Table of Contents

Introduction 1.1
Overview 1.2
Persiapan 1.3
Hello World! 1.4
Mengenal Directive 1.5
    v-bind 1.5.1
    v-model 1.5.2
    v-text 1.5.3
    v-html 1.5.4
    v-on 1.5.5
    v-if 1.5.6
    v-show 1.5.7
    v-for 1.5.8
Mengenal Component 1.6
    Basic Component pada Vue.js 1.6.1
Menggunakan vue-cli untuk scaffolding 1.7
Mengenal Single File Component 1.8
Daftar Produk 1.9
Tambahkan Produk 1.10
Update Produk 1.11
Hapus Produk 1.12
Routing Dengan Vue Router 1.13
Melakukan Request ke API 1.14
Menggunakan Flux Pattern dengan Vuex 1.15
Mengenal Vue.js 1.16

Vue.js Workshop Bahasa Indonesia

Vue.js Workshop Bahasa Indonesia merupakan sebuah tutorial sederhana untuk membangun aplikasi menggunakan Vue.js versi 2.x.x . Workshop ini merupakan versi bahasa Indonesia dari Jayway Vue.js Workshop dengan sedikit modifikasi dan penyesuaian menjadi lebih detail dan sederhana, sehingga dapat lebih mudah diikuti atau dipelajari.

Diharapakan dengan mengikuti Vue.js Workshop Bahasa Indonesia ini anda dapat memahami berbagai macam teknik untuk mengembangkan sebuah aplikasi menggunakan Vue.js bergantung pada skala aplikasi yang anda buat, serta menghadapi berbagai permasalahan dan tantangan dalam pengembangan aplikasi menggunakan Vue.js 2.x.x.

Overview

Pada workshop kali ini kita akan mempelajari tentang bagaimana cara membuat sebuah aplikasi CRUD Management Produk sederhana. Disini anda akan membuat setiap component dari awal.

Anda akan mempelajari bagaimana untuk :

- Membuat aplikasi web dari awal menggunakan Vue.js 2.x.x
- Menggunakan vue-router utnuk melakukan routing pada setiap bagian dari aplikasi anda.
- Membuat unidirectional flow atau flux pattern menggunakan vuex untuk memisahkan data dan tampilan
- Menghubungkan aplikasi web kita dengan backend REST API menggunakan axios sebagai http request library

---

Key to perfection

Selama proses workshop akan ada beberap bagian "**Key to perfection**". Yang merupakan tugas dengan beberapa petunjuk yang berkaitan dengan permasalahan dunia nyata. Tugas-tugas tersebut dapat anda kerjakan apabila anda menginginkan tantangan lebih.

Persiapan

Sebelum kita memulai membuat aplikasi kita mengguanakan Vue.js, berikut ini adalah beberapa hal yang kita perlukan untuk mulai bekerja dengan baik :

- Node.js versi 6 ke atas.
- Code editor pilihan masing-masing, (disarankan menggunakan Atom dengan package Vue-component languange yang terinstall).
- Web Browser Modern (Disarankan mengguakan Google Chrome dengan Vue Dev Tools yang sudah terinstall)

Hello World!

Salah satu kelebihan paling utama dari Vue.js adalah flexibilitas dimana kita dapat membuat sebuah web application dengan berbagai cara tergantung pada kebutuhan dan skala dari web application kita.

Kita akan memulai dengan membuat sebuah aplikasi "Hello World!" yang sangat sederhana. Untuk memulai buatlah sebuah file html dengan code sebagai berikut :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Hello world!</title>
  <!-- memasukkan Vue.js dari CDN unpkg-->
  <script src="https://unpkg.com/vue"></script>
  <script>
    // menunggu hingga element <body> selesai dimuat
    // pada jQuery sama seperti $(document).ready()
    document.addEventListener('DOMContentLoaded', function () {
      // membuat instance vue yang baru dengan element template serta
      // data atau state yang perlu ditampilkan
      new Vue({
        el: '#app',
        data: {
          message: 'Hello World!'
        }
      })
    })
  </script>
</head>
<body>
  <div id="app">
    {{ message }}
  </div>
</body>
</html>
```

Bukan file tersebut di web browser anda, dan lihat tampilan yang dimunculkan pada web browser tersebut.

Itu semua merupakan hal yang kita perlukan untuk mulai membuat sebuah aplikasi Vue.js sederhana dengan memisahkan bagian data atau state dengan bagian view atau presentation.

Sekalipun kedepannya kita akan mencoba untuk membagun web application yang lebih kompleks, dengan contoh "Hello World!" saya ingin menunjukkan bahwa sebenarnya sangat mudah untuk memulai membuat aplikasi dengan Vue.js dibandingkan dengan menggunakna framework yang lain. Seperti yang sudah diterangkan pada bagian sebelumnya, salah satu kelebihan dari Vue.js dibanding framework lain adalah fleksibilitas dimana kita bisa memulai project dari yang paling sederhana dan menambahkan beberapa add-ons official seperti vue-router dan vuex.

Mengenal Directive (v-*)

Salah satu fitur yang dimiliki oleh Vue.js adalah penggunaan **_directive.** _Directive sendiri merupakan sebuah fitur yang memungkinkan kita untuk menambahkan sebuah special attribute pada tag HTML dari web application kita. Special attribute tersebut kemudian akan memberitahu library atau framework yang kita gunakan untuk melakukan manupilasi terhadap DOM. Pada Vue.js sendiri setiap attribute directive yang biasa digunakan memiliki awalan atau prefix `v-` .

Contoh Sederhana :

```
<p v-text="message"></p>
```

Hello world

Hello world

CATATAN! Script lengkap lihat di sini.

Pada contoh script di atas terdapat sebuah tag `<p>` yang memiliki attributes `v-text` . Bisa kita lihat bahwa tag <p> tersebut tidak memiliki isi content. Akan tetapi karena memiliki attribute `v-text` yang value-nya merujuk ke state `message` , maka secara otomatis Vue.js akan melakukan manipulasi DOM dengan menambahkan isi content pada tag `<p>` tersebut sesuai dengan value dari state `message` script Vue.js kita.

Secara teknis attribute `v-text` akan mengupdate value `textContent` setiap saat value message di `ViewModel` atau `state` kita berubah.

Untuk passing sebuah value ke content dari sebuah element html kita juga dapat melakukan interpolation dengan menggunakan script double curly braces `{{ }}` seperti yang terlihat di screenshot di atas.

Daftar Lengkap Directive pada Vue.js

- `v-bind`
- `v-model`
- `v-text`
- `v-html`
- `v-on`
- `v-if`
- `v-else-if`
- `v-else`
- `v-show`
- `v-for`

v-bind

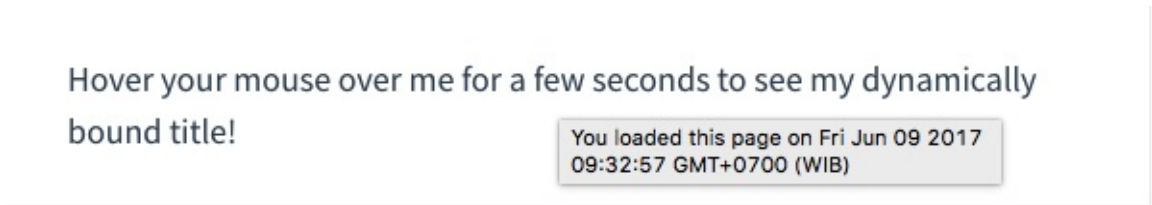
v-bind berfungsi untuk melakukan data binding dengan sebuah attribute dari sebuah tag HTML. Contoh penggunaannya sebegai berikut :

```
<div id="app">
  <span v-bind:title="message">
    Hover your mouse over me for a few seconds
    to see my dynamically bound title!
  </span>
</div>
```

```
new Vue({
  el: '#app',
  data: {
    message: 'You loaded this page on ' + new Date()
  }
})
```

Script di atas berfungsi untuk melakukan binding data dari attribute title dari tag span tersebut dengan value dari state message dari instance Vue. Jadi setiap kali terjadi perubahan value pada state `message` value pada attribute `title` tersebut juga akan ikut berubah menyesuaikan.

Hasil dari script di atas kurang lebih akan seperti pada gambar di bawah ini. DImana ketika kita melakukan mose hove di atas text tersebut akan muncul text sesuai dengan value di state `message` .



v-model

v-model berfungsi untuk melakukan two-way data binding yang umumnya dilakukan pada form input. DImana value pada input yang memiliki attribute directive v-model akan merujuk pada state yang menjadi reference-nya, dan juga setiap kali terjadi perubahan value pada form input tersebut, value dari state juga akan turut berubah.

```
<div id="app">
  <p>{{ message }}</p>
  <input v-model="message">
</div>
```

```
new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

Contoh script lengkap dan hasilnya dapat dilihat di sini.

v-text

v-text berfungsi untuk melakukan binding sebuah value dari state yang berupa text ke dalam innerHTML dari sebuah element. Attribute directive ini memiliki fungsi yang mirip dengan penggunaan interpolation dengan double curly braces `{{ }}` .Contoh dari penggunaan v-text adalah sebagai berikut.

```
<p v-text="message"></p>
```

```
new Vue({
 el: '#app',
 data: {
  message: 'Hello Vue!'
 }
})
```

Contoh script lengkap dan hasilnya dapat dilihat di sini.

v-html

v-html memiliki fungsi yang hampir sama dengan v-text. Namun berbeda dengan v-text yang akan merender sebuah value dari sebuah state yang berupa plain text, v-html akan merender value dari state yang berupa kode HTML. Contoh penggunaan v-html adalah sebagai berikut :

```
<div v-html="message"></div>
```

```
new Vue({
 el: '#app',
 data: {
  message: '<p>Hello <strong>Vue!</strong></p>'
 }
})
```

Contoh script lengkap dan hasilnya dapat dilihat di sini.

v-on

Berbeda dengan attribute directive sebelumnya yang lebih fokus pada operasi binding data. Attribute v-on digunakan untuk melakukan event-handling. Attribute v-on akan untuk mengeksekusi atau menjalankan sebuah fungsi atau methods dari instance Vue yang diterima sebagai value dari directive atrribute ini. Sebelum menerima value, attribute ini juga menerima keypath yang akan menentukan jenis event yang akan dihandle seperti click, keyup, submit, dsb.

Contoh penggunaan-nya sebagai berikut :

```
<button type="button" v-on:click="showAlert">Click Me!</button>
```

```
new Vue({
 el: '#app',
 data: {
  message: 'Hello Vue!'
 },
 methods: {
  showAlert () {
   alert(this.message)
  }
 }
})
```

Contoh script lengkap dan hasilnya dapat dilihat di sini.

v-if

Attribute directive v-if digunakan untuk melakukan conditional rendering. Dimana tag atau element HTML yang memiliki attribute ini akan ditampilkan bedasarkan kondisi value pada state, atau sama sekali tidak dirender ketika kondisi value pada state tersebut false. Attribute ini dapat dikombinasikan dengan attribute v-else-if dan juga v-else.

Contoh penggunaan-nya sebagai berikut :

```
<h1 v-if="status !== true && status !== false">The status is ambiguous</h1>
<h1 v-else-if="status">The status is true</h1>
<h1 v-else>The status is false</h1>
```

```
new Vue({
 el: '#app',
 data: {
  status: 'I make it ambigous'
 }
})
```

Contoh script lengkap dan hasilnya dapat dilihat di sini.

v-show

v-show memiliki fungsi yang mirip dengan v-if, akan tetapi dengan sedikit perbedaan, dimana apabila kondisi yang menjadi parameter atau value dari v-if adalah false maka element tersebut tidak akan dirender sama sekali. Berbeda dengan v-show apabila kondisi parameter-nya tidak terpenuhi elemen tersebut akan tetap di render, hanya saja akan di hide dengan property display pada styling-nya menjadi none.

v-for

Attribute directive ini digunakan untuk melakukan list rendering dari list yang berdasarkan data yang memiliki bentuk array.

`v-for` membutuhkan syntax special yang berbentuk `item in items` , dimana `items` adalah sumber data array dan `item` adalah alias untuk setiap element dari array yang di iterasi.

Contoh penggunaannya sebagai berikut :

```
<ul>
  <li v-for="student in students">{{ student.name }}</li>
</ul>
```

```
new Vue({
  el: '#app',
  data: {
    students: [
      {
        name: 'Peter Parker',
        gender: 'male'
      },
      {
        name: 'Steve Rogers',
        gender: 'male'
      },
      {
        name: 'Natasha Romanov',
        gender: 'female'
      },
      {
        name: 'Tony Stark',
        gender: 'male'
      }
    ]
  }
})
```

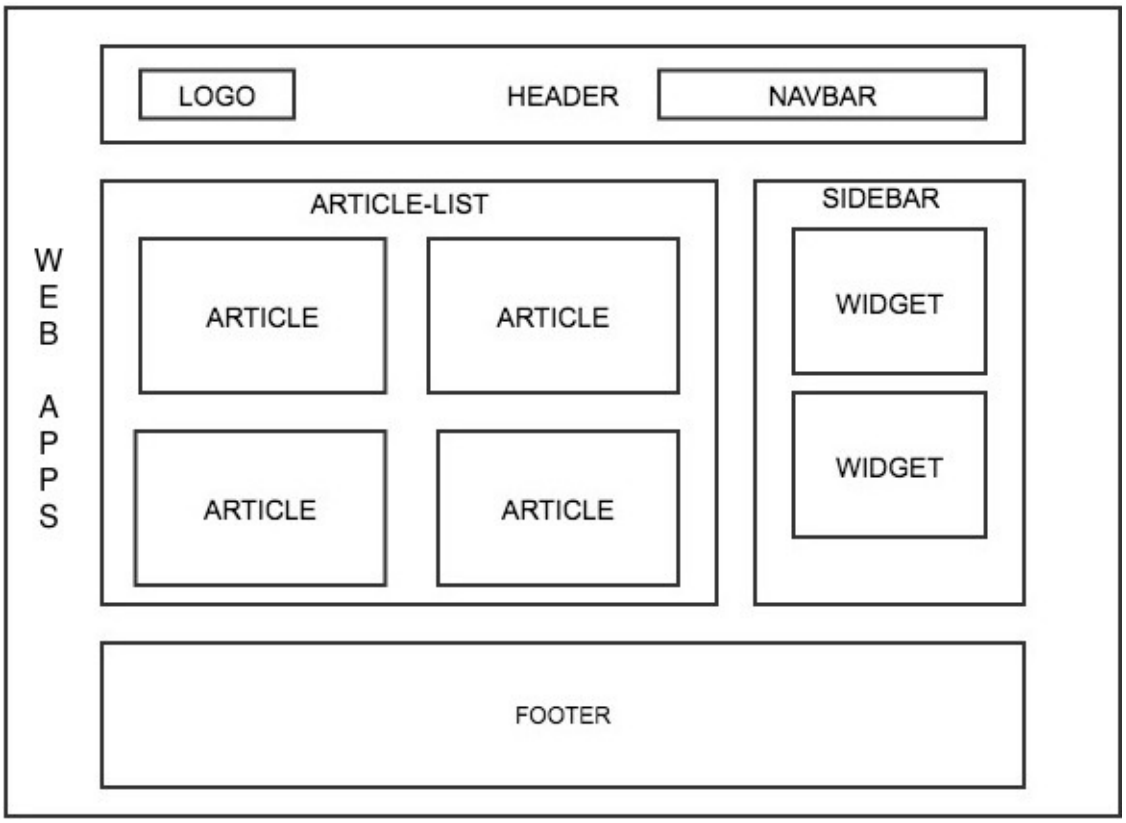
Contoh script lengkap dan hasilnya dapat dilihat di sini.

Mengenal Component

Salah satu fitur yang paling powerful dari Vue.js adalah Component. Component sendiri merupakan sebuah sistem yang memungkinkan kita untuk menambahkan atau meningkatkan fungsionalitas dari HTML. Dengan component kita dapat membungkus atau mengenkapsulasi beberapa tag HTML menjadi sebuah tag baru yang dapat digunakan berulang-ulan (reusable) pada sebuah web application.

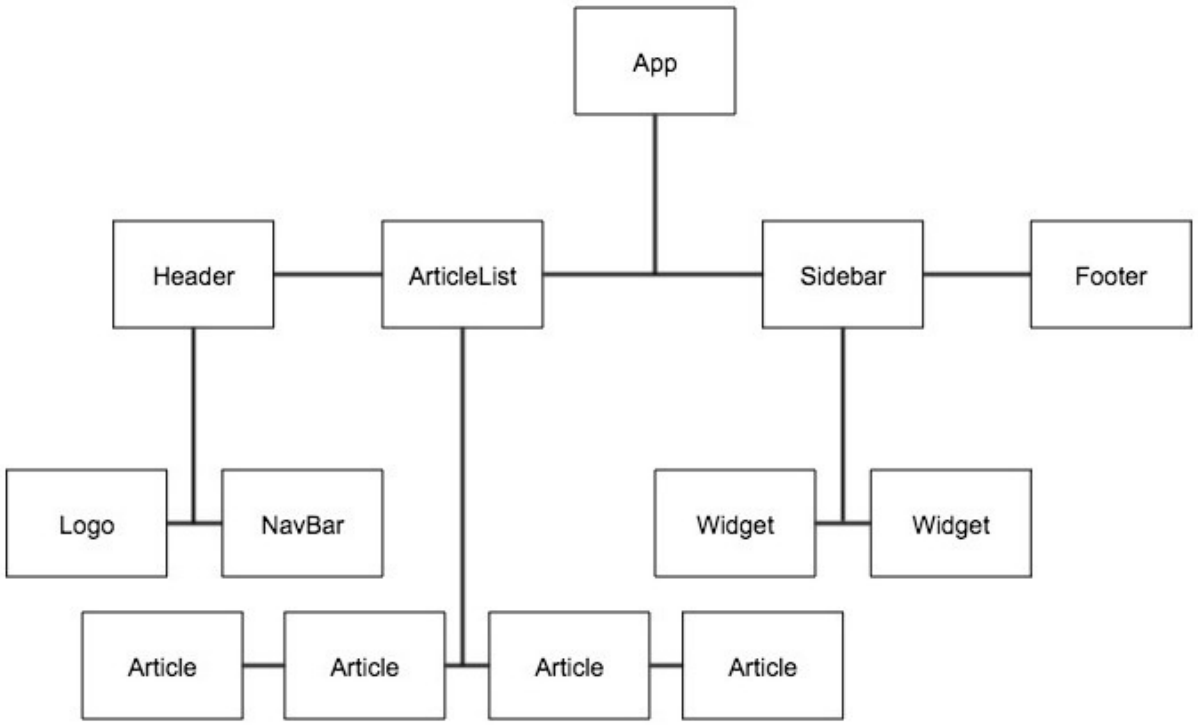
Pada umum nya di Vue.js setiap component merupakan sebuah custom element yang memiliki behavior atau sifat dari Vue compiler. Di mana pada setiap component memiliki `data/state` , `props` , `methods` , serta `lifecycle hooks` masing-masing.

Konsep Component



Pada dasarnya di dalam konsep component, setiap bagian dari sebuah aplikasi web merupakan sebuah component. Sebuah aplikasi web sendiri merupakan sebuah satu component besar yang menaungi bayak component lain. Dan sebuah component juga dapat terdiri dari banyak component lainnya. Seperti misal-nya pada gambar mock-up

halaman web di atas yang memiliki beberapa bagian seperti header, article-list, sidebar, dan footer. Di mana setiap bagian tersebut merupakan component, yang juga bisa saja terdiri dari berbagai component lain.



Terdapat konsep parent-child para web applications yang dibangun berdasarkan konsep component itu sendiri. Di mana sebuah component besar yang menaungi dan terdiri atau tersusun dari banyak component lain merupakan component parent, dan juga component kecil yang dinaungi dan merupakan bagian dari penyusun sebuah component yang lebih besar merupakan component children. Dari konsep ini dapat digambarkan sebuah component-tree yang memperlihatkan hierarki atau kedudukan masing-masing component.

Basic Component pada Vue.js

Global Registration

Untuk membuat sebuah component pada Vue.js sendiri cukup jelas dan cukup mudah. Untuk mendaftarkan sebuah global component, kita dapat menggunakan script `Vue.component(tagName, options)` . Sebagai contoh:

```
Vue.component('my-component', {
    //options
})
```

Setelah diregistrasi sebagai component global, component tersebut dapat kita gunakan sebagai sebuah custom HTML element dalam sebuah Vue Instance. Pastikan untuk meregistrasi sebuah component sebelum inisialisasi Vue Instance.

```
<div id="app">
  <my-component></my-component>
</div>
```

```
// register
Vue.component('my-component', {
  template: '<div>A custom component!</div>'
})
// create a root instance
new Vue({
  el: '#app'
})
```

Hasil render akan sebagai berikut :

```
<div id="app">
  <div>A custom component!</div>
</div>
```

Local Registration

Selain meregistrasi setiap component secara global, kita juga dapat meregistrasi component secara local dimana sebuah component hanya dapat digunakan dalam instance Vue tertentu sesuai dengan yang sudah kita definisikan menggunaka option `components` pada instance Vue.

```
var Child = {
  template: '<div>A custom component!</div>'
}
new Vue({
  // ...
  components: {
    // <my-component> will only be available in parent's template
    'my-component': Child
  }
})
```

State/Data pada Component

Seperti yang sudah disampaikan di bab sebelumnya, setiap component memiliki `data/state` , `props` , `methods` , serta `lifecycle hooks` masing-masing.

Kali ini kita akan mempelajari cara untuk menambahkan state atau data pada sebuah Vue component. Sedikit hal yang berbeda dari data pada Vue Instance dengan data pada sebuah Component adalah bentuknya. Jika pada sebuah Vue Instance data berupa sebuah object, maka berbeda pada component, di mana data memiliki bentuk sebuah function yang me-return object.

Apabila kita membuat data dalam bentuk object pada sebuah component seperti dibawah ini :

```
Vue.component('my-component', {
  template: '<span>{{ message }}</span>',
  data: {
    message: 'hello'
  }
})
```

Maka Vue akan memunculkan pesan ***warning*** *_yang menerangkan bahwa state pada setiap component harus berupa function yang me-_return _object. Hal ini dimaksudkan untuk membuat setiap state tersebut bersifat _scoped* atau terbatas pada component itu saja. Sehingga apabila terjadi perubahan data pada sebuah component tidak akan mempengaruhi component lain.

Bentuk penerapan state atau data yang benar pada sebuah component :

```
Vue.component('my-component', {
  template: '<span>{{ message }}</span>',
  data: function () {
    return {
      message: 'hello'
    }
  }
})
```

Contoh penerapan state pada component di sini.

Props pada Component

Terkadang kita memperlukan data yang dinamis pada sebuah component, di mana kita butuh untuk menampilkan component yang sama secara berulang-ulang hanya saja dengan isi data yang berbeda. Contoh nyatanya misal dalam sebuah aplikasi todo, kita bisa memanggil component *task-item _yang akan memuat informasi singkat tentang beberapa _task* yang berbeda yang ada pada sebuah *todo--list*.

Untuk permasalahan seperti ini kita dapat memanfaatkan fitur *props* atau yang ada pada Vue.js. Props merupakan sebuah custom attribute pada component yang dapat kita gunakan untuk mengirim dan/atau menerima data antar component.

Contoh penerapan props adalah seperti berikut:

```
Vue.component('todo-item', {
    // dalam setiap component bisa terdapat
    // lebih dari satu props,
    // setiap props dapat dideklarasikan sebagai
    // string atau object
    props: ['task'], // pada component todo-item terdapat props 'task'
    template: '<li>{{ task.title }}</li>
})
```

Component `todo-item` itu kemudian dapat kita panggil di dalam component lain. Misal sebagai berikut :

```
Vue.component('todo-list', {
    // di bawah ini kita melooping component todo-item
    // berdasarkan object task yang ada dalam array state task
    // lalu melakukan binding props task pada todo-item
    // dengan data task hasil looping
    template: `
    <ul>
      <todo-item v-for="task in tasks" :task="task"></todo-item>
    </ul>
    `,
    data: function () {
      return {
        task: [
          { id: 0, title: 'Pergi ke pasar', detail: 'Beli buah'},
          { id: 1, title: 'Ngerjain PR', detail: 'PR matematika halaman 13'}
        ]
      }
    }
})
```

WIP . . .

Menggunakan vue-cli untuk scaffolding project

Untuk memulai sebuah project yang sederhana apa yang kita butuhkan hanyalah beberapa file HTML, CSS, dan Javascript. Akan tetapi ketika kita mulai membuat project dengan skala yang lebih besar maka kita perlu memikirkan beberapa hal seberti asset management, minification, module bundling, linting, testing, dsb.

Untuk membantu mengatasi hal itu, terdapat sebuah command line tool yang disebut vue-cli yang dapat anda gunakan untuk membuat scaffolding atau kerangka sebuah project Vue.js baru dengan berbagai teknologi web modern yang biasa digunakan di masa kini.

Untuk menginstall vue-cli jalankan perintah :

```
npm i -g vue-cli
```

Setelah vue-cli terinstall, anda dapat mulai membuat scaffolding untuk project Vue.js anda cukup dengan satu perintah :

```
vue init <template> <nama_project>
```

Untung melanjutkan ke tahap berikutnya dari workshop ini mulailah membuat sebuah scaffold atau kerangka project Vue.js baru menggunakan vue-cli dengan mengetikkan perintah di atas. Kemudian anda akan diberikan berbagai macam opsi untuk kerangka project anda. Sebagai contoh :

```
❯❯❯❯ vue init webpack vuejs-workshop-id                ~/D/tamvanslab

? Project name vuejs-workshop-id
? Project description A Vue.js project
? Author Septian A Tama <sfirst.tama@gmail.com>
? Vue build standalone
? Install vue-router? Yes
? Use ESLint to lint your code? Yes
? Pick an ESLint preset Standard
? Setup unit tests with Karma + Mocha? Yes
? Setup e2e tests with Nightwatch? Yes

   vue-cli · Generated "vuejs-workshop-id".

   To get started:

     cd vuejs-workshop-id
     npm install
     npm run dev

   Documentation can be found at https://vuejs-templates.github.io/webpack
```

Tampilan di atas adalah proses step-by-step dalam menentukan detail dan berbagai opsi untuk kerangka project anda. Pada contoh tersebut saya mengaktifkan setiap opsi yang tersedia, termasuk menginstall vue-router, ESLint dengan format Standard Js, dan menyiapkan package untuk unit testing menggunakan Karma dan Mocha, serta end-to-end testing menggunakan Nightwatch. Berikut kurang lebih struktur file dan direktori sebuah kerangka project Vue.js dengan template webpack secara lengkap :

```
.
├── README.md
├── build
│   ├── build.js
│   ├── check-versions.js
│   ├── dev-client.js
│   ├── dev-server.js
│   ├── utils.js
│   ├── vue-loader.conf.js
│   ├── webpack.base.conf.js
│   ├── webpack.dev.conf.js
│   ├── webpack.prod.conf.js
│   └── webpack.test.conf.js
├── config
│   ├── dev.env.js
│   ├── index.js
│   ├── prod.env.js
│   └── test.env.js
├── index.html
├── package.json
├── src
│   ├── App.vue
│   ├── assets
│   │   └── logo.png
│   ├── components
│   │   └── Hello.vue
│   ├── main.js
│   └── router
│       └── index.js
├── static
└── test
    ├── e2e
    │   ├── custom-assertions
    │   ├── nightwatch.conf.js
    │   ├── runner.js
    │   └── specs
    └── unit
        ├── index.js
        ├── karma.conf.js
        └── specs

13 directories, 26 files
```

PENTING!

Vue-cli mendukung berbagai macam template untuk kerangka project baru anda. Pada project kali ini kita akan menggunakan `webpack`. Jika anda ingin menggunakan template lain, juga terdapat template seperti `browserify`

Mengenal Single File Component

Pada scaffold atau kerangka yang dihasilkan dari vue-cli kita dapat menemukan file berekstensi `*.vue` pada folder `src/` dan juga `src/components/`

Penggunaan file dengan ekstensi *.vue merupakan implementasi dari Single File Component pada Vue.js. dimana kita menggabungkan kode HTML, CSS, dan JavaScript dari sebuah component ke dalam satu file. Jika anda ingin mengetahui bagaimana sistem tersebut bekerja, anda dapat membaca tentang vue-loader.

Pada setiap file `*.vue` terdapat tiga bagian utama, yang ditandai dengan tag `<template></template>` , `<script></script>` , `<style></style>` .

Penjelasan tentang tiga bagian tersebut sebagai berikut :

- ***template*** : merupakan tempat untuk menempatkan script markup HTML dari sebuah komponen. Di bagian ini, selain menggunakan HTML biasa kita juga dapat menggunakan templating engine seperti pug/jade, handlebars, dsb. cukup dengan menambahkan attribute `lang="nama-templating-engine"` yang kita gunakan. Misal `<template lang="pug"></template>` .
- ***script*** : merupakan tempat untuk menuliskan script javascript dari sebuah component. Isi dari script JavaScript itu sendiri biasanya mengandung data/state, methods, dan berbagai logic yang dimiliki oleh sebuah component.
- ***style*** : merupakan tempat untuk menempatkan script css untuk styling component. Selain menggunakan CSS biasa, kita juga dapat menggunakan CSS pre-processor seperti SASS, LESS, dan Stylus untuk styling dengan menambahkan attribute `lang = "nama-css-preprocessor"` . Misal `<style lang="stylus"></style>` .

Berikut contoh single file component vue.js yang paling sederhana :

```
<template>
  <h1>{{ greeting }}</h1>
</template>

<script>
export default {
  data () {
    return {
     greeting: 'Hello World!'
   }
}
</script>

<style>
h1 {
  font-weight: lighter;
  text-align: center;
}
</style>
```

Untuk mengetahui lebih lengkap tentang bagaiman cara single file vue component bekerja, anda dapat membaca referensi tentang webpack loader vue-loader yang memungkinkan sebuah single file vue component bekerja.

Daftar Produk

Setelah menyiapkan berbagai tools yang kita perlukan, serta menyiapkan sebuah kerangka untuk project Vue.js baru menggunakan vue-cli, kita dapat mulai membuat web application kita menggunakan Vue.js. Kita akan membuat sebuah tampilan sederhana untuk menampilkan daftar data produk.

Membuat Component Daftar Produk

Anda dapat memulai dengan membuat sebuah file component baru bernama `ListProduct.vue` pada folder `src` yang berada pada kerangka project baru yang sebelumnya sudah anda siapkan menggunakan `vue-cli` .

isi dari file `ListProduct.vue` kurang lebih seperti berikut :

```
<!-- src/components/ProductList.vue -->
<template>
  <table class="table table-hover product-table">
    <thead>
      <tr>
        <th>Name</th>
        <th>Description</th>
        <th>Price</th>
      </tr>
    </thead>
    <tbody>
      <tr v-for="product in products" track-by="id">
        <td>{{product.name}}</td>
        <td>{{product.description}}</td>
        <td>{{product.price}}:-</td>
      </tr>
    </tbody>
  </table>
</template>

<script>
export default {
  data () {
    return {
      products: [
        {
          id: 'cc919e21-ae5b-5e1f-d023-c40ee669520c',
          name: 'COBOL 101 vintage',
          description: 'Learn COBOL with this vintage programming book',
          price: 399,
        },
        {
          id: 'bcd755a6-9a19-94e1-0a5d-426c0303454f',
          name: 'Sharp C2719 curved TV',
          description: 'Watch TV like never before with the brand new curved ' +
            'screen technology',
          price: 1995,
        },
        {
          id: '727026b7-7f2f-c5a0-ace9-cc227e686b8e',
          name: 'Remmington X mechanical keyboard',
          description: 'Excellent for gaming and typing, this Remmington X ' +
            'keyboard features tactile, clicky switches for speed and accuracy',
          price: 595,
        }
      ]
    }
  }
}
</script>
```

COMING SOON

COMING SOON

COMING SOON

COMING SOON

COMING SOON

COMING SOON