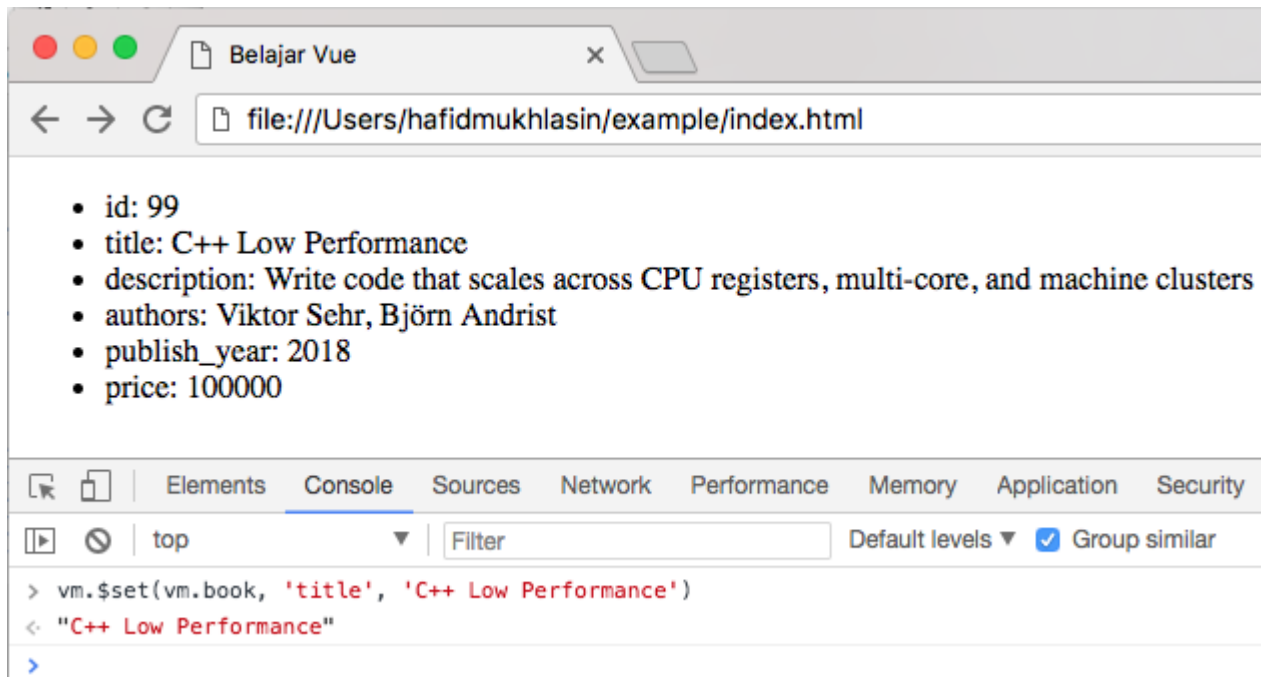
```
book: {  
  id: 99,  
  title: 'C++ High Performance',  
  description: 'Write code that scales across CPU registers, multi-core,  
and machine clusters',  
  authors: 'Viktor Sehr, Björn Andrist',  
  publish_year: 2018,  
  price: 100000,  
}
```

Untuk melakukan penambahan data properti pada objek, kita bisa gunakan fungsi set bawaan Vue.

```
Vue.set(object, key, value)
```

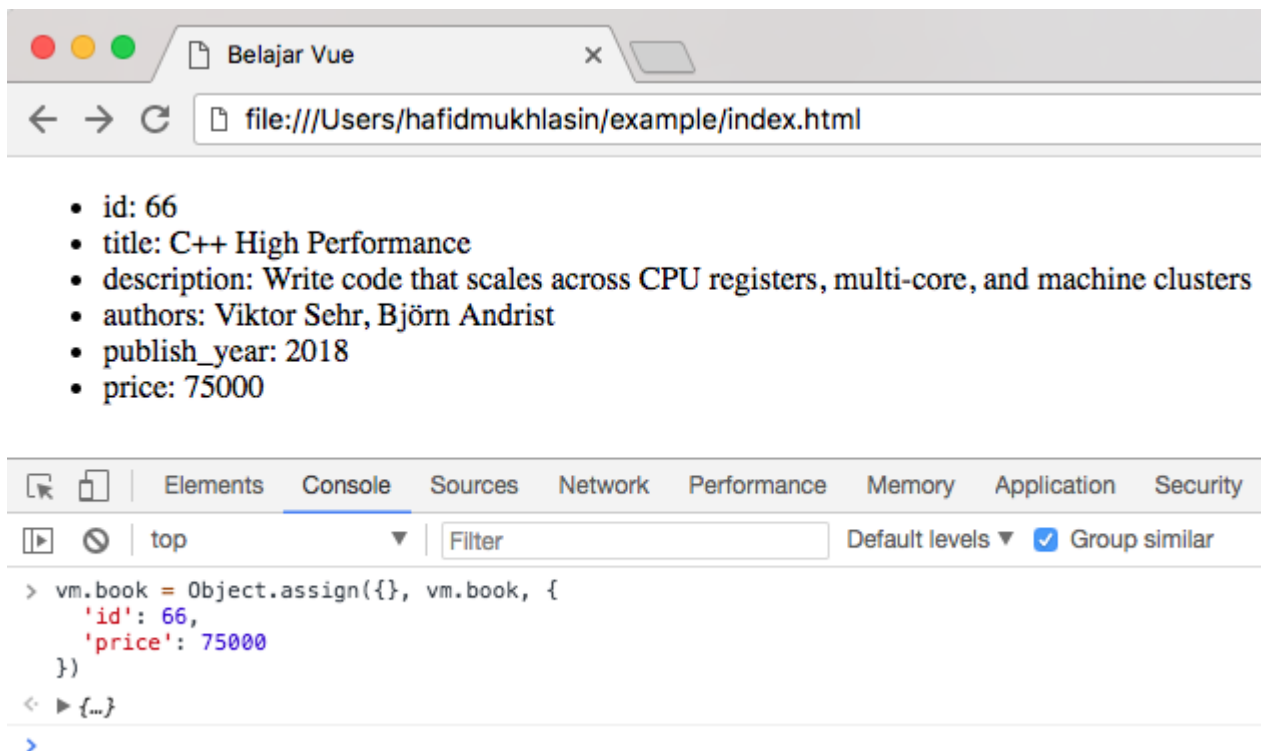
atau

```
vm.$set(object, key, value)
```



Namun, jika properti yang ingin kita tambahkan lebih dari satu maka kita bisa gunakan cara berikut.

```
vm.book = Object.assign({}, vm.book, {
  umur: 27,
  status: 'Perjaka'
})
```



Kesimpulan

Pada bab ini kita telah belajar bagaimana menampilkan data dalam bentuk list (array, object atau array of object) dengan berbagai macam bentuk variasinya. Menggunakan filter untuk menampilkan data tertentu saja. Tidak

hanya itu, kita juga belajar bagaimana memanipulasi data dalam bentuk list supaya reaktif. List ini akan cukup banyak kita jumpai implementasinya dalam kasus nyata nanti.

Setelah belajar bagaimana cara menampilkan data maka pada bab selanjutnya kita akan belajar tentang bagaimana cara menangani input data dari user melalui form dan sejenisnya.

Keep on the track!

Form

Pada bab ini kita akan belajar tentang bagaimana menangani input data dari user melalui form serta bagaimana memanipulasi tampilan datanya.

Catatan: untuk latihan pada bab ini, **save as** file html yang berisi kode Vue kita menjadi form.html

Input Binding

Form HTML memiliki berbagai jenis field input seperti text, password, radio, dsb yang peruntukannya tentu berbeda tergantung dari data yang ingin ditangkap.

```
<input name="username" type="text">
<input name="password" type="password">
<input name="gender" type="radio">
```

Terkait dengan input binding ini, yang kita butuhkan adalah two way data binding, di mana nilai dari field input terhubung dengan data secara dua arah. Artinya perubahan field input yang dilakukan oleh user akan menyebabkan perubahan variabel data, sebaliknya perubahan variabel data akan menyebabkan perubahan pada field input.

Berdasarkan, bahasan tentang directive, mungkin kita segera bisa menemukan triknya. Ya, yang kita butuhkan dua directive sekaligus v-on sebagai listener perubahan field input, dan v-bind yang bertugas mem-*bind* perubahan variabel data untuk diterapkan pada field input.

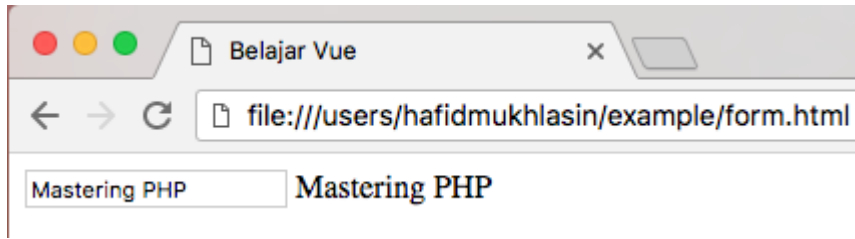
```
<div id="app">
  <form>
    <input type="text" name="title" :value="title" @input="title =
    $event.target.value" />

    {{ title }}
  </form>
</div>

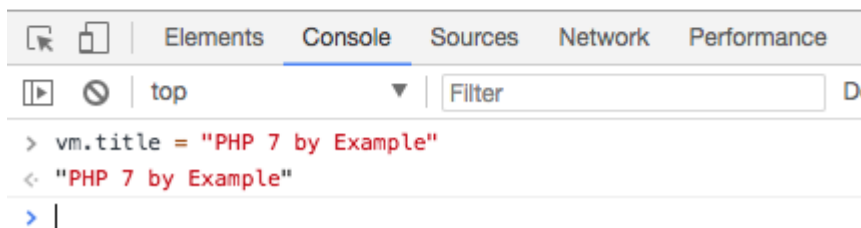
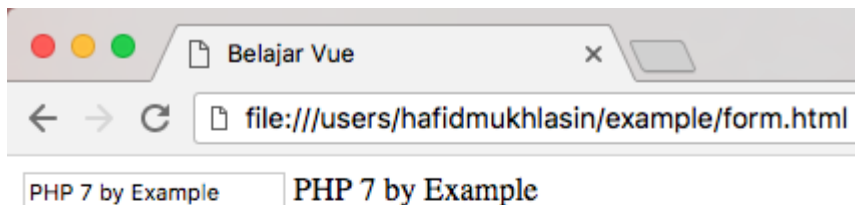
<script>
var vm = new Vue({
  el: '#app',
  data: {
    title: "Mastering "
  },
})
</script>
```

Atribut value di-*bind* dan event oninput di-*listen*. Kode `:value="title"` menunjukkan bahwa nilai dari field input ini di-*bind* dengan variabel `title`, sedangkan kode `@input="title = $event.target.value"` artinya ketika field diinput maka nilai variabel `title` akan diubah sesuai isian user.

Berikut ini hasilnya.



Ketika field input diubah maka variabel title juga berubah.



Ketika variabel title diubah via console `vm.title = "PHP 7 by Example"` maka nilai dari field input juga akan mengikuti.

Catatan: metode ini akan dipakai ketika kita bermain dengan komponen.

Karenanya, untuk mengatasi "kerumitan" ini, Vue memperkenalkan directive `v-model` yang bertugas melakukan two way data binding tersebut. Berikut ini contohnya.

```
<form>
  <input type="text" name="title" v-model="title" placeholder="masukkan
  judul">
    {{ title }}
  <br><br>
  <textarea name="description" v-model="description"
  placeholder="masukkan deskripsi"></textarea>
    {{ description }}
</form>
```

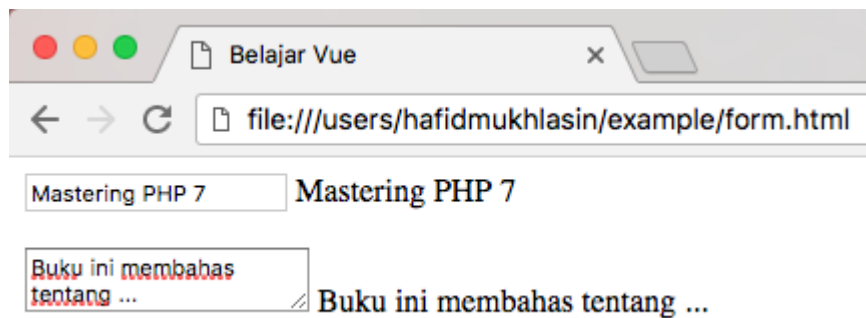
Catatan: pastikan selalu menggunakan atribut name pada setiap field yang digunakan, disamping merupakan best practice juga akan berguna nanti pada bahasan selanjutnya. Nilai atribut name tidak

harus sama dengan nilai dari atribut v-model, hanya saja karena keduanya identik maka sebaiknya disamakan saja.

Tentu saja kita perlu tambahkan dua variabel yaitu title dan description.

```
data: {  
  title: '',  
  description: ''  
},
```

Hasilnya sebagai berikut.



Text

Pada contoh di atas, elemen field input text, password dan textarea menerima data dalam bentuk teks atau string, sehingga tipe data pada variabel datanya adalah string juga.

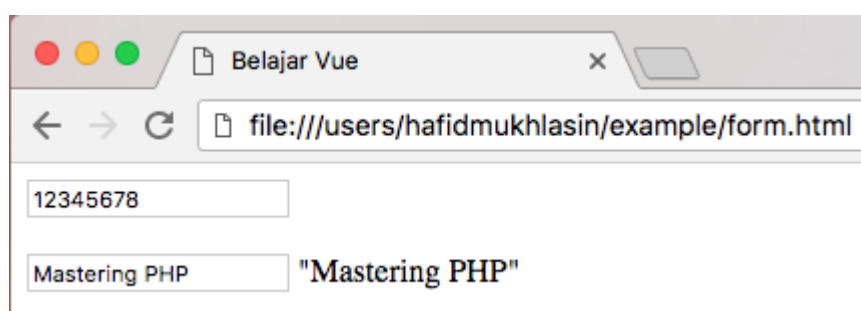
Directive v-model juga memiliki modifier, diantaranya adalah **number** dan **trim**.

Modifier **number** digunakan untuk memastikan input data dari user bertipe numeric. Sedangkan modifier **trim** digunakan untuk menghapus spasi putih diawal atau akhir dari string.

```
<input type="number" name="price" v-model.number="price">  
<br><br>  
<input type="text" name="title" v-model.trim="title"  
placeholder="masukkan judul"> "{{ title }}"
```

Catatan: tambahkan variabel price.

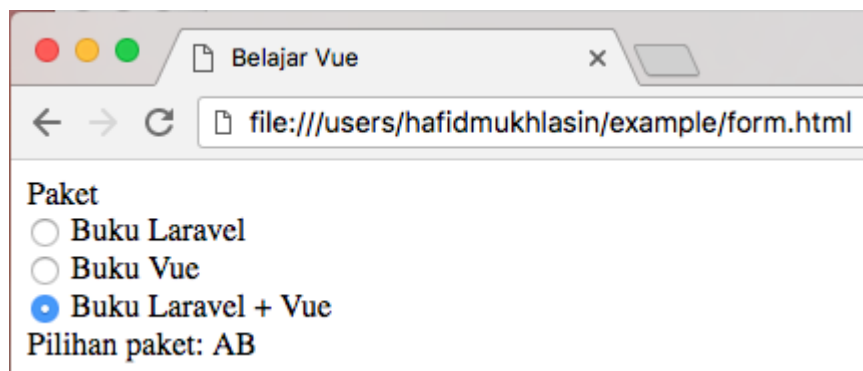
Mari kita lihat hasilnya.



Field input radion bertipe data teks.

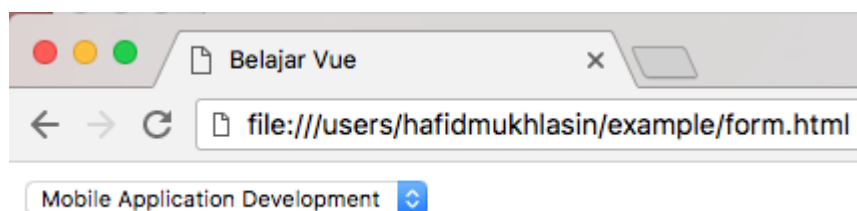
```
Paket <br>
<input type="radio" name="bukuA" value="A" v-model="paket">
<label for="bukuA">Buku Laravel</label>
<br>
<input type="radio" name="bukuB" value="B" v-model="paket">
<label for="bukuB">Buku Vue</label>
<br>
<input type="radio" name="bukuAB" value="AB" v-model="paket">
<label for="bukuAB">Buku Laravel + Vue</label>
<br>
<span>Pilihan paket: {{ paket }}</span>
```

```
data: {
  paket: ''
}
```



Field input select (single select) juga bertipe data teks. (tambahkan variabel category)

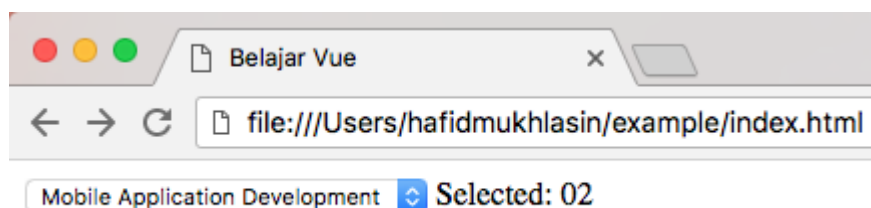
```
<select name="category" v-model="category">
  <option disabled value="">Please select one</option>
  <option>Graphics Programming</option>
  <option>Mobile Application Development</option>
  <option>Virtual and Augmented Reality</option>
</select>
```



Catatan: Jika menggunakan field select, sebaiknya menambahkan elemen option disabled dengan nilai kosong sebagaimana contoh di atas untuk mengatasi masalah pada IOS.

Pada contoh di atas, atribut value pada elemen option tidak didefinisikan sehingga nilai dari variabel category mengikuti text diantara elemen option. Namun apabila value didefinisikan maka yang digunakan adalah value tersebut.

```
<select name="category" v-model="category">
  <option disabled value="">Please select one</option>
  <option value="01">Graphics Programming</option>
  <option value="02">Mobile Application Development</option>
  <option value="03">Virtual and Augmented Reality</option>
</select>
<span>Selected: {{ category }}</span>
```

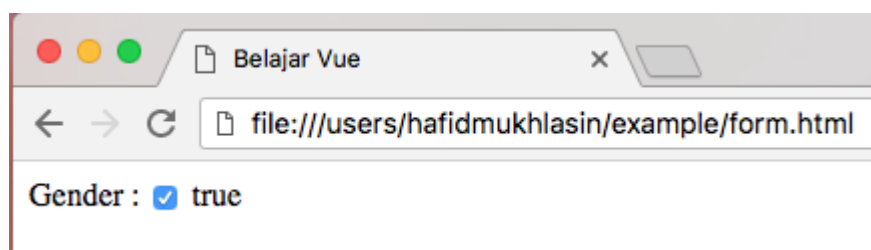


Boolean

Input data bertipe data boolean (true atau false) biasanya terjadi pada field input checkbox tunggal. Contoh pada input data gender.

```
Gender :
<input type="checkbox" name="checkbox" v-model="gender">
<label for="checkbox">{{ gender }}</label>
```

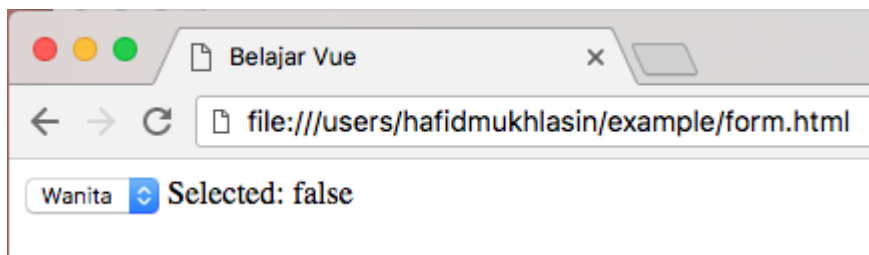
```
data: {
  gender: true
}
```



Field input select tunggal dengan dua option juga bisa bertipe boolean. Perhatikan contoh berikut.

```
<select name="gender" v-model="gender">
  <option value="false">Wanita</option>
  <option value="true">Pria</option>
```

```
</select>
<span>Selected: {{ gender }}</span>
```



Array

Input data bertipe array terjadi pada field input dengan kemungkinan pilihan lebih dari satu seperti checkbox multiple dan select multiple.

```
Hobi <br>
<input type="checkbox" name="hobby1" value="nonton" v-model="hobbies">
<label for="hobby1">Nonton</label>
<input type="checkbox" name="hobby2" value="jalan" v-model="hobbies">
<label for="hobby2">Jalan</label>
<input type="checkbox" name="hobby3" value="makan" v-model="hobbies">
<label for="hobby3">Makan</label>
<br>
<span>Pilihan hobi: {{ hobbies }}</span>

<hr>

<select v-model="categories" multiple>
  <option value="01">Graphics Programming</option>
  <option value="02">Mobile Application Development</option>
  <option value="03">Virtual and Augmented Reality</option>
</select>
<span>Selected: {{ categories }}</span>
```

Adapun definisi dari variabel data harus sebagai array.

```
var vm = new Vue({
  el: '#app',
  data: {
    hobbies: [],
    categories: []
  }
})
```

Tampilan yang berulang dengan pola tertentu seperti option pada field input select, tentu saja bisa kita *generate* menggunakan *directive v-for* sebagaimana yang telah dibahas pada bagian terdahulu.

```
data: {
  categories: [],
  options: [
    { text: 'Graphics Programming', value: '01' },
    { text: 'Mobile Application Development', value: '02' },
    { text: 'Virtual and Augmented Reality', value: '03' }
  ]
}
```

```
<select name="categories" v-model="categories" multiple>
  <option v-for="option in options" :value="option.value">
    {{ option.text }}
  </option>
</select>
<span>Selected: {{ categories }}</span>
```

Atribut value *dibind*.

Filtering Data List

Sebelum kita gunakan form untuk submit data, ada satu materi terkait dengan list namun sengaja dibahas pada bab ini karena terkait dengan form input. Sederhana sekali sebenarnya karena secara umum konsepnya telah dibahas pada bab list tersebut.

Materi ini adalah membuat field pencarian atau filtering data buku dalam bentuk list. Pada template kita perlu satu filed input dan list.

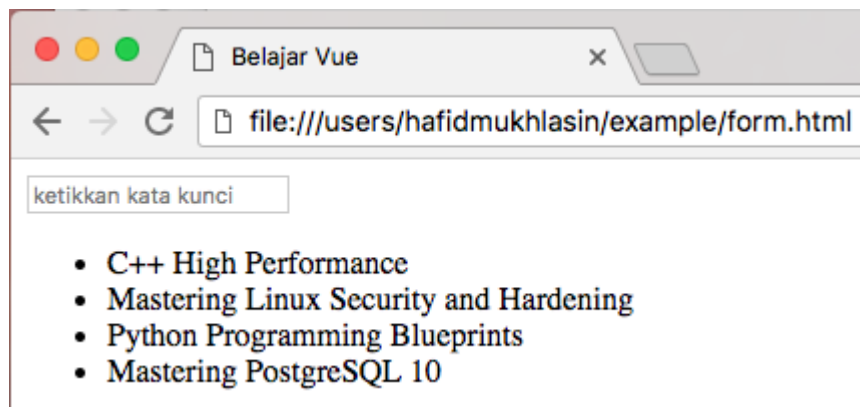
```
<div id="app">
  <input type="text" v-model="keyword" placeholder="ketikkan kata
kunci">
  <ul>
    <li v-for="(book, index) of filterBooks" :key="index">
      {{ book }}
    </li>
  </ul>
</div>
```

Adapun kode Javascript-nya sebagai berikut.

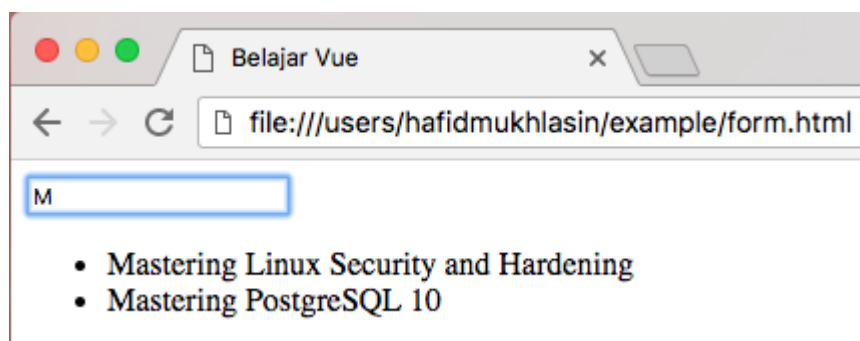
```
var vm = new Vue({
  el: '#app',
  data: {
    keyword: '',
    books : [
      'C++ High Performance',
      'Mastering Linux Security and Hardening', 'Python Programming
Blueprints',
      'Mastering PostgreSQL 10'
    ]
  },
  computed: {
    filterBooks() {
      return this.books.filter((book)=>{
        return book.includes(this.keyword)
      })
    }
  }
})
```

Ada dua variabel yaitu keyword untuk menampung kata kunci dan books untuk menampung data buku. Disamping itu ada satu fungsi pada properti computed yaitu filterBooks. Fungsi filterBook akan melakukan pengecekan adakah item di variabel books yang berisi variabel keyword menggunakan method bawaan Javascript yaitu `includes`.

Hasilnya sebagai berikut.



Ketika diinput teks



Handling Submit Form & Validation

Pertanyaan: Apa fungsi elemen HTML form jika field bisa lewat begitu saja melalui v-model tanpa perlu submit?

Sebagaimana kita ketahui bahwa form memiliki event submit yang umumnya digunakan untuk mengirimkan data (yang berasal dari field input user) ke server atau ke bagian lain dari aplikasi.

Sudah menjadi konsensus bahwa sebuah form yang berisi beberapa field input data membutuhkan button submit yang menandai bahwa semua data yang diisi melalui field input tersebut siap dikirimkan.

Berikut ini contoh template form input data buku.

```
<form @submit="submitForm($event)" action="http://example.com/add-product"
method="post">
  <label>Title:</label>
  <input type="text" v-model="title" />

  <label>Description:</label>
  <textarea v-model="description"></textarea>

  <label>Authors:</label>
  <input type="text" v-model="authors">

  <label>Price:</label>
  <input type="number" v-model.number="price">

  <label>Categories:</label>
  <select v-model="categories" multiple>
```

```

        <option v-for="option in options" :value="option.value">
            {{ option.text }}
        </option>
    </select>

    <label></label>
    <input type="submit" value="Submit">
</form>

```

Pada elemen form `<form @submit="submitForm" action="http://example.com/add-product" method="post">`, kita menggunakan 3 atribut yaitu directive `v-on:submit` atau `@submit` (yang akan dijalankan ketika form disubmit), `action` (endpoint pengiriman data), dan `method` (metode pengiriman data apakah get atau post).

Jika kita menggunakan Vue untuk membuat aplikasi SPA (Single Page Application) maka dua atribut terakhir yaitu `action` dan `method` menjadi *unfaedah* 😊. Oleh karena itu, atribut sekaligus directive `@submit` yang kita jadikan tumpuan untuk dibahas pada bagian ini.

Directive ini pada contoh diatas memanggil fungsi `submitForm` yang mana pada fungsi ini kita bisa gunakan untuk misalnya: memvalidasi isian dari user, melakukan kalkulasi jika diperlukan, mengirimkan data isian tersebut ke server melalui http client, dsb.

```

var vm = new Vue({
  el: '#app',
  data: {
    title: '',
    description: '',
    authors: '',
    price: 0,
    categories: [],
    options: [
      { text: 'Graphics Programming', value: '01' },
      { text: 'Mobile Application Development', value: '02' },
      { text: 'Virtual and Augmented Reality', value: '03' }
    ]
  },
  methods: {
    submitForm(event){
      console.log(event)

      // kode validasi

      // kode kirim ke server

      // kode status informasi
      alert('Terima kasih')

      // block redirect ke action
      event.preventDefault()
    }
  }
})

```

```
}  
})
```

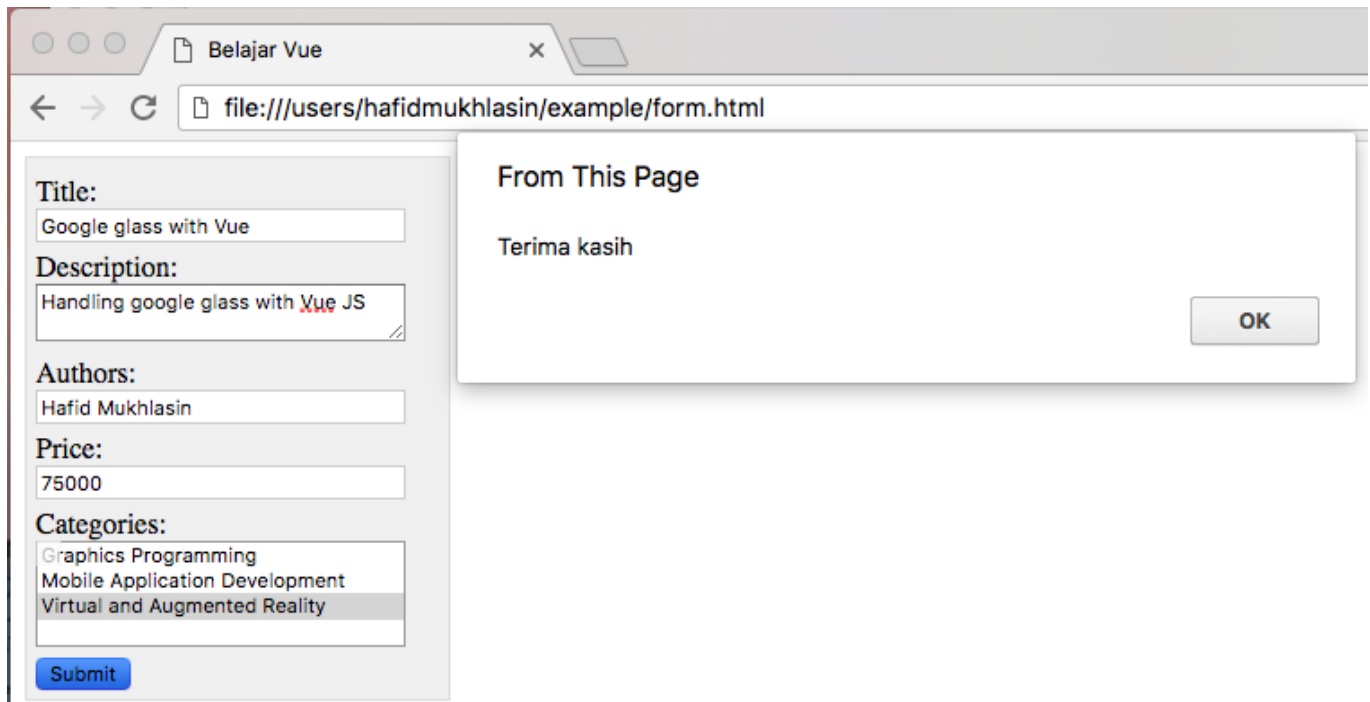
Kode `event.preventDefault()` pada methods `submitForm` berfungsi untuk mencegah agar form tidak diredirect ke alamat yang didefinisikan di atribut action, hal ini dikarenakan pengiriman data tidak melalui cara normal HTML melainkan melalui Javascript (http client). Disamping cara ini, kita bisa juga menggunakan modifier `prevent` pada directive `@submit`

```
<form @submit.prevent="submit"...
```

Supaya tampilan form lebih rapi, untuk sementara kita menggunakan style CSS berikut (letakkan pada elemen head HTML).

```
<style>  
form {  
  border: 1px solid #ddd;  
  padding: 5px;  
  width: 225px;  
  background: #efefef;  
}  
label {  
  display: block;  
  margin-top: 5px;  
}  
  
input, textarea, select, option {  
  min-width: 200px;  
}  
</style>
```

Mari kita lihat hasilnya.



Selanjutnya, kita akan fokus pada bagian methods `submitForm()`.

Validasi Data

Proses validasi adalah proses memastikan setiap isian yang diinput oleh user melalui form tersebut sesuai dengan persyaratan minimal yang kita tentukan. Misalnya: title harus diisi dan minimal 3 karakter, description boleh tidak diisi namun kalau diisi maka tidak boleh lebih dari 500 karakter, price tidak boleh minus, dsb. Jika ditemukan terdapat isian yang tidak memenuhi syarat minimal maka akan dimunculkan pesan peringatan berupa alert serta dicatat sebagai error (counter). Pada bagian akhir kemudian dicek secara total apakah terdapat error (error lebih dari 0) ataupun tidak (error = 0), jika tidak ada error maka ditampilkan pesan terima kasih dan kode untuk mengirim data tersebut ke server.

```
submitForm(event){
  let error = 0

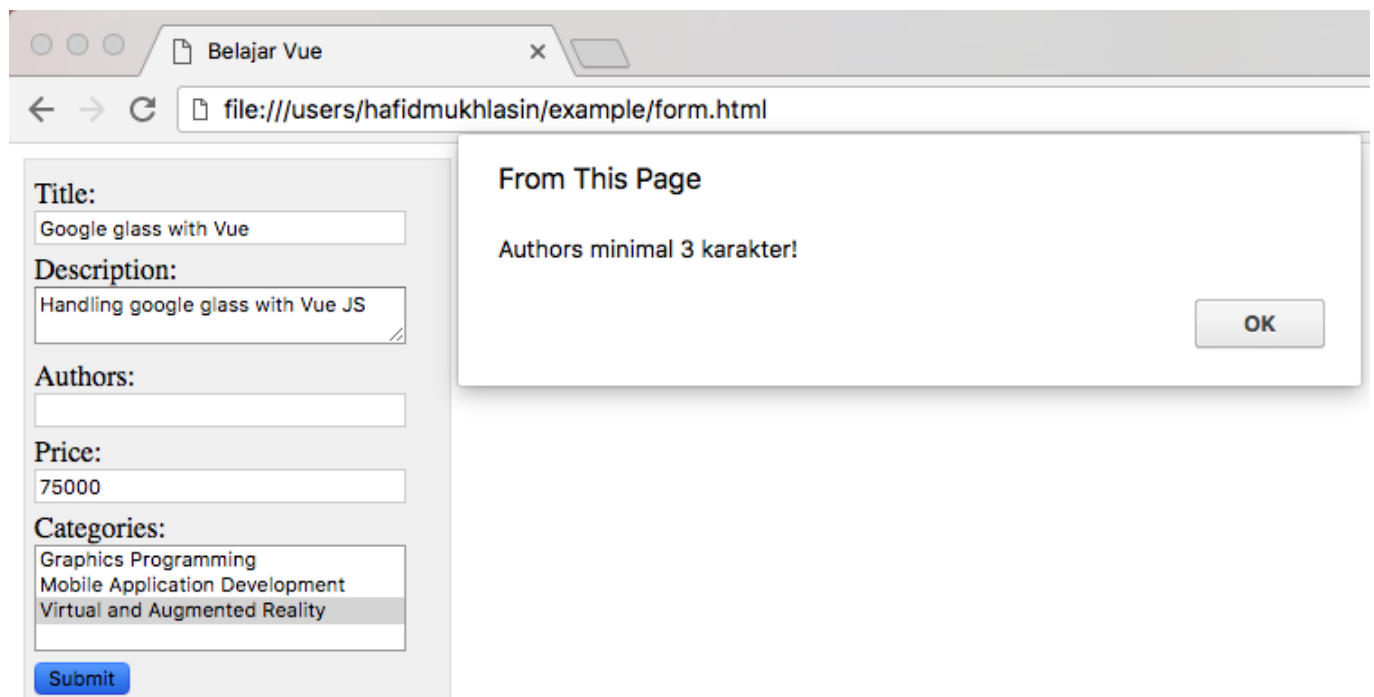
  if(this.title.length < 3){
    error++
    alert('Title minimal 3 karakter!')
  }
  else if(this.description.length > 500){
    error++
    alert('Description maximal 500 karakter!')
  }
  else if(this.authors.length < 3){
    error++
    alert('Authors minimal 3 karakter!')
  }
  else if(this.price < 0){
    error++
    alert('Price tidak boleh minus!')
  }
  else if(this.categories.length === 0){
```

```
        error++
        alert('Pilih minimal 1 category!')
    }

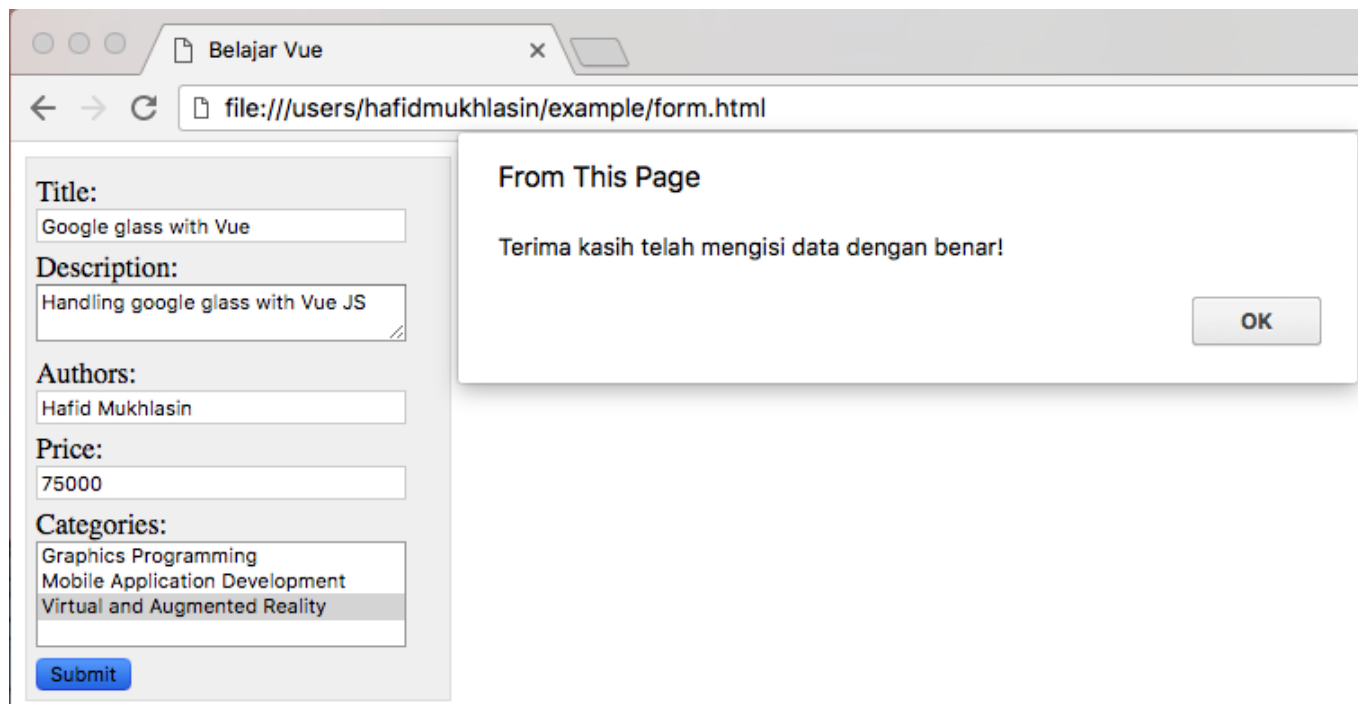
    if( error === 0 ){
        alert('Terima kasih telah mengisi data dengan benar!')
        // kirim data ke server
    }

    event.preventDefault()
}
```

Berikut hasilnya jika ada field yang tidak memenuhi minimal persyaratan



Dan berikut ini jika semua semua field telah memenuhi persyaratan.



Kita bisa juga menambahkan method `setFocus` pada input yang belum memenuhi syarat. Sehingga akan memudahkan dari sisi user. Caranya dengan menambahkan directive `ref` sebagai penanda unik pada field input.

```
<input type="text" name="title" ref="title" v-model="title">
```

Melalui directive tersebut kemudian bisa kita akses dengan kode berikut.

```
if(this.title.length < 3){
  error++
  this.$refs.title.focus()
  alert('Title minimal 3 karakter!')
}
```

Bisa juga dengan kode ini `this.$refs.title.select()`.

Catatan: tambahkan juga directive `ref` pada form supaya lebih mudah nantinya menangani elemen form tersebut.

Supaya lebih user friendly, pesan **error** tidak kita munculkan dalam bentuk **alert**, melainkan dalam bentuk teks di browser. Kita bisa kombinasikan dengan teknik list untuk menampilkan **error** yang akan kita tampung dalam bentuk array.

Mari kita ubah kode konstruktor Vue menjadi sebagai berikut.

```
var vm = new Vue({
  el: '#app',
  data: {
```

```

        title: 'Google Glass with VueJS',
        description: 'Control Google Glass with VueJS',
        authors: 'Hafid Mukhlasin',
        price: 75000,
        categories: [],
        options: [
            { text: 'Graphics Programming', value: '01' },
            { text: 'Mobile Application Development', value: '02' },
            { text: 'Virtual and Augmented Reality', value: '03' }
        ],
        errors: []
    },
    methods: {
        submitForm(event){
            this.errors = []
            if(this.title.length < 3){
                this.errors.push('Title minimal 3 karakter!')
                this.$refs.title.select()
            }
            if(this.description.length > 500){
                this.errors.push('Description maximal 500 karakter!')
                this.$refs.description.select()
            }
            if(this.authors.length < 3){
                this.errors.push('Authors minimal 3 karakter!')
                this.$refs.authors.select()
            }
            if(this.price < 0){
                this.errors.push('Price tidak boleh minus!')
                this.$refs.price.select()
            }
            if(this.categories.length === 0){
                this.errors.push('Pilih minimal 1 category!')
                this.$refs.categories.focus()
            }

            if( this.errors.length === 0 ){
                alert('Terima kasih telah mengisi data dengan benar!')
                // kirim data ke server
            }
        }
    }
})

```

Kemudian pada template, kita ubah menjadi sebagai berikut.

```

<form ref="formBook" @submit.prevent="submitForm($event)"
action="http://example.com/add-product" method="post">

    <p v-if="errors.length">
        <b>Please correct the following error(s):</b>
        <ul>

```



```
        <li v-for="error in errors">{{ error }}</li>
      </ul>
    </p>

    <label>Title:</label>
    <input name="title" ref="title" type="text" v-model="title">

    <label>Description:</label>
    <textarea name="description" ref="description" v-model="description">
  </textarea>

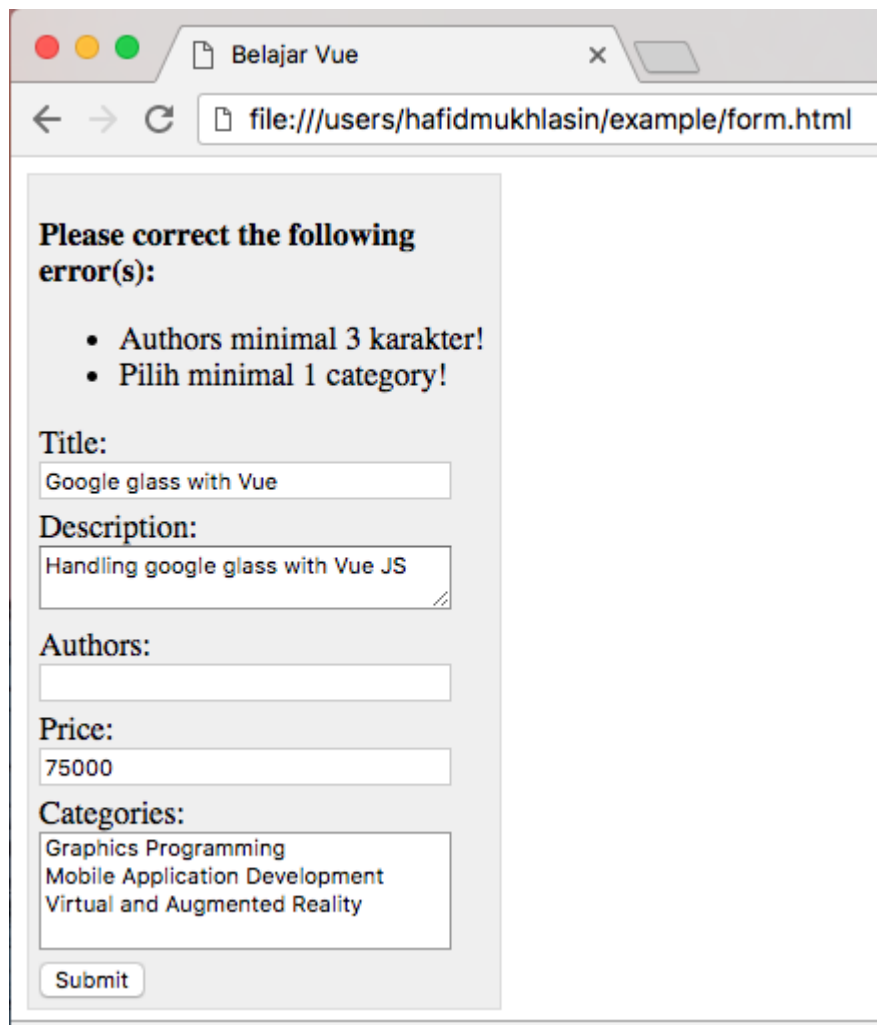
  <label>Authors:</label>
  <input name="authors" ref="authors" type="text" v-model="authors">

  <label>Price:</label>
  <input name="price" ref="price" type="number" v-model.number="price">

  <label>Categories:</label>
  <select name="categories" ref="categories" v-model="categories"
multiple>
    <option v-for="option in options" :value="option.value">
      {{ option.text }}
    </option>
  </select>

  <label></label>
  <input type="submit" value="Submit">
</form>
```

Kode di atas akan menghasilkan tampilan berikut.



Please correct the following error(s):

- Authors minimal 3 karakter!
- Pilih minimal 1 category!

Title:

Description:

Authors:

Price:

Categories:

Catatan: tutorial ini hanya menunjukkan tentang bagaimana validasi itu bekerja, selanjutnya tentu kamu bisa gunakan berbagai cara untuk memastikan bahwa input data user sesuai dengan yang diharapkan. Disamping cara manual ini, ada beberapa pustaka Vue yang bisa kita gunakan untuk memudahkan kita melakukan validasi data form, diantaranya:

- <https://github.com/monterail/vuelidate>
- <http://vee-validate.logaretm.com>

Peringatan: jangan lupa bahwa validasi ini hanya dari sisi client yang sangat rawan untuk dimanipulasi oleh user "nakal". Oleh karena itu validasi dari sisi server, merupakan hal mutlak yang harus kita lakukan untuk memastikan bahwa apa yang diinput oleh user itu sesuai dengan harapan kita.

Prepare Data Submit

Setelah validasi form dilakukan, langkah berikutnya adalah mempersiapkan data sebelum dikirimkan ke server. Kita bisa menggunakan object FormData (<https://developer.mozilla.org/en-US/docs/Web/API/FormData/FormData>) untuk mem-pack data hasil isian form menjadi sebuah object.

```
let formData = new FormData();  
// tambahkan satu persatu field  
formData.append('username', 'Chris');
```

Berikut ini implementasi pada kasus kita.

```
if( this.errors.length === 0 ){
  alert('Terima kasih telah mengisi data dengan benar!')
  // persiapkan data
  let formData = new FormData()
  formData.append('title', this.title)
  formData.append('description', this.description)
  formData.append('authors', this.authors)
  formData.append('price', this.price)
  formData.append('categories', this.categories)

  // kirim data ke server
}
```

Objek formData inilah yang akan dikirimkan ke server atau diproses lebih lanjut.

Di samping cara di atas yaitu manual satu persatu dalam memasukkan data field, FormData juga menyediakan cara cepat untuk memasukkan semua data field yang dimiliki form ke dalam objek formData. Berikut yang kita dapat dari dokumentasi FormData.

```
let myForm = document.getElementById( 'myForm' );
formData = new FormData(myForm);
```

Dengan sedikit modifikasi maka implementasi pada kasus kita menjadi sebagai berikut

```
// persiapkan data
let formBook = this.$refs.formBook
formData = new FormData(formBook);
// kirim data ke server
```

Selesai.

Catatan: untuk bisa menggunakan cara singkat ini syaratnya satu yaitu field pada form harus ada atribut `name`-nya.

Send Data To Server

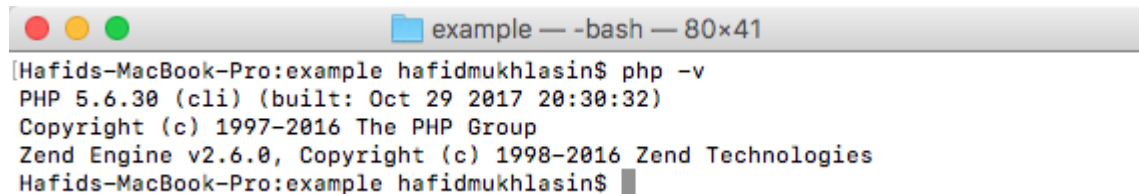
Setelah data di-*bundle* dalam satu objek formData, maka data siap untuk dikirim ke server. Pada bagian ini, kita akan mensimulasikan pengiriman data form ke server menggunakan PHP native. Oleh karenanya siapkan web server (nginx atau apache) serta PHP untuk dapat mencoba simulasi ini, atau bisa juga dengan menggunakan PHP built-in server.

Catatan: pada bab ini tidak akan dijelaskan tentang bagaimana instalasi PHP, dan Web Server. kamu bisa menggunakan panduan lain untuk menginstalasinya atau merujuk langsung ke website resmi PHP (<https://php.net>)

Pada tutorial ini, kita akan menggunakan built-in web server bawaan PHP. Oleh karena itu, pastikan PHP sudah terinstalasi dengan baik sehingga bisa diakses melalui terminal (command prompt atau powershell pada Windows).

Jalankan perintah

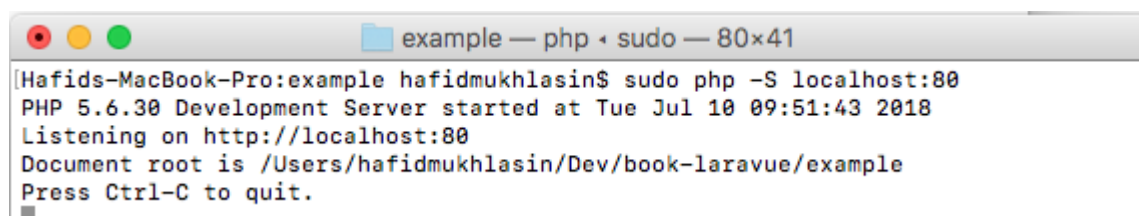
```
php -v
```



```
example — -bash — 80x41
[Hafids-MacBook-Pro:example hafidmukhlasin$ php -v
PHP 5.6.30 (cli) (built: Oct 29 2017 20:30:32)
Copyright (c) 1997-2016 The PHP Group
Zend Engine v2.6.0, Copyright (c) 1998-2016 Zend Technologies
Hafids-MacBook-Pro:example hafidmukhlasin$
```

Kemudian jalankan PHP built-in server dengan menggunakan format perintah berikut:

```
php -S localhost:80
```



```
example — php -S localhost:80 — 80x41
[Hafids-MacBook-Pro:example hafidmukhlasin$ sudo php -S localhost:80
PHP 5.6.30 Development Server started at Tue Jul 10 09:51:43 2018
Listening on http://localhost:80
Document root is /Users/hafidmukhlasin/Dev/book-laravue/example
Press Ctrl-C to quit.
```

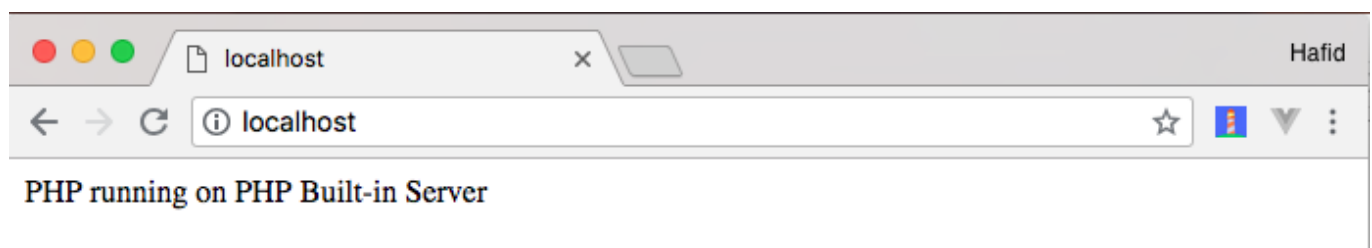
Catatan: 80 adalah nomer port dari webserver, kita bebas mengubahnya dengan nomer lain yang sedang tidak digunakan.

Perintah ini akan menjalankan web server dan menjadikan current directory sebagai web root.

Untuk menguji apakah kode PHP dapat berjalan dengan baik, maka buat file index.php pada **current directory** (pada contoh ini penulis meletakkan pada direktori yang sama dengan file form.html atau kode Vue). Isinya sebagai berikut.

```
<?php
echo "PHP running on PHP Built-in Server";
```

Jalankan pada browser alamat <http://localhost:80>, maka hasilnya.



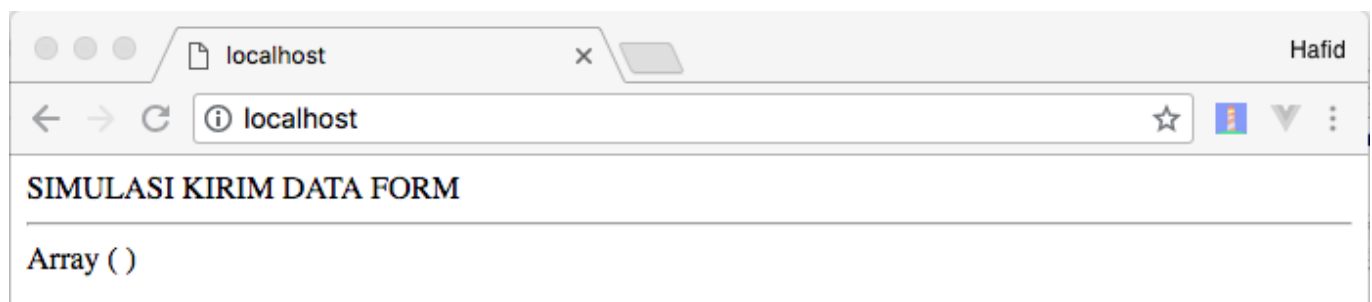
Jika berhasil, maka selanjutnya kita akan membuat kode PHP yang bertugas sebagai **endpoint** penerima data kiriman dari form dan mengembalikan data tersebut dalam bentuk array. Berikut ini kodenya.

```
<?php
// untuk mencegah error akibat CORS
header("Access-Control-Allow-Origin: *");
header('Access-Control-Allow-Credentials: true');
header("Access-Control-Allow-Methods: GET, POST, OPTIONS");

echo "SIMULASI KIRIM DATA FORM <hr>";

// menampilkan data yang dikirimkan dengan method post
print_r($_POST);
```

Kemudian coba akses melalui browser.



Catatan: fungsi CORS pada contoh ini digunakan agar Vue melalui pustaka HTTP client dapat mengakses file PHP yang diletakkan pada server yang berbeda dengan server dimana Vue di-*hosting*. Selengkapnya silakan baca di <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

Untuk lebih mudahnya, pindahkan juga atau atur agar file-file Vue yang telah kita buat sebelumnya juga berada pada direktori yang sama dengan file index.php

Langkah selanjutnya adalah membuat kode Javascript/Vue untuk mengirimkan data ke server. Ada berbagai cara untuk melakukan request data ke server, diantaranya dengan engine XMLHttpRequest, fungsi fetch (fungsi native Javascript), dan pustaka axios.

Pada bagian ini, kita hanya akan membahas tentang cara request ke server dengan menggunakan XMLHttpRequest. Jadi XMLHttpRequest adalah engine yang digunakan untuk menangani permintaan data ke dan dari server. Engine ini cukup populer digunakan terutama untuk menjalankan AJAX.

```
if( this.errors.length === 0 ){
  //alert('Terima kasih telah mengisi data dengan benar!')

  // persiapan data
  let formBook = this.$refs.formBook
  formData = new FormData(formBook);

  // kirim data ke server
  let xhttp = new XMLHttpRequest() // create objek XMLHttpRequest

  // definisikan fungsi ketika terjadi perubahan state
  xhttp.onreadystatechange = function() {
    // state ini menunjukkan data terkirim dan diterima server dengan
```

```

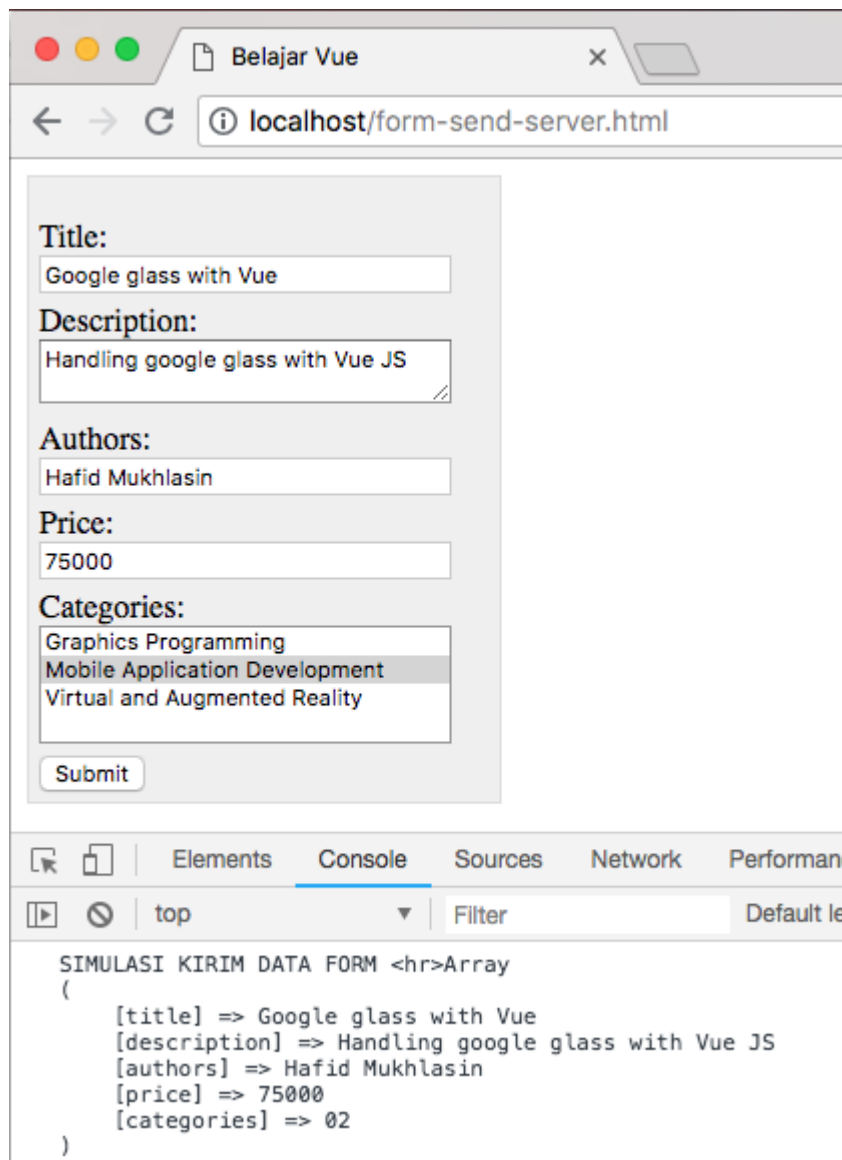
baik
    if (this.readyState == 4 && this.status == 200) {
        // respon text dari server
        console.log(this.responseText)
    }
}
// sesuaikan dengan lokasi file index.php di lokasi komputer kamu
xhr.open("POST", "http://localhost/index.php", true)

// bisa juga langsung nama filenya jika berada dalam satu folder yang
sama
// xhr.open("POST", "index.php", true)

// kirim objek formData
xhr.send(formData)
}

```

Mari kita ujicoba kode di atas pada browser.



Handling File Upload

Pada sisi client, penanganan field bertipe file pada form pada dasarnya hampir sama saja dengan field bertipe lain.

Sebagai simulasi, kita gunakan kode sebelumnya. Pada template form tambahkan field input bertipe file.

```
<label>Cover:</label>
<input name="cover" ref="cover" type="file">
```

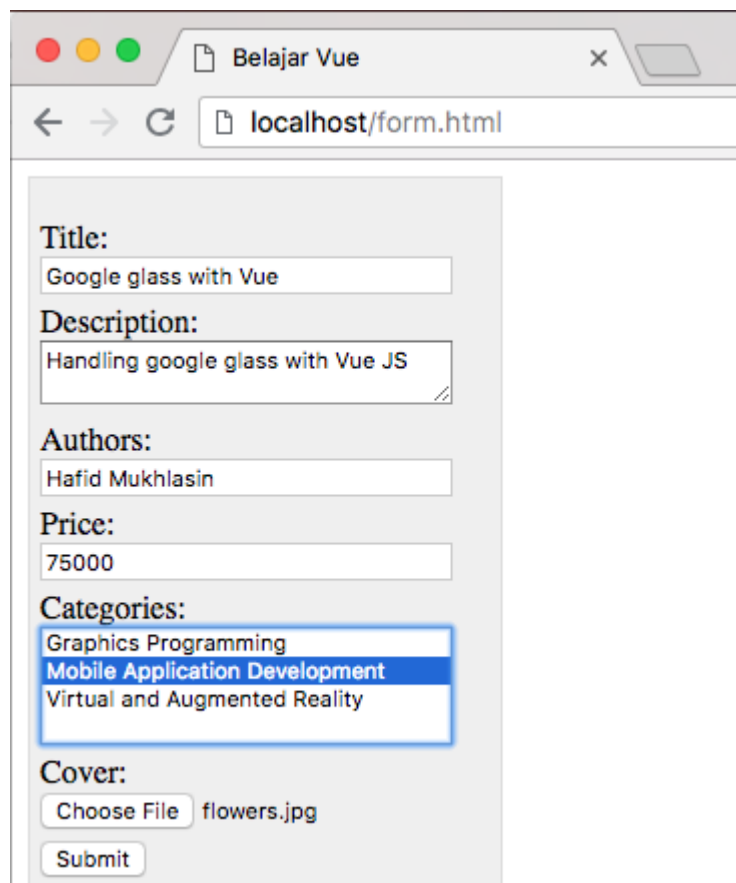
Kemudian pada kode Javascript-nya tambahkan kode berikut

```
// get file yang dibrowse user
let cover = this.$refs.cover.files[0]
// tambahkan ke object formData
formData.append("cover", cover);
```

Dengan cara ini maka file akan terkirim ke server. Untuk mensimulasikannya, edit file `index.php` dan tambahkan kode berikut.

```
// ...
print_r($_FILES['cover']);
```

Hasilnya.

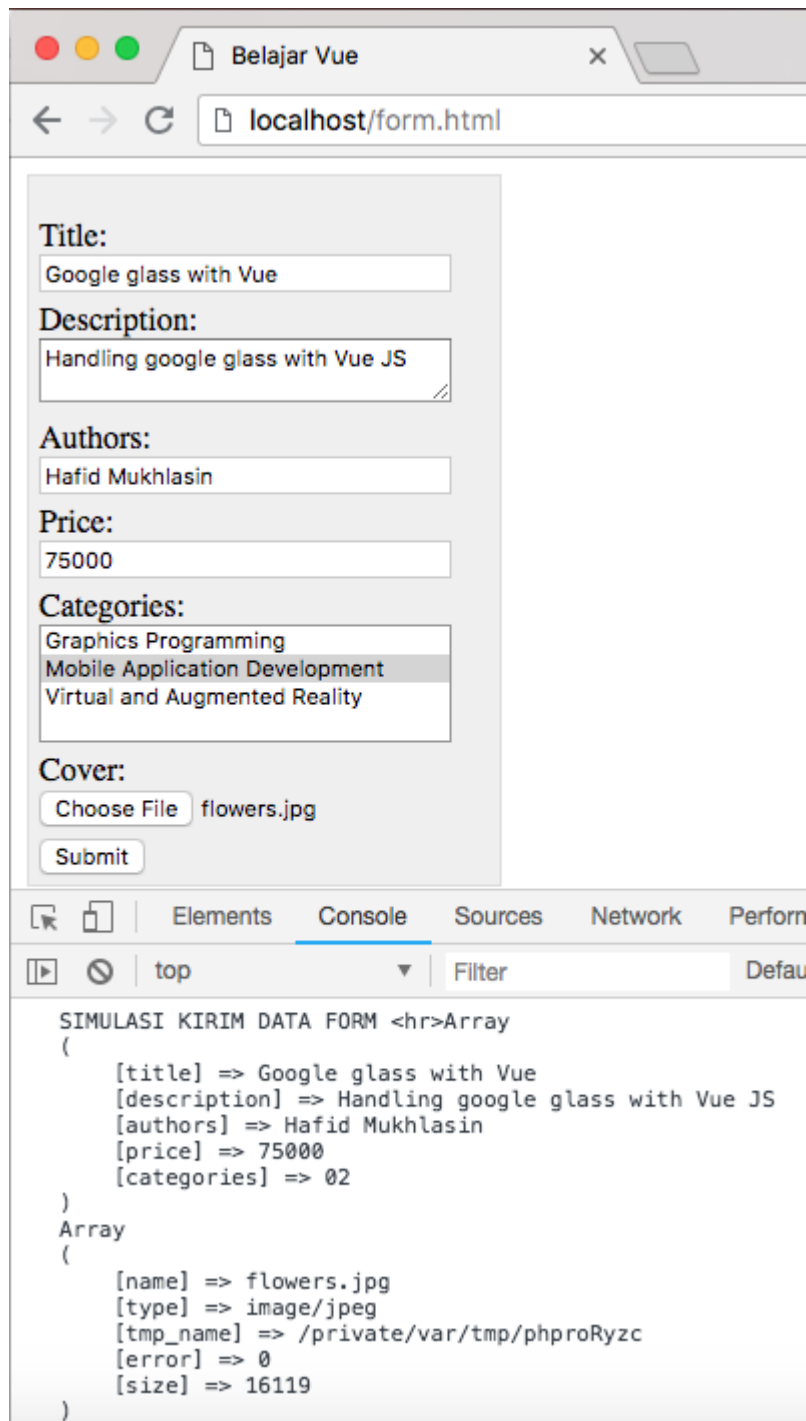


The screenshot shows a web browser window with the title 'Belajar Vue' and the address bar displaying 'localhost/form.html'. The form contains the following fields:

- Title:** Google glass with Vue
- Description:** Handling google glass with Vue JS
- Authors:** Hafid Mukhlisin
- Price:** 75000
- Categories:** A dropdown menu is open, showing three options: 'Graphics Programming', 'Mobile Application Development' (which is highlighted in blue), and 'Virtual and Augmented Reality'.
- Cover:** A file upload field showing 'Choose File' and 'flowers.jpg'.

At the bottom of the form is a 'Submit' button.

Ketika form disubmit maka pada console log akan terlihat sebagai berikut.



Kesimpulan

Form merupakan media yang digunakan user untuk berinteraksi dengan aplikasi atau web. User dapat menginput data dalam bentuk teks, array, atau file. Validasi merupakan hal yang harus kita lakukan sebab kita tidak tau apakah user benar-benar memasukkan data sesuai dengan harapan kita atau tidak, sekaligus dalam rangka keamanan data.

Pengiriman data ke server menggunakan Javascript membutuhkan tools http client, pada bab ini dibahas mengenai penggunaan tools XMLHttpRequest yang biasa digunakan untuk AJAX. Pilihan lainnya, kita bisa gunakan Fetch (native Javascript) atau yang direkomendasikan Vue yaitu pustaka Axios.

Pada bab berikutnya, kita akan belajar tentang komponen yang merupakan bagian dari Vue yang cukup penting untuk dipelajari guna memudahkan kita dalam mengembangkan aplikasi yang kompleks.

Ayoo lebih semangat lagi!