Book Series

of

jago{coding}

Learn & Share {Your Code}

www.jagocoding.com

Tutorial DOM Templating Menggunakan Mustache.js

Oleh: Cecep Yusuf

Pengenalan dan cara penggunaan Mustache.js dalam pengembangan halaman HTML berbasis DOM Javascript supaya reusable dan maintenable.

Selamat malam temen-temen, sudah lama sekali saya tidak membuat tutorial di situs tutorial terbaik ini karena saya akhir-akhir ini sibuk sekali, yaa selain sibuk pekerjaan di luar pekerjaan kantor, ada juga kesibukan lainnya. Tetapi di dalam kesibukan tersebut saya berusaha untuk selalu menggali dan menggali sesuatu yang baru, dan seketika saya mencari yang baru, saya malah teringat ada sesuatu yang belum saya share pada sahabat **jacoders**, yaitu **Mustache.js**. Sebenarnya library Javascript ini sudah lama keberadaannya menemani para **Javascripters** yang sering memanipulasi DOM di dalam programnya. Karena keberadaannya juga sudah cukup lama, mungkin saja di sini banyak temen-temen jacoders yang sudah tahu untuk apa fungsi library Javascript ini. Adapun buat temen-temen yang belum tahu, kini saya akan memberi tahu kalian bahwa saya menge-*share* sebuah library yang betul-betul sangat dibutuhkan oleh para **Javascripters** yang sehari-harinya bergelut dengan DOM, *so* di sini saya akan *share* apa itu **Mustache.js** dan cara menggunakannya.

Mengapa Memakai Mustache.js?

Biasanya, ketika kita mengembangkan sebuah aplikasi yang memaksimalkan manipulasi DOM untuk membuat halaman HTML yang dinamis, kita selalu membutuhkan kode-kode Javascript untuk merender HTML menggunakan kode-kode tersebut, baik dengan menggunakan value string manual maupun dengan library jQuery secara *programmaticaly,* yaitu dengan menggunakan perintah syntax-syntax Javascript. Kita bisa menggabungkan string HTML di dalam variable Javascript, kemudian memasukkannya ke sebuah element menggunakan function html() atau append() bawaan Javascript, begitu kan? Lihat contoh di bawah ini:

```
$.each(pesan.getAll(), function(i, pesan) {
  $('#listPesan').append(
    '<li><span class="nama">' +
    pesan.nama + '</span>' +
    '<abbr class="tanggal" title="' +
    pesan.tanggal + '"></abbr>' +
    '<p class="content">' + pesan.isi + '</p></li>');
```

```
});
```

Contoh di atas adalah contoh di mana kita membuat sebuah elemen **ul li** di dalam div yang ID nya adalah **listPesan**. Di dalam ul li tersebut berisi data-data dari hasil pengulangan **pesan.getAll()**. Jika kita melihat kode di atas, elemen yang dibuat oleh javascript, yaitu elemen ul li itu dibuat secara manual menggunakan append string. Teknik kode di atas tersebut tidak salah, tetapi akan sangat susah digunakan kembali pada modul lain (*reusability*) dan susah untuk di-*maintenance*. Jadi, bagaimana cara yang tepat? Caranya adalah dengan menggunakan templating. Pada tutorial kali ini, kita akan membahas bagaimana cara membuat template untuk DOM menggunakan library **Mustache.js**.

Ternyata ada banyak library yang menangani HTML DOM templating ini, jika saya sebutkan yaitu ada jQuery Templates, Underscore.js, dan Mustache.js. Mustache.js adalah yang paling sering dipilih dan digunakan banyak developer karena mempunyai *powerful syntax* dan cepat dalam rendering/parsing templatenya.

Mustache.js dapat disebut sebagai **"Logic-less template syntax"**, yang berarti tidak benar-benar menggunakan syntax-syntax logika seperti if, else, for, dan sebagainya, melainkan menggunakan tag-tag sebagai pendefinisian kondisi dan perulangannya. Mustache.js pun tidak hanya bekerja untuk Javascript saja, melainkan untuk bahasa pemrograman populer lain. Ada banyak library Mustache untuk beberapa bahasa pemrograman, antara lain Ruby, JavaScript, Python, PHP, Perl, Objective-C, Java, .NET, Android, C++, Go, Lua, Scala, dan sebagainya. Di dalam tutorial ini, kita akan belajar secara cepat mengenai bagaimana cara menggunakan **Mustache.js** untuk Javascript.

Untuk lebih detail tentang Mustache.js, buka saja: http://en.wikipedia.org/wiki/Mustache_(template_system)

Situs resmi: http://mustache.github.io/

Pertama-tama, load javascript menggunakan tag <script> seperti berikut:

```
<script src="js/mustache.js" type="text/javascript"></script>
```

File Mustache.js tersedia pada link berikut: https://github.com/janl/mustache.js/blob/master/mustache.js

Ok kita langsung saja ke pembahasan untuk penggunakan Mustache.js. Caranya yaitu dengan mengikuti dan mempelajari contoh-contoh di bawah ini.

Contoh 1: Templating Dasar

Contoh di bawah ini adalah jika variable **me** mempunyai data array javascript secara internal, alias bukan melalui AJAX/Http Request. Kemudian setelah ada data di dalam variable **me**, maka selanjutnya dimasukkan ke dalam variable **template** dengan isi yaitu HTML template yang akan direnderer oleh Mustache. Selanjutnya html hasil render Mustache akan dimasukkan ke dalam div **#result** dengan menggunakan function html(). Lihatlah kode berikut ini:

```
var me = {
  namaLengkap: "Cecep Yusuf",
```

```
    email: "cheyuz@gmail.com",
    url: "http://cheyuz.com"
  };
  var template = "<p>{{namaLengkap}}</p>E-Mail: {{email}}, Situs:
{{url}}";
  var html = Mustache.to_html(template, me);
  $('#result').html(html);
```

Maka hasilnya adalah:

Cecep Yusuf

E-Mail: cheyuz@gmail.com, Situs: http://cheyuz.com

function **Mustache.to_html** adalah digunakan untuk merender isi dari variable **template** menjadi sebuah HTML yang sudah diparsing lengkap dengan data-data dari variable array **me**. Kemudian dimasukkan ke dalam element div dengan ID **#result**.

Contoh 2: Templating Dasar dengan Menggunakan AJAX/HTTPRequest

Untuk contoh di bawah ini, kita menggunakan metode AJAX untuk mengambil data dari server/web service. Jangan lupa bahwa data yang diretreive harus berformat JSON. Dan setelah itu masukkan datanya sama dengan contoh 1, yaitu ke dalam div #result. Adapun bentuk kodenya adalah seperti berikut ini:

```
  $.getJSON('json/data.json', function(data) {
    var template = "<p>{{namaLengkap}}</p>E-Mail: {{email}}, Situs:
{{url}}";
    var html = Mustache.to_html(template, data);
    $('#result').html(html);
  });
```

Hasilnya akan sama seperti di atas:

Cecep Yusuf

E-Mail: cheyuz@gmail.com, Situs: http://cheyuz.com

Contoh 3: Menggunakan File Template Eksternal

Mustache.js juga dapat digunakan untuk file template yang kita buat terpisah dengan file html utama, dengan data yang diambil dengan menggunakan AJAX untuk meretreive hasil JSON. Lihatlah

kode di bawah ini:

```
  $.getJSON('json/data2.json', function(data) {
    var template = $('#meTpl').html();
    var html = Mustache.to_html(template, data);
    $('#result').html(html);
  });
```

Dan ada satu file terpisah, misal nama filenya adalah **me.html** yang isinya seperti berikut:

```
  <p>{{namaLengkap}}</p>
  <p>E-Mail: {{email}}</p>
  <p>Situs: {{url}}</p>
```

Dan di HTML utama, sebelum kode untuk load JSON dibuat, tambahkan script berikut ini:

```
<script id="meTpl" type="text/template" src="me.html"></script>
```

Maka hasilnya adalah:

Cecep Yusuf

E-Mail: cheyuz@gmail.com

Situs: http://cheyuz.com

Contoh 4: Perulangan Data dengan Templating

Mustache.js itu keren, karena dapat membuat iterasi perulangan data di dalam template itu sendiri. Coba lihat kode berikut:

```
  var data = {
    nama: "Cecep Yusuf",
    hobbyku: [
      'Coding',
      'Menulis',
      'Bernyanyi'
    ]};
  var tpl = "Hobby {{nama}}:
<ul>{{#hobbyku}}<li>{{.}}</li>{{/hobbyku}}</ul>";
  var html = Mustache.to_html(tpl, data);
  $('#result').html(html);
```

Untuk membuat iterasi perulangan, kita cukup memanggil element array yang mempunyai child

dengan menambahkan tanda pagar (#), sehingga jika kita buat **{{#hobbyku}}** maka setiap data yang ada di dalam **hobbyku** akan ditampilkan. Dan di situ ada syntax **{{.}}**, tanda titik ini berarti mengambil data dari setiap array yang ada di dalam **hobbyku** yang tidak mempunyai *key*.

Maka, hasilnya adalah:

Hobby Cecep Yusuf:

- Coding
- Menulis
- Bernyanyi

Contoh 5: Perulangan Data dengan Object

Mirip dengan contoh 4, hanya saja yang dilooping datanya di sini berupa object, mempunyai key dan value.

```
  var data = {
    mahasiswa:
      [{
        nama: "Cecep Yusuf",
        email: "cheyuz@gmail.com"
      },{
        nama: "Irvan Riswanto",
        lastName: "iriswanto@gmail.com"
      }]
  };
  var template = "Mahasiswa:<ul>{{#mahasiswa}}" +
    "<li>{{nama}}, email: {{email}}</li>" +
    "{{/mahasiswa}}</ul>";
  var html = Mustache.to_html(template, data);
  $('#result').html(html);
```

Hasilnya adalah sebagai berikut:

Mahasiswa:

- Cecep Yusuf, email: cheyuz@gmail.com
- Irvan Riswanto, email: iriswanto@gmail.com

Contoh 6: Nested Objects

Dengan Mustache.js, kita juga dapat parsing data yang berupa *nested object*, seperti misalnya

NamaObject.NamaKey. Agar lebih jelas silakan lihat contoh kode berikut:

```
var user = {
  nama: "Cecep Yusuf",
  email: "cheyuz@gmail.com",
  url: "http://cheyuz.com",
  account : {
    username: "cheyuz",
    group: "administrator"
  }
};
var template = "<p>{{nama}}, {{email}}</p><p>Situs: {{url}}</p>" +
  "Account: {{account.username}} sebagai {{account.group}}";
var html = Mustache.to_html(template, user);
$('#result').html(html);
```

Maka, hasilnya adalah:

Cecep Yusuf, cheyuz@gmail.com

Situs: http://cheyuz.com

Account: cheyuz sebagai administrator

Contoh 7: Dereferencing

Sebenarnya untuk contoh ini sama dengan contoh 6, tetapi bedanya di sini kita membuat bentuk yang berbeda, yaitu menghilangkan object reference beserta tanda titiknya, tetapi sudah didefinisikan dengan memanggil nama variable induk dengan ditambah tanda pagar (#).

```
var user = {
  nama: "Cecep Yusuf",
  email: "cheyuz@gmail.com",
  url: "http://cheyuz.com",
  account : {
    username: "cheyuz",
    group: "administrator"
  }
};
var template = "<h1>{{nama}}, {{email}}</h1><p>Situs: {{url}}</p>" +
  "{{#account}}Account: {{username}} sebagai {{group}}{{/account}}";
var html = Mustache.to_html(template, user);
$('#result').html(html);
```

Hasilnya sama, yaitu:

Cecep Yusuf, cheyuz@gmail.com

Situs: http://cheyuz.com

Account: cheyuz sebagai administrator

Contoh 8: Function

Jangan salah, Mustache.js juga punya kemampuan untuk memparsing function yang sudah kita definisikan, seperti contoh berikut:

```
var produk = {
  nama: "Canon 600D",
  harga: 6000000,
  jumlah: 2,
  total: function() {
    return this.harga * this.jumlah;
  }
};
var template = "<p>Nama Produk: {{nama}}</p>Harga Total Rp{{total}}";
var html = Mustache.to_html(template, product);
$('#result').html(html);
```

Di situ **total** merupakan sebuah function javascript. Hasilnya akan seperti ini:

Nama Produk: Canon 600D

Harga Total Rp12000000

Contoh 9: Partial

Partial dapat diartikan bahwa template dibuat menjadi bagian-bagian terpisah yang dapat digabungkan. Contohnya adalah seperti ini:

```
var data = {
  nama: "Cecep Yusuf",
  namaPanggilan: "Cheyuz",
  jalan: "Jl.Menteng No.37",
  kota: "DKI Jakarta",
  kodePos: "999999"
};
```

```
  var template = "<p>{{nama}} '{{namaPanggilan}}'</p>{{>alamat}}";
  var partials = {alamat: "<p>{{jalan}}, {{kota}} {{kodePos}}"};
  var html = Mustache.to_html(template, data, partials);
  $('#result').html(html);
```

Di atas terlihat bahwa ada variable baru namanya **partials** yang berisi template string untuk menampilkan jalan, kota dan kode pos. Variable tersebut dapat digabungkan dengan template pada variable **template** dengan menambahkan syntaxt **{{>alamat}}**. Maka hasilnya adalah sebagai berikut:

Cecep Yusuf 'Cheyuz'

Jl.Menteng No.37, DKI Jakarta 999999

Contoh 10: Partial di Dalam Looping

Mustache dapat juga menggunakan *data looping* di dalam *partial template*.

```
  var data = { jabatan: [
      {   nama: "System Analyst",
        karyawan: [
          {namaDepan: "Cecep", namaBelakang: "Yusuf"},
          {namaDepan: "Irvan", namaBelakang: "Riswanto"}]
      },
      {   nama: "Programmer",
        karyawan: [
          {namaDepan: "Titan", namaBelakang: "Firman"},
          {namaDepan: "Winprins", namaBelakang: "Tambunan"}]
      }]
  };

  var tpl = "{{#jabatan}}<p>{{nama}}</p>" +
    "<ul>{{#karyawan}}{{>detail_karyawan}}{{/karyawan}}</ul>{{/jabatan}}";
  var partials = {detail_karyawan: "<li>{{namaDepan}}
{{namaBelakang}}</li>"};
  var html = Mustache.to_html(tpl, data, partials);
  $('#result').html(html);
```

Adapun hasilnya adalah sebagai berikut:

System Analyst:

- Cecep Yusuf
- Irvan Riswanto

Programmer

- Titan Firman
- Winprins Tambunan

Keuntungannya sih untuk saya sendiri jika kita memakai templating menggunakan Mustache.js ini adalah, kita dapat membuat file-file template terpisah yang dapat digunakan kembali (reusable) dan mudah dalam maintenance, karena kita tidak *diribetkan* dengan kode-kode Javascript DOM, sehingga kita tidak membuat DOM secara *hardcore* di dalam kode Javascript.

Ok jika ada pertanyaan langsung aja komentar di bawah ok :)

~ Cheyuz

Tentang Penulis

Cecep Yusuf

Hi, my name is Cecep Yusuf. However, in the virtual world I am more likely to use the name Cheyuz, which is an abbreviation of two words "Cecep" and "Yusuf". I am founder of Jagocoding.com, u can view more of me in Cheyuz.id