Mengembangkan CSS Framework Sendiri

EBOOK BAHASA INDONESIA

Irfan Maulana

SOFTWARE DEVELOPMENT ENGINEER | BLIBLI.COM

1. Daftar Isi

2. Tentang CSS Framework

Sebelum membahas apa itu CSS Framework perlu sedikit saya jelaskan mengenai CSS itu sendiri.
CSS (Cascading Style Sheet) sederhananya adalah sebuah script yang digunakan untuk memberikan *style* pada html page.
Bila pada HTML sebelumnya kita biasa menggunakan attribute langsung untuk melakukan *styling* seperti :
`<table valign="center" border="1"></table>`, maka pada CSS kita bisa melakukan hal yang sama dengan cara lebih cantik, seperti :
`<table style="vertical-align: middle;border: 1px solid grey;"></table>`.

Framework sendiri secara umum adalah kerangka kerja yang membantu dalam mengerjakan sesuatu hal, dalam dunia technology sendiri kata ini relevan terhadap kumpulan hal/fungsi yang telah diwadahkan menjadi satu sehingga memudahkan pengguna bila membutuhkan hal-hal yang umum dan sering digunakan tanpa harus membuat sendiri.

CSS Framework sendiri adalah kumpulan class / fungsi yang telah disediakan dalam satu wadah yang jelas akan memudahkan penggunanya dalam mengembangkan suatu website tanpa harus membuat segalanya dari awal.

3. Mengapa CSS Framework

Ada banyak faktor yang membuat seseorang memilih menggunakan CSS Framework dibandngkan membuat secara *native* atau manual sedari awal. Tapi beberapa alasan mungkin akan sama, jadi saya coba sebutkan beberapa hal umum yang menjadi alasan pengguna CSS Framework, sebagai berikut :

1. Cepat dan Praktis
   Cepat dan praktis adalah alasan yang utama, bayangkan ketika kalian harus membuat sendiri satu website dengan berbagai komponen didalamnya, ada menu navigasi, ada button, ada pesan error dan lain-lain. Sebagian besar css framework pada dasarnya akan mengakomodir kebutuhan tersebut, karena kebutuhan-kebutuhan komponen seperti diatas adalah umum bagi sebuah website.

2. Lengkap
   Seperti di poin sebelumnya, CSS Framework menyediakan banyak hal, banyak komponen, dan fungsi bahkan sampai kebutuhan yang sebenarnya tidak kita gunankan.

3. Seragam
   Menggunakan CSS Framework berarti kita mengikuti standar yang telah ditetapkan dalam framework tersebut, yang artinya akan membawa keseragaman dalam suatu website. Ini penting, karena konsistensi tampilan akan berpengaruh juga terhadap kesan pengguna terhadap si website tersebut.

4. Mudah
   Fakta bahwa CSS Framework telah memiliki standard yang ditetapkan bisa jadi bagus bisa jadi tidak, tapi pada dasarnya bagus Karena memudahkan pengguna untuk mempelajarinya apalagi didukung dengan dokumentasi yang menjelaskan fungsi-fungsi yang ada di dalamnya.

4. Mengenal CSS Framework Yang Ada

Ada banyak sekali CSS Framework diluar sana, namun saya coba kenalkan beberapa framework popular yang biasa digunakan oleh web developer, sebagai berikut :

1. Bootstrap
   Bootstrap adalah CSS framework yang dikembangkan oleh twitter yang merupakan salah satu yang paling popular dan banyak sekali digunakan sampai saat ini.
   Bootstrap menyediakan hampir semua kebutuhan kita dalam mengembangkan website, mulai dari grid sistem, komponen yang beragam, sampai dengan *library* umum yang biasanya akan kita butuhkan.
   Bagi yang sudah biasa menggunakan bootstrap pasti familiar dengan class seperti .button-primary, .button-success, dll.
   Bootstrap juga membolehkan pengguna atau komunitas untuk mengembangkan tema sendiri untuk bootstrap mereka, sehingga di luar sana banyak sekali pilihan tema untuk framework ini.

2. Foundation
   Foundation salah satu framework professional yang dikembangkan oleh Zurb Foundation, sama seperti bootstrap yang menyediakan berbagai kebutuhan foundation juga memiliki hampir sebagian besar dari fitur tersebut.
   Namun yang membedakan foundation hadir dengan lebih proesional, baik dari segi class naming, grid sistem maupun tema yang diusung olehnya.

3. Materialize
   Materialize adalah framework yang dikembangkan berdasarkan material design guideline dari Google, jadi bila ingin menggembangkan website dengan kosep material design materialize ini salah satu pilihan terdepan.

5. Mengembangkan CSS Framework Sendiri

Bila sudah banyak framework diluar sana kenapa harus repot-repot membuat framework sendiri ? kalau buat saya sendiri adalah sebagai bahan belajar dan memahami bagaimana framework itu bekerja dalam level *native.* Kebetulan di kantor sekarang (Blibli.com) tidak menggunakan framework secara utuh dan lebih memilih menggunakan beberapa bagian yang dibutuhkan sedangkan sisanya dibangun dari awal.
Sedikit bercerita bahwa dalam beberapa kali interview kandidat front end developer yang terjadi adalah *framework-minded* yang artinya ketergantungan dengan satu framework bahkan tanpa tahu apa yang dilakukan oleh framework tersebut dilevel *native*. Ini bencana kalau buat saya sendiri karena pada dasarnya framework-pun dibangun dari level *native*, jadi paling tidak kita harus berangkat dari sana jikalau ingin mempelajari sesuatu.

Berangkat dari sini, kita di ebook ini akan coba ditunjukan bagaimana membangun CSS framework sendiri, mungkin tidak sampai se-kompleks framework yang sudah ada namun semoga bisa memberi gambaran bagaimana esensinya sebuah CSS framework bekerja dibelakangnya sehingga kalaupun kita nanti akan menggunakan sebuah framework tetap akan paham dengan apa yang kita kerjakan.

5.1 Pemilihan Technology

Saya akan coba jelaskan beberapa tools yang akan saya gunakan untuk membangun CSS framework kita nantinya, akan terlihat agak subjektif karena saya memilih berdasarkan kemampan saya menggunakan tools tersebut tidak semata-mata karena tools tersebut adalah terbaik di kelasnya, namun semoga masih relevan.

Dan tools yang perlu disiapkan sebagai berikut :

1. NodeJS Engine, akan digunakan untuk instalasi beberapa tools yang kita butuhkan dibawahnya.
2. SASS CSS Preprocessor, agar css kita bisa lebih tertata rapi, modular dan mudah dalam development. Kita akan menggunakan node-sass jadi tidak perlu install Ruby sebagai compiler nantinya.
3. Grunt, sebagai task runner yang akan menjalankan beberapa tugas seperti compile sass, minify css dan js untuk production mode.
4. Sublime Text Editor
5. Browser

5.2 Menyiapkan Environment

Paling pertama adalah instalasi NodeJs, ini bisa langsung dilihat di official website nodejs.
Setelah berhasil diinstall anda bisa mengecek versi nodejs dan npm yang akan otomatis terinstall juga dengan command `node -v` dan `npm -v`.

Setelah terinstall node dan npm selanjutnya kita akan menginstall grunt dan grunt-cli secara global, dengan command `npm install -g grunt` dan `npm install -g grunt-cli`.

Setelah itu kita akan mulai menyiapkan beberapa file yang akan kita butuhkan :

1. Package.json, berisi daftar dependency kita, seperti berikut :

```
"devDependencies": {
  "autoprefixer": "^6.5.1",
  "grunt": "^1.0.1",
  "grunt-banner": "^0.6.0",
  "grunt-concurrent": "^2.2.1",
  "grunt-contrib-clean": "^1.0.0",
  "grunt-contrib-concat": "^1.0.0",
  "grunt-contrib-csslint": "^2.0.0",
  "grunt-contrib-cssmin": "^1.0.1",
  "grunt-contrib-uglify": "^2.0.0",
  "grunt-contrib-watch": "^1.0.0",
  "grunt-postcss": "^0.8.0",
  "grunt-sass": "^1.2.1"
}
```

2. Menginstall semua dependency dengan command `npm install` di folder project

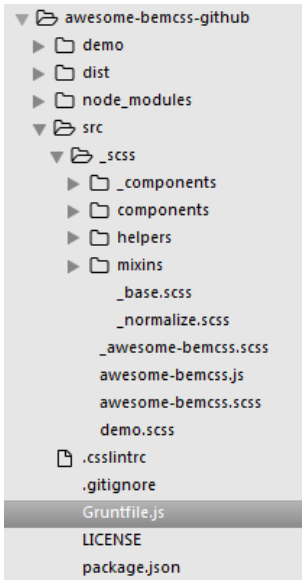
3. Gruntfile.js, merupakan task yang akan dijalankan oleh grunt nantinya, untuk initialisasi kita hanya akan membuat dua task yakni untuk compile sass file dan auto watch terhadap perubahan sass file tersebut sehingga tidak perlu compile berulang-ulang ketika sedang development, contoh gruntfile kita adalah sebagai berikut :

```
module.exports = function (grunt) {
    grunt.initConfig({
        pkg: grunt.file.readJSON('package.json'),
            sass: {
                dist: {
                    options:{
                        sourcemap: 'auto',
                        style: 'compact',
                        update: true
                    },
                    files: [{
                        expand: true,
                        cwd:  'src',
                        src:  ['**/*.scss'],
                        dest: 'dist',
                        ext:  '.css'
                    }]

                }
            },
            watch: {
                css: {
                    files: '**/*.scss',
                    tasks: ['sass']
                }
            }
    });

    grunt.loadNpmTasks('grunt-sass');
    grunt.loadNpmTasks('grunt-contrib-watch');
    grunt.registerTask('default', ['sass']);
};
```

4. Struktur folder project ini kurang lebih akan saya buat seperti ini :



5.3 Memulai Pengembangan

Setelah menyiapkan semua hal diatas, kita bisa langsung memulai untuk membuat framework kita sendiri, silahkan disimak langkah-langkahnya sebagai berikut :

1. Membuat scss utama sebagai orkestrator, saya membuat file awesome-bemcss.scss, file ini nanti berisi import dari semua yang kita butuhkan. Untuk mengecek grunt setup kita sudah bisa berjalan atau belum, kita bisa membuat satu class sebagai test di file ini contoh : .test { font-size:14px; } kemudian jalankan command grunt melalui cmd, tentunya setelah pindah ke folder project tersebut. Kemudian cek folder **dist** akan terbuat file awesome-bemcss.css yang merupakan hasil compile dari file sass kita tadi.
2. Saya akan menambahkan _normalize.scss sebagai reset css untuk menyamakan pengalaman di berbagai browser berbeda, saya tambahkan sebagai component yang artinya akan saya tambahkan _ (garis bawah) di depan nama file tersebut, file ini saya letakkan di dalam folder _scss, kemudian saya tambahkan import di file awesome-bemcss.scss.
   ```
   @import "_scss/normalize";
   ```
3. Kita bisa menjalankan command grunt watch agar tidak perlu compile ulang pada saat development.
4. Selanjutnya kita akan define warna-warna yang akan kita gunakan sebagai variable di sass, saya buat file _colors.scss dibawah folder helpers. Kita bisa juga menggunakan fungsi darken dan lighten untuk generate warna turunan, seperti potongan code berikut :

```
$white      : #ffffff !default;
$black      : #333333 !default;
$grey       : #B6B6B6 !default;
$red        : #ed1c24 !default;
$green      : #00b25a !default;
$blue       : #068AC9 !default;
$yellow     : #ff9700 !default;
$orange     : #FF5400 !default;
$brown      : #6A453C !default;

$blackDark   : darken($black,10%) !default;
$blackDarker : darken($black,20%) !default;
$blackLight  : lighten($black,10%) !default;
$blackLighter: lighten($black,20%) !default;
```

5. Setelah itu kita import file ini :
   ```
   @import "_scss/helpers/colors";
   ```
6. Kita juga bisa menyiapkan beberapa mixins variable yang nanti bisa kita gunakan untuk memudahkan development. Saya membuat beberapa mixin variable dalam file yang dipecah-pecah seperti _background,scss, _border-radius.scss, _boxes.scss, _clearfix.scss, _image.scss, _placeholders.scss, _transitions.scss.
   Kemudian file-file ini akan kita import di file utama kita awesome-bemcss.scss.
   ```
   @import "_scss/mixins/background";
   @import "_scss/mixins/border-radius";
   @import "_scss/mixins/boxes";
   @import "_scss/mixins/clearfix";
   @import "_scss/mixins/image";
   @import "_scss/mixins/placeholders";
   @import "_scss/mixins/transitions";
   ```
7. Setelah itu kita akan membuat base setting untuk framework kita, ini dimaksudkan agar ketika menggunakan element standar html akan mendapatkan style standard yang telah kita set. Namun saya tidak terlalu banyak men-set style untuk element standard html dikarenakan tidak mau terlalu banyak terjadi tumpeng tindih kalau pengguna justru ingin custom pada style nya, contoh _base.scss :

```scss
html {
    @include box-sizing(border-box);
    font-size: 62.5%;
    color: $black;
    -ms-overflow-style: scrollbar;
    -webkit-tap-highlight-color: transparent;
}
*, *:before, *:after {
    box-sizing: inherit;
}
body {
    font-size: 1.3rem;
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, sans-serif;
}

a {
  color: $blue;
  text-decoration: none;

  &:hover, &:focus{
      text-decoration: underline;
  }
}

code, kbd, pre, samp {
    background-color: $greyLighter;
    padding: .1em .5em;
    border-radius: .1em;
}

blockquote{
  padding: 10px;
  margin: 0 0 20px;
  font-size: 1.6rem;
  border-left: 5px solid #eee;
}

textarea {
    resize: none;
}
```

8. Setelahnya kita bisa import file tersebut dibawah `_normalize.scss`.
```scss
@import "_scss/normalize";
@import "_scss/base";
```
9. Setelah semua langkah diatas, saatnya kita bisa mulai mengembangkan komponen yang akan kita tetapkan di framework kita ini.
kita coba dengan hal-hal yang umum dan sederhana saja.
kita split masing-masing komponen ke masing-masing file scss agar lebih modular.
10. Kita mulai buat file komponen `_textfield.scss`, kita bisa menggunakan beberapa feature dari sass seperti `@include` yang berarti menggunakan variable mixins yang kita define sebelumnya, atau menggunakan `&` dengan maksud menghasilkan class dengan prefix seperti parent-nya.
contoh file `_textfield.scss` sebagai berikut :

```scss
.textfield
{
    padding: 6px 12px;
    margin: 3px 0;
    width: 100%;
    outline: none;
    border: 1px solid $greyLight;
    border-radius: 0;
    @include placeholder-color($white);
}
```

11. Screenshoot diatas hanya rule sederhana untuk sebuah textfield, dimana bisa digunakan untuk sebuah input text, text area, input select dll.
    Kita juga akan menambahkan beberapa element dan modifier tambahan guna mengakomodir kebutuhan umumnya, seperti textfield dengn error state, textfield dengan border radius, juga error message label.
    Contoh pembuatan element dan modifier, sebagai berikut :

```scss
.textfield
{
    padding: 6px 12px;
    margin: 3px 0;
    width: 100%;
    outline: none;
    border: 1px solid $greyLight;
    border-radius: 0;
    @include placeholder-color($white);

    &__helper {
        color: $greyDark;
        font-size: 1.2rem;
        line-height: 2;
    }

    &__error {
        color: $red;
        font-size: 1.2rem;
        line-height: 2;
    }

    &--radius
    {
        @include border-radius(0.25em);
    }

    &--error
    {
        border: 1px solid $red;
    }

    &--large
    {
        padding: 20px;
    }
}
```

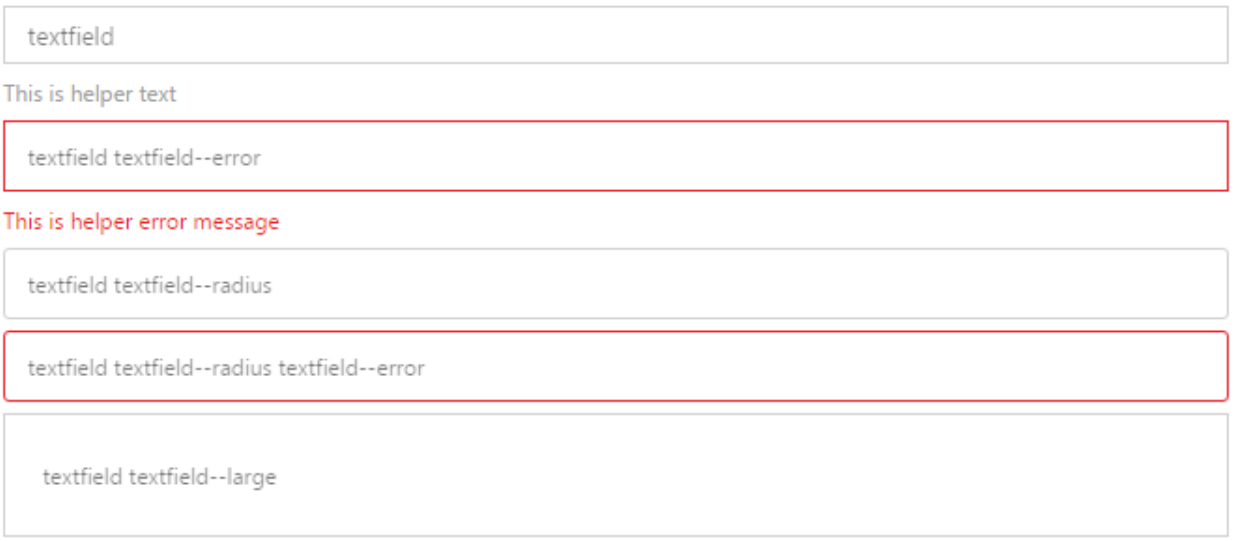
12. Setelah itu kita import pada file utama kita awesome-bemcss.scss agar bisa digunakan :

```
@import "_scss/components/textfield";
```

13. Kita bisa menggunakan class yang telah kita buat di html kita, seperti :

```
<input class="textfield" id="textfield-1" placeholder="textfield" />
<span class="textfield__helper">This is helper text<span>
<input class="textfield textfield--error" id="textfield-2" placeholder="textfield textfield--error"/>
<span class="textfield__error">This is helper error message<span>
<input class="textfield textfield--radius" id="textfield-3" placeholder="textfield textfield--radius"/>
<input class="textfield textfield--radius textfield--error" id="textfield-4" placeholder="textfield textfield--radius
textfield--error"/>
<input class="textfield textfield--large" id="textfield-4" placeholder="textfield textfield--large" />
```

14. Html tersebut akan menghasilkan tampilan seperti ini :



15. Setelah berhasil membuat komponen yang sederhana seperti textfield, sekarang kita coba buat yang agak kompleks, namun tetap sederhana karena tujuannya untuk pembelajaran, berikutnya kita akan buat grid system sederhana sendiri.
    Kita buat file _grid.scss kemudian kita define beberapa kebutuhan grid system kita seperti clearfix, row dan column dan flex sederhana, berikut contoh rule class yang sudah kita buat :

```scss
.clearfix{
    @include clearfix();
}

.row {
    @include clearfix();
    margin: 0 auto;
    max-width: 92.308em;
}

.flex{
    display: flex;
    justify-content: space-between;
    align-items: center;
}

$columnCount : 16 !default;
$break-medium: 63.750em;

@for $i from 1 through $columnCount {
    .col-#{$i}{
        float: left;
        width: ((100/$columnCount) * $i) *1%;
    }
}
@for $i from 1 through $columnCount {
    .col-offset-#{$i}{
        margin-left: ((100/$columnCount) * $i) *1%;
    }
}
```

16. Setelah itu kita import pada file utama kita awesome-bemcss.scss:

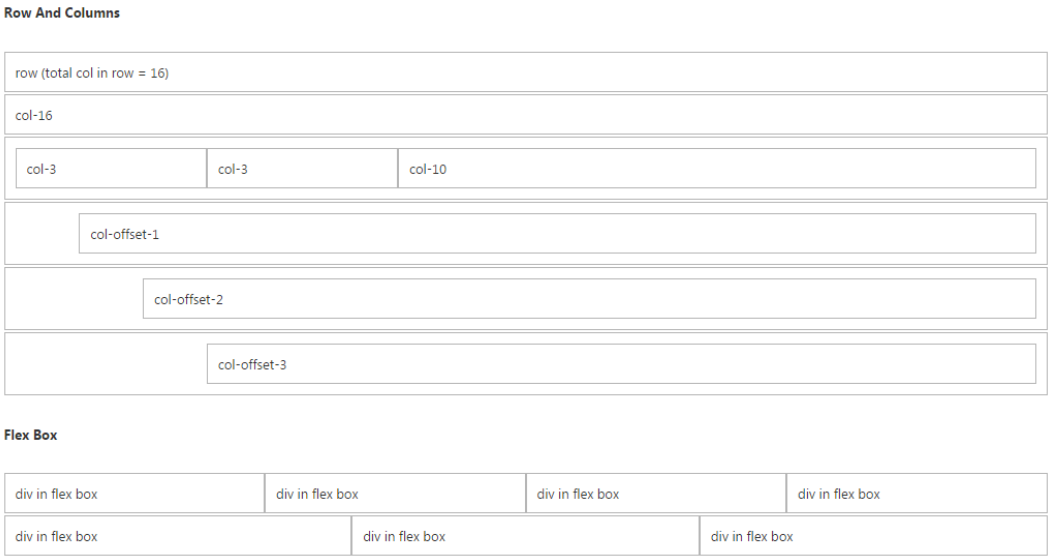
```scss
@import "_scss/components/grids";
```

17. Kita bisa menggunakan class tersebut di Html seperti berikut :

```html
<div class="row">row (total col in row = 16)</div>
<div class="row">
  <div class="col-16">col-16</div>
</div>
<div class="row">
  <div class="col-3">col-3</div>
  <div class="col-3">col-3</div>
  <div class="col-10">col-10</div>
</div>
<div class="row">
  <div class="col-offset-1">col-offset-1</div>
</div>
<div class="row">
  <div class="col-offset-2">col-offset-2</div>
</div>
<div class="row">
  <div class="col-offset-3">col-offset-3</div>
</div>
```

```
<div class="flex">
  <div class="item">div in flex box</div>
  <div class="item">div in flex box</div>
  <div class="item">div in flex box</div>
  <div class="item">div in flex box</div>
</div>
```

18. Dan akan menghasilkan



19. Setelah selesai dengan grid system, kita akan coba buat komponen button dengan berbagai variannya. Kita buat file `_button.scss` sebagai modulnya.
    Kebutuhan dasar class button adalah sebagai berikut :

```
.button
{

    display: inline-block;
    padding: 10px 20px;
    outline: none;
    border: none;
    text-align: center;
    text-decoration: none;
    cursor: pointer;
    -webkit-appearance: none;
}
```

20. Kita akan tambahkan modifier radius di dalam button, bilamana ada yang kurang senang dengan flat button

```
&--radius{
    @include border-radius(.25em);
}
```

21. Karena button ini akan kita buat beberapa varian warna, maka saya akan membuat array yang nanti akan kita loop saja untuk generate varian ini, saya tambahkan variable berikut di _colors.scss

```
$buttonVariansName: green, red, blue, orange, gray;
$buttonVariansHex: $green, $red, $blue, $orange, $grey;
$buttonVariansHover: $greenLight, $redLight, $blueLight, $orangeLight, $greyLight;
$buttonVariansFont: $white, $white, $white, $white, $black;
```

22. Setelah itu bisa kita loop di dalam _button.scss nya sebagai berikut :

```
@for $i from 1 through length($buttonVariansName) {
    &--#{nth($buttonVariansName, $i)} {
        background: nth($buttonVariansHex, $i);
        color: nth($buttonVariansFont, $i);

        &:hover
        {
            background: nth($buttonVariansHover, $i);
            color: nth($buttonVariansFont, $i);
            text-decoration: none;
        }
    }
}
```

23. Setelah selesai kita bisa import di awesome-bemcss.scss :

```
@import "_scss/components/buttons";
```

24. Dan bisa langsung digunakan di html seperti berikut :

```
<button class="button button--green">button--green</button>
<button class="button button--red">button--red</button>
<button class="button button--blue">button--blue</button>
<button class="button button--orange">button--orange</button>
<button class="button button--gray">button--gray</button>
```

25. Akan menghasilkan tampilan sebagai berikut :

5.4 Production Mode

Untuk melengkapi pekerjaan kita, maka akan kita siapkan file yang *production ready* atau siap bila digunakan di public website.
Beberapa hal yang akan kita lakukan merupakan hal yang sederhana, antara lain sebagai berikut :

1. Hapus semua isi folder **dist** sebelum menjalankan task, menggunakan plugin grunt-contrib-clean sebagai berikut :

```
clean: {
  build: {
    src: ['dist']
  }
},
```

2. Tambahkan vendor prefix untuk cross browser support menggunakan plugin autoprefix dan grunt-postcss, sebagai berikut :

```
postcss: {
    options: {
      map: {
          inline: false,
          annotation: 'dist/map/'
      },

      processors: [
        require('autoprefixer')({browsers: '> 1%, last 2 version'})
      ]
    },

    dist: {
      src: [
        'dist/**/*.css',
        '!dist/**/*min.css'
      ]
    }
},
```

3. Kita gunakan grunt-contrib-csslint untuk validasi css yang kita gunakan sesuai dengan standard atau tidak, sebagai berikut :

```
csslint: {
    options: {
        csslintrc: '.csslintrc',
        import: false
    },
    check: {
        src: [
            'dist/**/*.css'
        ]
    }
},
```

4. Compress file css menggunakan grunt-contrib-cssmin, sebagai berikut :

```
cssmin: {
    options:{
        sourcemap: true,
        shorthandCompacting: true,
        keepSpecialComments: 0,
        removeDuplicates: false,
        restructure: true,
        mergeAdjacent: true,
        mergeMediaQueries: true
    },
    target: {
        files: [{
            expand: true,
            cwd: 'dist',
            src: ['**/*.css'],
            dest: 'dist',
            ext: '.min.css'
        }]
    }
},
```

5. Compress file javascript dengan grunt-contrib-uglify, sebagai berikut :

```
uglify: {
    compress: {
        sourcemap: true,
        sequences: true,
        dead_code: true,
        conditionals: true,
        booleans: true,
        unused: true,
        if_return: true,
        join_vars: true,
        drop_console: true,
        preserveComments : 'all'
    },
    target: {
        files: [{
            expand: true,
            cwd: 'src',
            src: ['**/*.js'],
            dest: 'dist',
            ext: '.min.js'
        }]
    }
}
```

6. Tambahkan banner sebagai stamp di file hasil compress menggunakan grunt-banner, sebagai berikut :

```
usebanner: {
    taskName: {
      options: {
        position: 'top',
        banner: '/*! awesome-bemcss v<%= pkg.version %> build <%= grunt.template.today("yyyy-mm-dd hh:MM") %> */',
        linebreak: true
      },
      files: {
        src: [
            'dist/**/*.min.css',
            'dist/**/*.min.js'
        ]
      }
    }
},
```

7. Gunakan grunt-concurrent agar bisa kita atur mana yang mesti jalan sync dan mana yang async, sebagai berikut :

```
concurrent: {
    target1: ['clean'],
    target2: ['sass'],
    target3: ['postcss:dist'],
    target4: ['csslint'],
    target5: ['cssmin'],
    target6: ['uglify'],
    target7: ['usebanner']
},
```

8. Register semua task tersebut di grunt

```
grunt.loadNpmTasks('grunt-concurrent');
grunt.loadNpmTasks('grunt-contrib-clean');
grunt.loadNpmTasks('grunt-sass');
grunt.loadNpmTasks('grunt-contrib-uglify');
grunt.loadNpmTasks('grunt-contrib-csslint');
grunt.loadNpmTasks('grunt-postcss');
grunt.loadNpmTasks('grunt-contrib-cssmin');
grunt.loadNpmTasks('grunt-contrib-watch');
grunt.loadNpmTasks('grunt-banner');

grunt.registerTask('default',
        [
            'concurrent:target1',
            'concurrent:target2',
            'concurrent:target3',
            'concurrent:target4',
            'concurrent:target5',
            'concurrent:target6',
            'concurrent:target7'
        ]);
```

6. Bahan Bacaan

1. Semua file sudah tersedia di repository github saya di : https://github.com/mazipan/awesome-bemcss
2. Baca lebih lanjut mengenai bem css : http://getbem.com/
3. Baca mengenai SASS http://sass-lang.com/guide
4. Baca mengenai Gruntjs http://gruntjs.com/getting-started
5. Baca ebook saya lainnya di https://mazipan.github.io/demo

7. Tentang Penulis

Irfan Maulana, saat menulis ebook ini di akhir 2016 masih bekerja sebagai Software Development Engineer di salah satu ecommerce di Indonesia Blibli.com.
Telah bekerja dengan CSS selama lebih dari 3 tahun belakangan dan sepertinya akan semakin lama lagi.
Antusias dengan komunitas yang berkaitan dunia pemrograman di Indonesia khususnya di Jakarta.

Email : mazipanneh@gmail.com
Blog : mazipanneh.com
Facebook : /mazipanneh
Twitter : @Maz_Ipan
Linkedin : /in/irfanmaulanamazipan
Github : mazipan