

Buku Digital

Praktek Menggunakan **CSS Preprocessor**

E.R. Nurwijayadi

Halaman ini sengaja dikosongkan.

Buku Digital Bebas Unduh

Praktek Menggunakan CSS Preprocessor

Edisi 0.4: November 2020

Target edisi 0.4:

Materi sampai bab empat.

Akan dilanjutkan di edisi 0.5.

Oleh: E.R. Nurwijayadi

epsi-rns.gitlab.io

Praktek Menggunakan CSS Preprocessor

Copyright © 2020 E.R. Nurwijayadi

Buku ini dilisensikan sebagai CC BY-NC-SA, sehingga selain bebas dibaca buku ini juga bebas digandakan dan diubah untuk keperluan apa saja selain komersial, dengan syarat menyebutkan secara lengkap, nama penulis dan pernyataan lisensi ini, menggunakan lisensi yang sama.

Mengenai E.R. Nurwijayadi

Di media sosial, penulis biasa dipanggil epsi.

Penulis tidak pernah memiliki pendidikan formal di bidang IT.

Buku ini adalah bukti bahwa, pengetahuan penulis masih sangat terbatas.

Blog:

epsi-rns.gitlab.io

epsi-rns.github.io

*Bahan pelajaran ini diberikan sebagai wawasan bagi pemula,
supaya mudah beranjak ke tingkat lanjut.*

*Bahan yang perlu dipahami memang banyak,
Dan harus dilewati.*

Maaaf merepotkan.

Prakata

Buku digital ini ditulis untuk pembaca yang telah memiliki dasar [HTML](#) maupun [CSS](#), namun ingin mendalami ilmu stylesheet dengan melakukan kustomisasi lebih lanjut. Pendekatan buku ini adalah langsung ke penerapan sehari-hari, supaya sohib pembaca memiliki bayangan, mengenai manfaat [CSS Preprocessor](#).

Buku ini dibagi menjadi beberapa bab.

- ▶ 1. Pengenalan HTML dan CSS
- ▶ 2. Menggunakan SASS
- ▶ 3. Menggunakan LESS
- ▶ 4. Menggunakan PostCSS
- ▶ 5. Penerapan di Bootstrap
- ▶ 6. Penerapan di Bulma
- ▶ 7. Task Runner dan Bundler
- ▶ 8. Custom CSS dengan Tailwind CSS

Bagian pertama berisi motivasi untuk belajar CSS Preprocessor, dan gambaran besar secara sistematis mengenai bagaimana pembelajaran ini harus ditempuh tahap demi tahap.

Bagian kedua dibagi antara bab kedua sampai keempat, berisi mengenai looping di dalam CSS Preprocessor. Penulis sengaja memilih looping sebagai contoh kasus, karena sudah mencakup beberapa hal mulai dari penetapan variabel, pairs dan hal lain yang terkait dasar pemrograman.

Bagian ketiga berisi penerapan CSS Preprocessor dalam kehidupan nyata. Mulai dari Bootstrap yang populer di masa lalu. Bulma yang ringan, supaya pemula tahu hal lain selain Bootstrap. Dan tentunya Tailwind CSS untuk tingkat yang lebih lanjut.

Mengapa Menulis?

Penulis merasakan sekian lama menjadi orang bodoh yang tidak tahu harus dari mana memulai belajar, sehingga begitu banyak waktu terbuang. Sampai sekarangpun penulis belum menjadi web developer profesional dengan hasil yang terukur. Namun setidaknya penulis mulai paham apa yang harus dilakukan. Langkah apa yang harus ditempuh, dan ini yang dibagi untuk sama-sama belajar.

Penulis memang memulai proyek mandiri dari membuat sistim informasi. Namun penulis dengan sengaja menurunkan derajat hanya menjadi blogger, dengan beberapa content, salah satunya adalah mengenai pembikinan SSG (*static site generator*). Mengapa repot menulis panduan mengenai cara membikin blog dengan SSG? Karena ini akan memudahkan orang untuk berbagi ilmu, khususnya yang terkait dengan teknologi infomasi.

Mengapa Berbagi?

Mengapa PDF berbahasa Indonesia? Karena penulis berjumpa dengan begitu banyak orang yang ingin belajar namun memiliki keterbatasan. Mulai dari keterbatasan membaca tulisan berbahasa Inggris. Maupun keterbatasan pulsa sehingga tidak dapat *online* setiap saat, sehingga perlu berkas PDF yang dapat diunduh untuk kemudian dipelajari.

Mengapa buku ini dibikin menjadi buku bebas unduh (*free-e-book*), dan bukan komersial (dapat dibeli di toko)? Karena hal yang disampaikan di buku ini adalah hal mendasar yang harus dipahami seorang *web developer*. Kita perlu percepatan penyebaran pengetahuan mengani hal-hal yang bersifat mendasar, sehingga penulis merasa perlu supaya buku ini harus dapat diakses dengan mudah oleh siapa saja yang membutuhkannya.

Di dalam forum diskusi kita akan sama-sama berkembang kalau kita semua memiliki takaran ilmu yang minimum untuk berdiskusi, setidaknya sudah paham HTML dan CSS. Perbedaan ilmu yang mencolok dapat menjadi ganjalan, karena dapat menghambat yang lain untuk berdiskusi tingkat lanjut. Buku ini ditulis untuk memudahkan teman-teman di forum untuk mencapai takaran ilmu minimum ini.

Setelah menguasai CSS sebagai dasar, pembaca dapat beranjak ke buku digital berikutnya. Masih banyak topik yang dapat dibahas atau bahkan dijadikan buku, baik buku bebas unduh maupun buku cetak berbayar.

Bab Pertama

Pengenalan HTML dan CSS

Panduan perjalanan bagi pemula untuk menjadi web developer.

Motivasi

Bagaimana memulai belajar web development? Jadi sohib ingin membikin sesuatu yang berarti dalam hidup. Sohib ingin memutuskan menjadi pengembang web. Bagaimana seorang pemula harus memulai?

Dari mana memulai adalah pertanyaan-pertanyaan umum yang berulang kali muncul. Ada begitu banyak tutorial di internet, namun sangat sedikit yang memiliki gambaran besar yang memberi panduan, apa saja yang mesti ditempuh pemula. Salah satu hal yang tersembunyi namun penting adalah *CSS Preprocessor* yang menjadi dasar bagi adanya *CSS Framework*, namun ini justru jarang dibahas tertimbun oleh popularitas *CSS Framework* itu sendiri.

Selain perlu paham mengenai gambaran besar, yaitu jenjang yang mesti ditempuh, maka pemula biasanya perlu juga contoh nyata dalam keseharian. Ini sebabnya buku ini memakai pendekatan praktek, alih-alih teori. Karena itu di dalam buku ini diberikan beberapa contoh kasus, mulai dari hasil coding-nya, hasil keluarannya, sampai hasil tampilannya.

Seseorang yang menguasai *CSS Preprocessor*, dapat dengan bebasnya membuat *custom CSS* bikinan sendiri. Dengan demikian akan dengan mudah membuat tema website sendiri tanpa tergantung bikinan orang lain. Dan dapat juga mengubah tema yang ada supaya sesuai dengan kebutuhan setempat.

Peta Jalan

Suatu website yaitu roadmap.sh, dengan lugas membagi peran para pengembang web sebagai berikut:

- ▶ *Frontend*
- ▶ *Backend*
- ▶ *Devops*

Bila ketiga hal tersebut digabungkan maka dikenal istilah lain yaitu *full stack web developer*. Dari sudut pandang secara umum, perusahaan yang memberikan lowongan pekerjaan *fullstack web developer*, akan tampak memiliki anggaran kecil, walaupun ini bukan patokan.

Mobile

Perlu dipahami pula bahwa pengembangan aplikasi berupa *mobile development*, berbeda dengan *web development*.

Belajar Mandiri

Di Mana Belajar?

Bila sohib adalah pemula, maka mulailah dari beberapa situs berikut:



- ▶ w3schools.com
- ▶ roadmap.sh

Yang biasa penulis lakukan adalah meminta pemula untuk membaca w3schools.com. Dan untuk yang bukan pemula saya meminta untuk menelusuri diagram-diagram yang terdapat di roadmap.sh.

Bilamana dibutuhkan masih ada beberapa lagi.



- ▶ css-tricks.com
- ▶ <https://google.github.io/styleguide/htmlcssguide.html>

Penulis sendiri menulis presentasi di blog.



- ▶ <https://epsi-rns.gitlab.io/frontend/2020/10/11/slides-concept-css/>

Bilamana sohib pembaca membutuhkan panduan langkah demi langkah, maka youtube.com adalah rujukan yang cukup bagus. Banyak kanal yang bagus yang dapat ditonton, baik yang berbahasa Inggris, maupun yang berbahasa Indonesia.

Keahlian Dasar Wajib di Komunitas

- ▶ Berdiskusilah di group, dan hindari jalur pribadi kalau tidak kenal.
 - ▶ Tips: Buatlah catatan harian.
1. Bahasa Inggris wajib paham.
 2. Mengetahui cara mencari di [google.com](https://www.google.com) atau stackoverflow.com.
 3. Mampu membaca dokumentasi resmi dan buku panduan *manual*.
 4. Mengetahui cara membikin tangkapan layar (*screenshot*) untuk group.
 5. Mengetahui cara berkomunikasi dengan baik di dalam group.

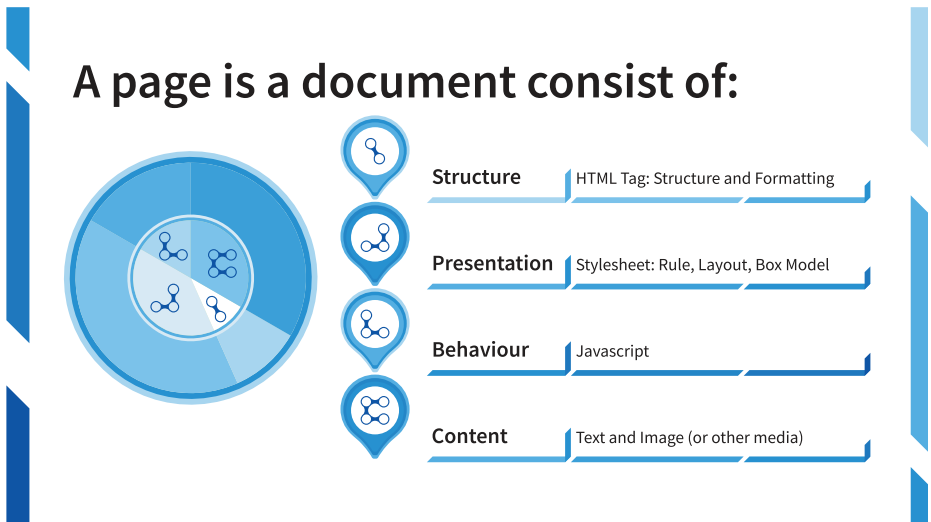
Pendekatan

- ▶ *Stack* secara umum: `html+css+js`.

Terdiri Dari Apakah Halaman Web?

Suatu halaman web sebetulnya hanya berisi ini

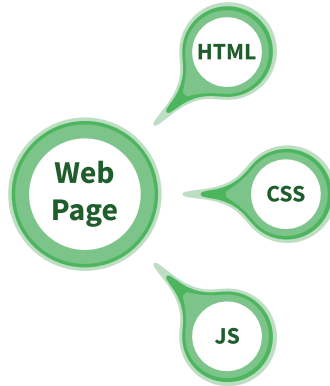
1. *Structure + Presentation + Behaviour*
2. *Custom User Content*: Teks dan gambar (atau media lain).



Yang pertama adalah, *stack* yang kita kenal sebagai `html+css+js`. Sedangkan yang kedua adalah, isi apa saja yang dapat dimasukkan ke dalam berkas halaman tersebut.

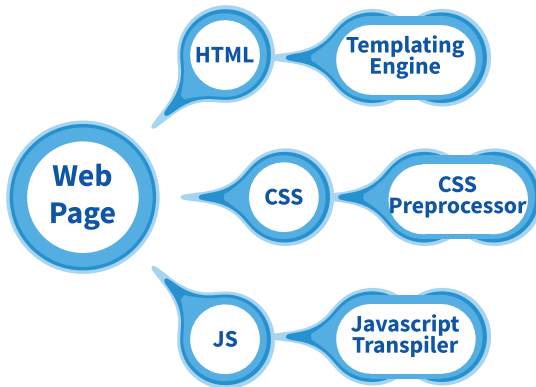
Secara Teknis

- Bagaimana cara menerangkan dari hal dasar ke era modern?



Ikatan Tersembunyi

- Tidak adanya panduan lengkap mengenai *template engine*.

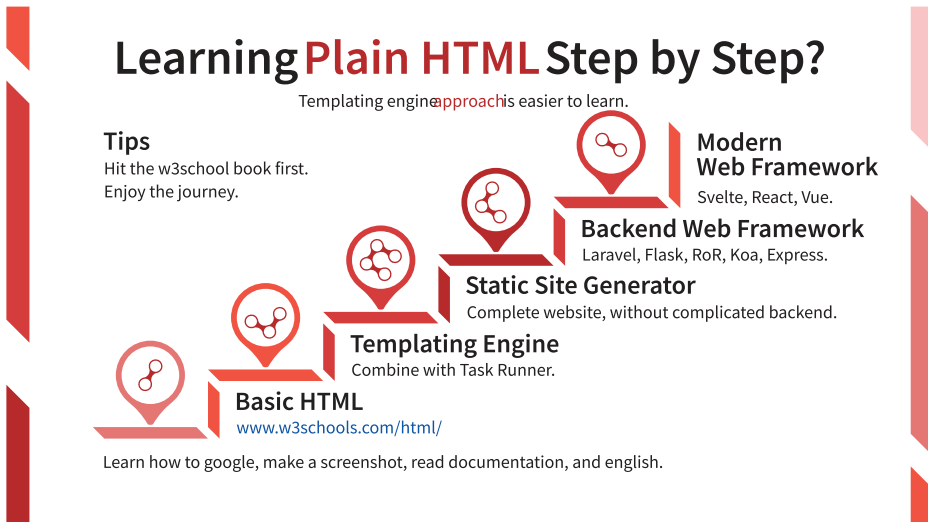


Ya, ada celah kekurangan, di dalam *tutorial* untuk *web development* pada umumnya.

HTML

- Lebih mudah untuk belajar HTML yang murni, dengan pendekatan *template engine*.

HTML5 adalah topik yang luas. HTML5 bukan hanya terkait HTML tag. Pertimbangkanlah untuk mempelajari dahulu hanya bagian dasarnya saja, kemudian langsung praktekkan untuk membuat berkas HTML, di dalam suatu proyek pribadi yang sederhana. *Blog* atau *portfolio*, adalah pilihan bagus untuk proyek pribadi yang sederhana. Ini disebabkan karena untuk membikin suatu *blog*, tidak ada beban untuk membuat *database* yang rumit, dan juga tidak ada kebutuhan untuk melakukan autentikasi melalui *login*.



Langkah-langkah-nya adalah semudah berikut ini:

1. Dasar HTML: w3schools.com/html/.
2. *Template Engine*: Bersama dengan *Task Runner*.
3. *Static Site Generator*: Situs lengkap, tanpa *backend* yang rumit.
4. *Backend Web Framework*: [Laravel](#), [Flask](#), [RoR](#), [Koa](#), [Express](#).
5. *Modern Web Framework*: [Svelte](#), [React](#), [Vue](#).

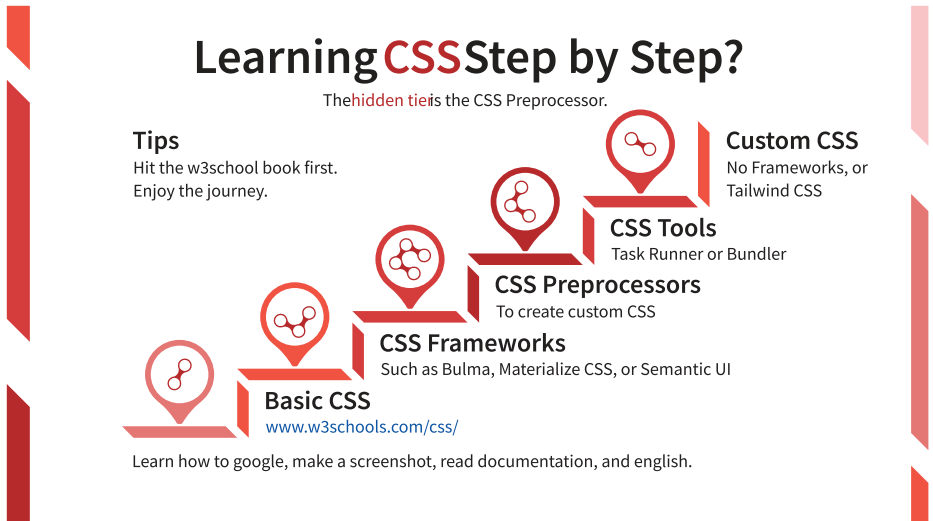
HTML bukanlah bahasa pemrograman. *Template Engine*, memiliki kemampuan pemrograman sederhana yang sengaja dibuat terbatas.

Template engine adalah bidang backend.

Stylesheet

- Bagian yang tersembunyi adalah *CSS Preprocessors*.

Sohib pembaca dapat belajar menggunakan *CSS Framework*, lalu belajar menggunakan *CSS Preprocessor*, lalu kembali untuk melakukan kustomisasi di *CSS Framework*.



Langkah-langkah-nya adalah semudah berikut ini:

1. Dasar *CSS*: w3schools.com/css/
2. *CSS Frameworks*: Misalnya *Bulma*, *Materialize CSS*, atau *Semantic UI*.
3. *CSS Preprocessors*: Untuk membikin tambahan kustomisasi *CSS*
4. *CSS Tools*: *Task Runner*, ataupun *Bundler*.
5. *Custom CSS*: Tanpa *Framework*, atau dengan *Tailwind CSS*.

Yang perlu diperhatikan sohib pembaca adalah *CSS* juga bukan bahasa pemrograman. Demikian pula *CSS Preprocessor* juga tidak dirancang untuk memiliki kemampuan pemrograman secara lengkap. Kemampuannya memang sengaja dibuat tetap terbatas. Dipertahankan tetap sederhana.

Stylesheet adalah bidang frontend.

Javascript

- ▶ tidak dibahas di sini.

Javascript adalah topik yang sangat luas, dan layak mendapatkan penjelasan yang khusus.

Content

Content ini dapat berisi macam-macam, misalnya untuk sistem informasi content dapat berupa *database*. Sedangkan di media sosial, *content* dapat berupa *chat*.

Untuk proyek sederhana, misalnya *blogging* ataupun pembikinan *portfolio*, maka sohib harus membikin *content* sendiri. Karena itu perlu ada keahlian tambahan, yaitu untuk mendapat teks yang tepat dan mengolah gambar dengan baik.

1. Teks

- ▶ Belajar menulis esai.
- ▶ Bahasa Inggris, ataupun bahasa Indonesia, yang tepat.

2. Gambar

- ▶ Mengambil gambar dengan *smartphone*.
- ▶ Membikin ilustrasi *raster*: belajar menggunakan *GIMP*.
- ▶ Membikin ilustrasi *vektor*: belajar menggunakan *Inkscape*.

Yang Tidak Tertulis

Selain hal-hal di atas, ada hal lain yang perlu dipelajari bila ingin menjadi *web developer*, yaitu kemampuan menggunakan *terminal* atau *CLI* (*command line interface*). Terutama *git*.

Maka biasakanlah diri menggunakan *terminal*. Nyamankan terminal di mana sohib pembaca bekerja, dengan melakukan kustomisasi seperlunya.

Pilihan Perangkat

Bilamana memiliki PC atau notebook, maka pasanglah `Linux`, walaupun memakai `Windows` juga boleh saja awalan. `Linux` sangat nyaman dipakai untuk melakukan pemrograman, terutama ketika berhadapan dengan terminal. Demikian pula `Macintosh`. Sementara `Windows` memiliki keterbatasan dalam kaitannya dengan penggunaan `terminal`.

Bilamana hanya memiliki `smartphone` berjenis `android`, bisa saja sebagai awalan belajar, namun sohib harus belajar berdamai dengan keterbatasan, dengan memilih `tools` yang tepat. Kenyataannya banyak yang berhasil membikin blog sendiri hanya berbekal `termux` di `handphone`. Maka jangan jadikan keterbatasan sebagai halangan.

Peran

Jangan lupakan juga mengenai peran, karena walaupun tampaknya sepele, namun orang sering salah pengertian antara dua hal berikut di bawah. Tetapkan secara jujur dengan jelas peran sohib ada di mana.

1. Apakah sohib ingin membikin sesuatu? atau
2. Apakah sohib ingin orang lain untuk membikinkan sesuatu untuk sohib.

Saya ingin menjadi pembikin sesuatu.

Maka sohib harus belajar cara membikin sesuatu. Salah satu caranya adalah membaca buku ini.

Saya ingin seseorang membikinkan sesuatu untuk saya.

Maka sohib harus memperkerjakan pegawai. Atau membeli aplikasi dari `software house`. Dan tentunya tidak terlalu perlu untuk membaca buku ini secara rinci.

Penutup Bab Pertama

Pahami konsepnya, alih-alih hanya sekedar belajar cara memakai sesuatu.

Mari bersenang-senang dengan melakukan coding.

Bab Kedua

Menggunakan SASS

*Mulai dari yang mudah dan populer
Makan bubur mulai dari pinggirnya.*

Mari kita memulai dengan mempersiapkan lingkungan SASS, baru kemudian masuk ke urusan *coding*.

Mempersiapkan Lingkungan

Penulis sudah lama menjadi pengguna GNU/Linux, yang untuk selanjutnya saya sebut saja dengan kata Linux, karena tidak semua Linux memakai GNU. Yang lebih penting adalah pembaca paham maksudnya. Ternyata Linux sangat memudahkan *programmer* untuk menjadi nyaman dalam keseharian. Bagaimanapun penulis juga sadar, kalau kebanyakan pemula adalah pengguna Windows. Maka penulis mencoba sebisanya mempersiapkan lingkungan di Windows. Kebetulan penulis tidak pernah memakai Windows 10. Maka harap maklum kalau contoh yang diberikan adalah Windows 7.

Command Line Interface

Lingkungan apa yang perlu dipersiapkan? Yaitu membiasakan diri menggunakan CLI (*command line interface*) ataupun dikenal juga dengan istilah *terminal shell*. Kalau di Windows dapat memakai MS-DOS *prompt* (*cmd*), atau dengan *powershell*. Sedangkan di Linux pilihannya banyak karena sifatnya yang modular. Misalnya shell dapat menggunakan *bash*, atau *zsh*. Sedangkan *terminal*-nya dapat memakai *xfce4-terminal*, *urxvt*, atau *st* (*simple terminal*).

Lingkungan Windows 7

Dukungan *terminal* di Windows 7 masih sangat terbatas. Untungnya ada beberapa peralatan tambahan dari pihak ketiga yang dapat kita pakai:



- ▶ *cmder*: *terminal* yang dengan tampilan *cantique*, sebagai pengganti *cmd* di Windows 7.
- ▶ *chocolatey*: *package manager* untuk Windows
- ▶ *scoop*: *installer* alternatif, selain dari *chocolatey*.

Setelah beberapa saat, beberapa aplikasi tambahan yang tersebut dapat membuat **Windows** nyaman dipakai lagi. Sebagai kombinasi dari **choco** and **cmd**, jadinya kita memiliki *package manager* di dalam *terminal* yang *cantique*. Sebagai pengguna **Linux**, saya merasa seolah berada di rumah lagi.

Choco Package Manager

Seperti yang telah saya sampaikan sebelumnya, sebetulnya ada dua alternatif yang fungsinya kurang lebih sama.

- ▶ chocolatey.org: *Package Manager* untuk **Windows**.
- ▶ scoop.sh: *Installer* untuk **Windows**.

Penulis pilihkan **choco** yang kegunaannya lebih luas.

Install Choco

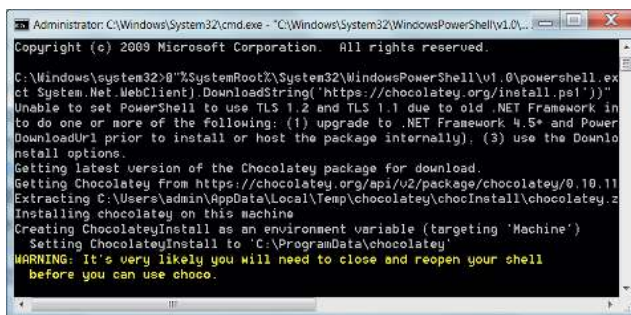
Langsung saja ke situsnya:

- ▶ <https://chocolatey.org/docs/installation#more-install-options>

Sohib tentunya dapat melihat perintah panjang *command line* yang dibutuhkan di website, yaitu untuk dijalankan di **cmd**.

```
>_ @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" bla
    bla bla ....
```

Sekarang bukalah **cmd** dengan *privilege* sebagai *administrator*, lalu salin-tempel (*copy-paste*) perintah tersebut. Jangan lupa untuk menekan tombol **enter** untuk menjalankan perintah tersebut.



```
Administrator: C:\Windows\System32\cmd.exe - "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

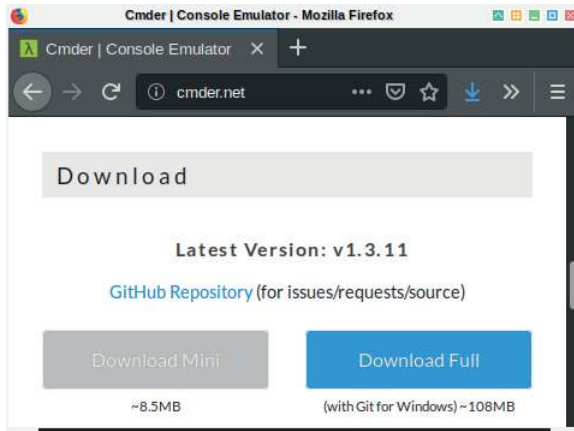
C:\Windows\system32>@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe"
[System.Net.WebClient].DownloadString('https://chocolatey.org/install.ps1')
Unable to set PowerShell to use TLS 1.2 and TLS 1.1 due to old .NET Framework in
to do one or more of the following: (1) upgrade to .NET Framework 4.5+ and Power
DownloadUrl prior to install or host the package internally), (3) use the Downlo
install options.
Getting latest version of the Chocolatey package for download.
Getting Chocolatey from https://chocolatey.org/api/v2/package/chocolatey/0.10.11
Extracting C:\Users\admin\AppData\Local\Temp\chocolatey\chocInstall\chocolatey.z
Installing chocolatey on this machine
Creating ChocolateyInstall as an environment variable (targeting 'Machine')
Setting ChocolateyInstall to "C:\ProgramData\chocolatey"
WARNING: It's very likely you will need to close and reopen your shell
before you can use choco.
```

Cmdr Console

Sekarang bagian yang membuat sohib nampak sakti, yaitu *terminal*.

► cmdr.net.

Sebagaimana terlihat, ada dua pilihan:



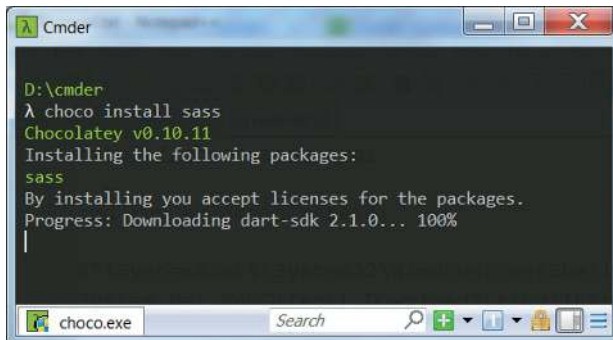
Pilih benar satu dari dua pilihan tersebut, unduh, ekstrak, dan buka. Berikutnya sohib dapat melakukan *pin* ke *taskbar* untuk supaya *cmdr* mudah digunakan. Sohib juga dapat menggunakan *cmdr* sebagai *administrator*, kapanpun dibutuhkan.

Di antara keduanya, penulis mendapatkan bahwa pilihan *cmdr mini* sudah cukup. Tentunya ada rinciannya, namun buku ini bukan tempatnya.

Memasang SASS

Memasang SASS cukup mudah.

```
$ choco install sass
```

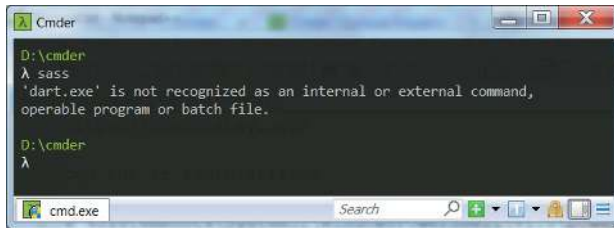


Pemulis menggunakan tanda *\$* sebagai command prompt yang umum di *Linux*. Di *Windows* biasanya memakai tampilan *C:>*.

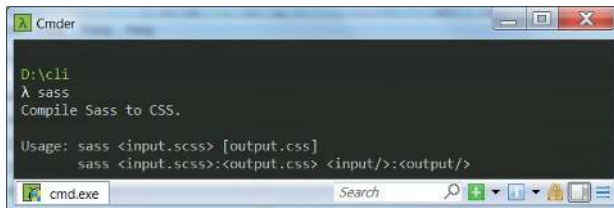
Banyak implementasi **SASS**. Penulis menggunakan **SASS** yang dibikin di atas bahasa **dart** yang tersedia di daftar paket di **chocolatey**. Sebagai alternatif, sohib dapat menggunakan **node-sass** yang dapat dipasang melalui **npm**.

Sohib mungkin mengalami hal tersebut di bawah, tepat saat pertama kali menjalankan **dart**.

```
>_ $ sass
'dart.exe' is not recognized as an internal or external command,
operable program or batch file.
```



Cara penyelesaiannya adalah cukup dengan menutup *terminal*, dan buka **SASS** di terminal yang baru. Maka **SASS** akan berjalan dengan baik-baik saja.



Penulis tidak ingin berkulat terlalu lama di bagian pemasangan, supaya pembaca dapat segera melakukan praktek.

Lingkungan Linux



Seperti telah disampaikan sebelumnya. Ada beberapa implementasi SASS. Selayaknya budaya *RTFM* (*read the fine manual*) di komunitas *Linux*, sebelum kita masuk lebih dalam, ada baiknya kita membaca dahulu dokumentasi resminya.

- ▶ sass-lang.com.

Beberapa Pilihan

Ada beberapa pilihan untuk mengkompilasi SASS.

- ▶ *Compiler* Umum: *ruby-sass*, *dart-sass*, dan *node-sass*.
- ▶ *LibSass Wrapper*: *SassC*, *sass.cr*, *go-libsass*, *jsass*, *sass.js*, *lua-sass*, *libsass-net*, *node-sass*, *CSS::Sass*, *SassPHP*, *libsass-python*, *sassc-ruby*, *sass_rs*, dan *Sass-Scala*.
- ▶ *Task Runner*: *Gulp*, *Grunt*. Dan juga *bundler*: *Rollup*, *Webpack*, and *Parcel*.

1: ruby-sass, yang kadaluarsa

Secara resmi, *Ruby Sass* sudah usang (*deprecated*). *Ruby Sass* ini sangat umum, dan ada berbagai tutorial tersedia di internet, maka tidak perlu dibahas lagi di sini. Bilamana ingin tahu, baca saja dokumentasi resminya.

- ▶ sass-lang.com/ruby-sass.

Mari kita lihat saja baris perintahnya di terminal supaya kita dapat membedakan antara satu implementasi dengan lainnya.

```
>_ $ sass \
    --watch -I sass/themes/oriclone:static/assets/css \
    --style compressed --sourcemap=none
```

Penulis menggunakan *backslash* (\) untuk memisahkan perintah panjang, sebagaimana umum dilakukan di *bash*.

2: dart-sass

Saat ini `ruby-sass` yang usang, sudah digantikan dengan `dart-sass`. Dokumentasi resmi juga ada:

► sass-lang.com/dart-sass.

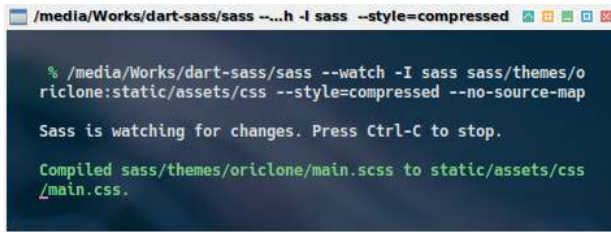
Memasang dart juga cukup mudah, langsung saja unduh, ekstrak dan langsung dapat dijalankan.

► <https://github.com/sass/dart-sass/releases/tag/1.29.0>

Mari kita lihat lagi baris perintahnya di terminal

>_

```
$ /media/Works/dart-sass/sass --watch \
-I sass sass/themes/oriclone:static/assets/css \
--style=compressed --no-source-map
```



Demi kemudahan penggunaan *terminal* sehari-hari, maka sohib dapat menggunakan alias, yaitu di `.bashrc` ataupun di `.zshrc`.

```
alias dart-sass=/media/Works/dart-sass/sass
```

3: node-sass

Pemakaian `node-sass` adalah sangat umum, dan sudah banyak tutorial di internet. Seperti biasa, langsung saja jenguk dokumentasi resminya.

- ▶ sass-lang.com/install

Bilamana sohib adalah pemula di bidang `NodeJS`, maka sohib perlu memahami mengenai pengaturan lingkungan `NPM`.

- ▶ Tidak perlu `sudo`!

Saat menjadi pemula di bidang `NodeJS`, penulis berulang kali terbentur masalah terkait *permission*. Sampai penulis menemukan artikel ini

- ▶ <https://docs.npmjs.com/resolving-eacces-permissions-errors-when-installing-packages-globally>

Pada dasarnya, ini hanyalah mengatur *global path* untuk pengguna Linux tanpa *root privilege*.



```
epsi@andalan: ~  
% mkdir ~/.npm-global  
% npm config set prefix '~/.npm-global'  
% export PATH=~/.npm-global/bin:$PATH  
%  
~ andalan  
~ andalan  
~ andalan  
~ andalan
```

Jangan lupa menambahkan di `.bashrc` ataupun di `.zshrc`.

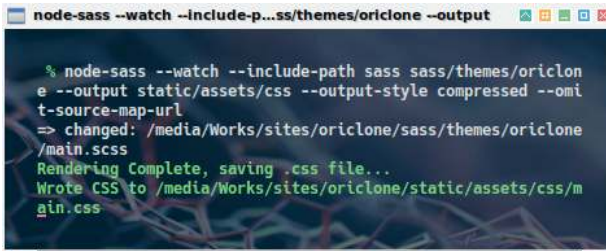
```
export PATH="$PATH:$HOME/.npm-global/bin"
```

Dengan cara ini maka, pengguna biasa dapat memasang `node-sass` dengan bebas, tanpa membutuhkan perintah `sudo`.

```
$ npm install -g node-sass
```

Sekarang kita dapat menggunakan `node-sass` di dalam *terminal*, dengan pilihan *parameter* yang berbeda dengan `ruby-sass` sebagaimana berikut:

```
>_ $ node-sass --watch \  
    --include-path sass sass/themes/oricloner \  
    --output static/assets/css \  
    --output-style compressed \  
    --omit-source-map-url
```



```
node-sass --watch --include-p...ss/themes/oriclone --output
% node-sass --watch --include-path sass sass/themes/oriclone
e --output static/assets/css --output-style compressed --omit-source-map-url
=> changed: /media/Works/sites/oriclone/sass/themes/oriclone
/main.scss
Rendering Complete, saving .css file...
Wrote CSS to /media/Works/sites/oriclone/static/assets/css/main.css
```

Berulangkali mengetik di *terminal* rawan dari kesalahan, maka ada baiknya perintah tersebut kita bungkus dalam bentuk *script* di dalam berkas *packages.json*. Salah satu contoh yang bagus ada di dokumentasi Bulma.

► <https://bulma.io/documentation/customize/with-node-sass/>

Penulis ringkas saja dan ambil yang perlu sebagai contoh yaitu:

```
npm
{
  "scripts": {
    "css-build": "node-sass
      --include-path sass sass/themes/oriclone
      --output static/assets/css",
    "css-watch": "npm run css-build --
      --watch --output-style compressed
      --omit-source-map-url",
    "start": "npm run css-watch"
  },
}
```

Tentu saja untuk menjalankan script di atas, perlu disiapkan directory-nya, dan semua perintah dalam tanda kutip berada dalam satu baris.

Dengan pengaturan sebagaimana di atas, sekarang kita cukup melakukan perintah pendek ini.

```
$ npm run css-watch
```

Bagaimana pengaturan yang sesuai, tentunya terserah saja kepada pembaca, karena ini sangat tergantung keadaan setempat. Semua orang memiliki *use case* yang berbeda, karena itu kebutuhannya berbeda pula.

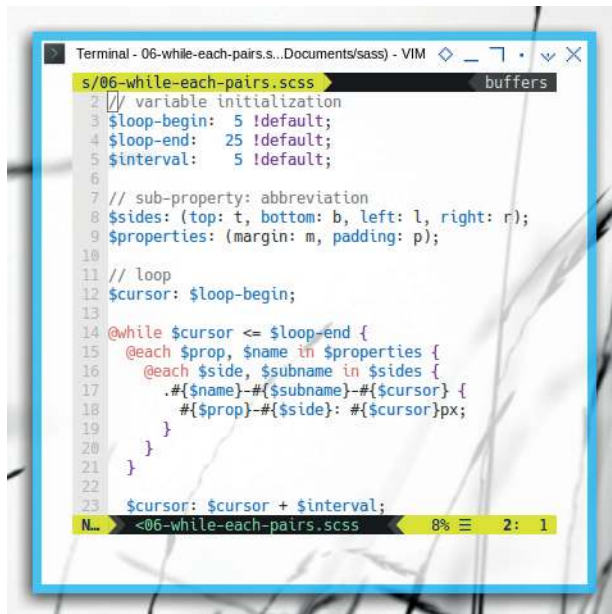
*Langkah demi langkah, unjuk kemampuan dari SASS,
untuk menghasilkan margin class maupun padding class.*

Spacing Class dengan SCSS

- **Tujuan:** Menghasilkan spacing class dengan loop di SASS.

Pendahuluan

Pembikinan theme, membutuhkan *custom CSS*. Ini juga berlaku, bahkan saat menggunakan *CSS Framework*. Salah satu yang penulis butuhkan di masa lalu adalah membuat *spacing class* di *Bulma* maupun *Materialize CSS*. Walaupun akhirnya di tahun 2019, *Bulma 0.8* telah memiliki *spacing class* sendiri, namun penulis tetap membutuhkan untuk proyek lainnya. *Spacing class* ini dapat dibikin dengan menggunakan *SASS*.



```
s/06-while-each-pairs.scss buffers
2 // variable initialization
3 $loop-begin: 5 !default;
4 $loop-end: 25 !default;
5 $interval: 5 !default;
6
7 // sub-property: abbreviation
8 $sides: (top: t, bottom: b, left: l, right: r);
9 $properties: (margin: m, padding: p);
10
11 // loop
12 $cursor: $loop-begin;
13
14 @while $cursor <= $loop-end {
15   @each $prop, $name in $properties {
16     @each $side, $subname in $sides {
17       .#{$name}-#{$subname}-#{$cursor} {
18         #{$prop}-#{$side}: #{$cursor}px;
19       }
20     }
21   }
22   $cursor: $cursor + $interval;
23 }
N. <06-while-each-pairs.scss 8% 2: 1
```

Panduan ini akan menghemat banyak waktu *coding*, karena sudah diberikan contoh. Dan tentunya juga menghemat banyaknya ketikan, karena dengan otomasi maka tidak perlu mengetik *CSS* secara *manual*.

Bacaan Rujukan

Dasar Perulangan (*looping*)

- ▶ <http://thesassway.com/intermediate/if-for-each-while>

Sebagian besar *coding* di bab ini terilhami oleh:

- ▶ github.com/jmaczan/.../margin-padding.sass

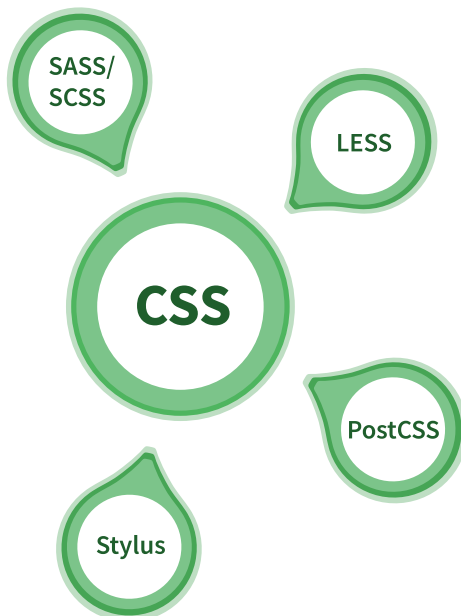
Bacaan lain:

- ▶ <https://alexwenzel.de/2017/707/sass-spacing-helper-classes>

Keluarga CSS Preprocessor

SASS memiliki dua format, yang pertama adalah **.scss** yang menggunakan kurung kurawal dan memisahkan dengan titik koma (*semicolon*), lalu yang kedua adalah **.sass**. yang mengandalkan indentasi. Untuk pembelajaran kita pakai jenis yang pertama, yang secara *syntax* mirip dengan **CSS**.

SASS/SCSS adalah salah satu alternatif diantara beberapa keluarga **CSS preprocessor**. Kebetulan **SASS/SCSS** ini adalah juga yang paling digemari karena dipakai di **Bootstrap**, **Bulma**, maupun **Materialize CSS**. Selanjutnya, mari langsung kita mulai saja. Praktek dengan contoh nyata.



1: @for

Bayangkan sohib pembaca membutuhkan beberapa kelas (*class*) yang berurut untuk menangani *margin*, sebagaimana *CSS* berikut:



```
.m-1 {  
  margin: 1px;  
}
```

Hasil jadi dari naskah di atas dapat dicapai dengan perulangan, tepatnya adalah dengan *loop constructor* sebagaimana berikut di bawah:

```
@for ... from ... through ... {  
  ...  
}
```

Penamaan *CSS Class* yang ingin dicapai adalah sebagai berikut:

```
.m-#{ $number }
```

Supaya nyaman maka perlu jelas, kita perlu menentukan nilai awal dari perulangan.



```
// variable initialization  
  
$loop-begin: 1 !default;  
$loop-stop: 3 !default;  
  
// loop  
  
@for $cursor from $loop-begin through $loop-stop {  
  .m-#{ $cursor } {  
    margin: #{ $cursor }px;  
  }  
}
```

Di dalam perulangan tersebut kita menjumpai interpolasi variabel yaitu *#{...}*, yang digunakan untuk meng-ekstrak suatu nilai ke dalam *CSS*.

► <https://sass-lang.com/documentation/interpolation>

Dengan `#{$cursor}` menangani isi dari perulangan, maka kita mendapatkan hasil sebagaimana berikut:



```
.m-1 {
  margin: 1px;
}

.m-2 {
  margin: 2px;
}

.m-3 {
  margin: 3px;
}
```

Yang menjadi keterbatasan dari `@for` adalah kita tidak dapat menggunakan *interval*. Kita dapat menghasilkan nilai berurut yaitu `[1, 2, 3, 4, 5]`, namun kita tidak dapat menghasilkan nilai berurut berupa `[5, 10, 15, 20, 25]`.

2: @while

Untuk menyelesaikan permasalahan nilai berurut `[5, 10, 15, 20, 25]` maka kita dapat menggunakan `@while`, sebagaimana di bawah ini:



```
$cursor: $loop-begin;

@while $cursor <= $loop-end {
  ...

  $cursor: $cursor + $interval;
}
```

Penamaan *CSS Class* yang ingin dicapai masih sama, yaitu sebagai berikut:

```
.m-#{ $number }
```

Maka kode sumber dari perulangan `@while`-nya adalah sebagaimana berikut:

Sass

```
// variable initialization
$loop-begin: 5 !default;
$loop-end: 25 !default;
$interval: 5 !default;

// loop
$cursor: $loop-begin;

@while $cursor <= $loop-end {
  .m-#{$cursor} {
    margin: #{$cursor}px;
  }
  $cursor: $cursor + $interval;
}
```

Sehingga kita mendapatkan hasil naskah `CSS` seperti tampak di bawah:

E

```
.m-5 {
  margin: 5px;
}

.m-10 {
  margin: 10px;
}

.m-15 {
  margin: 15px;
}

.m-20 {
  margin: 20px;
}

.m-25 {
  margin: 25px;
}
```

3: @each

Sebelum melanjutkan ke situasi yang lebih rumit, kerjakan contoh yang sederhana terlebih dahulu. Tantangannya adalah membikin *margin property* yang memiliki beberapa jenis varian, sebagaimana berikut:

- ▶ `margin`,
- ▶ `margin-top`,
- ▶ `margin-bottom`,
- ▶ `margin-left`,
- ▶ `margin-right`.

Untuk kesederhanaan, maka penulis tidak mengikutsertakan varian `margin`.

Penamaan *CSS Class* yang ingin dicapai berbeda, yaitu sebagai berikut:

```
.m-#{ $side }-5
```

Variabel *SASS* dapat menangani deklarasi *list*, yaitu:

```
$sides: (top, bottom, left, right);
```

Sedangkan daftar list tersebut dapat di-akses dengan *iterator*, yaitu `@each`:

```
@each ... in ... {
  ...
}
```

Mari kita lihat bagaimana cara menangani variabel `$sides` ini:



```
// property
$sides: (top, bottom, left, right);

@each $side in $sides {
  .m-#{ $side }-5 {
    margin-#{ $side }: 5px;
  }
}
```

Sekarang kita dapat melihat naskah hasil jadinya sebagaimana di bawah:



```
.m-top-5 {  
    margin-top: 5px;  
}  
  
.m-bottom-5 {  
    margin-bottom: 5px;  
}  
  
.m-left-5 {  
    margin-left: 5px;  
}  
  
.m-right-5 {  
    margin-right: 5px;  
}
```

Ternyata naskah hasil jadinya masih tidak sesuai dengan harapan, karena yang kita inginkan adalah berbentuk `.m-t-5`, alih-alih berbentuk `.m-top-5`.

4: @each untuk Pasangan Nilai

SASS memiliki solusi untuk permasalahan ini.

Penamaan *CSS Class* yang ingin dicapai adalah menggunakan singkatan, yaitu sebagai berikut:

```
.m-#{ $abbreviation }-5.
```

Untungnya SASS memperbolehkan kita menulis pasangan nilai (*pairs*) dalam daftar *list*. Sehingga kita mendapatkan *sub-property* sebagaimana berikut:

```
$sides: (top: t, bottom: b, left: l, right: r);
```

Mari kita tulis ulang kode sebelumnya menjadi sebagai berikut:



```
// property: abbreviation
$sides: (top: t, bottom: b, left: l, right: r);

@each $side, $name in $sides {
  .m-#{$name}-5 {
    margin-#{$side}: 5px;
  }
}
```

Sekarang kita mendapatkan naskah hasil jadi yang kita inginkan.



```
.m-t-5 {
  margin-top: 5px;
}

.m-b-5 {
  margin-bottom: 5px;
}

.m-l-5 {
  margin-left: 5px;
}

.m-r-5 {
  margin-right: 5px;
}
```

Perhatikan bahwa, penulis menggunakan `.m-t-5`, alih-alih `.mt-5`, untuk membedakan dengan spacing classes yang ada di salah satu *CSS Frameworks* yaitu *Bootstrap*.

5: @each di dalam @while

Pertimbangkan untuk meningkatkan kode untuk situasi yang lebih rumit. Kita membutuhkan varian dari kelas-kelas tadi untuk beberapa jarak angka yang berbeda secara berurutan.

Penamaan *CSS Class* yang ingin dicapai adalah sebagai berikut:

```
.m-#{ $subname }-#{ $number }
```

Perhatikan kerangka kode-nya. Kita meletakkan *iterator*, yaitu @each ke dalam perulangan @while.

```
@while $cursor <= $loop-end {  
  
  @each $side, $name in $sides {  
    ...  
  }  
  
  $cursor: $cursor + $interval;  
}
```

Maka kode sumber SASS selengkapnya adalah sebagai berikut:

Sass

```
// variable initialization  
$loop-begin: 5 !default;  
$loop-end: 25 !default;  
$interval: 5 !default;  
  
// sub-property: abbreviation  
$sides: (top: t, bottom: b, left: l, right: r);  
  
// loop  
$cursor: $loop-begin;
```


Karena panjang, kita lanjutkan di halaman berikutnya.



```
@while $cursor <= $loop-end {

  @each $side, $name in $sides {
    .m-#{ $name }-#{ $cursor } {
      margin-#{ $side }: #{ $cursor }px;
    }
  }

  $cursor: $cursor + $interval;
}
```

Sekarang mulai dapat kita nikmati hasilnya secara perlahan yaitu beberapa kelas secara berurut:



```
.m-t-5 {
  margin-top: 5px;
}

.m-b-5 {
  margin-bottom: 5px;
}

.m-l-5 {
  margin-left: 5px;
}

.m-r-5 {
  margin-right: 5px;
}

.m-t-10 {
  margin-top: 10px;
}
```

BAB DUA - MENGGUNAKAN SASS

```
.m-b-10 {  
  margin-bottom: 10px;  
}  
  
.m-l-10 {  
  margin-left: 10px;  
}  
  
.m-r-10 {  
  margin-right: 10px;  
}  
  
...  
  
...  
  
.m-t-25 {  
  margin-top: 25px;  
}  
  
.m-b-25 {  
  margin-bottom: 25px;  
}  
  
.m-l-25 {  
  margin-left: 25px;  
}  
  
.m-r-25 {  
  margin-right: 25px;  
}
```

Sengaja penulis potong di bagian tengah supaya tidak memenuhi isi buku.

6: @while Bersarang

Mari kita tambah kesulitannya.

Penamaan *CSS Class* yang ingin dicapai adalah sebagai berikut:

```
.#{$name}-#{$subname}-#{$number}
```

Kita tentukan dahulu deklarasi *list* untuk tiap-tiap *property* yang akan dihasilkan, yaitu:

- *Property*: *margin* dan *padding*.
- Masing-masing memiliki *sub-property*: *top*, *bottom*, *left*, *right*.

Tepatnya disajikan sebagai berikut:

```
$sides: (top: t, bottom: b, left: l, right: r);
$properties: (margin: m, padding: p);
```

Perhatikan kerangka kode-nya. Kita meletakkan dua *iterator* secara bersarang, yaitu *@each* ke dalam perulangan *@while*.

```
@while $cursor <= $loop-end {

    @each $prop, $name in $properties {
        @each $side, $subname in $sides {

            ...

        }
    }

    $cursor: $cursor + $interval;
}
```

Selanjutnya seperti biasa, yaitu kode sumber [SASS](#) selengkapnya sebagai berikut:



```
// variable initialization
$loop-begin: 5 !default;
$loop-end: 25 !default;
$interval: 5 !default;

// sub-property: abbreviation
$sides: (top: t, bottom: b, left: l, right: r);
$properties: (margin: m, padding: p);

// loop
$cursor: $loop-begin;

@while $cursor <= $loop-end {
  @each $prop, $name in $properties {
    @each $side, $subname in $sides {
      .#{$name}-#{$subname}-#{$cursor} {
        #{$prop}-#{$side}: #{$cursor}px;
      }
    }
  }
  $cursor: $cursor + $interval;
}
```

Hasilnya bukan hanya [margin](#), namun juga [padding](#).



```
.m-t-5 {
  margin-top: 5px;
}

.m-b-5 {
  margin-bottom: 5px;
}
```



```
.m-l-5 {  
  margin-left: 5px;  
}  
  
.m-r-5 {  
  margin-right: 5px;  
}  
  
.p-t-5 {  
  padding-top: 5px;  
}  
  
.p-b-5 {  
  padding-bottom: 5px;  
}  
  
.p-l-5 {  
  padding-left: 5px;  
}  
  
.p-r-5 {  
  padding-right: 5px;  
}  
  
.m-t-10 {  
  margin-top: 10px;  
}  
  
.m-b-10 {  
  margin-bottom: 10px;  
}  
  
...
```

Dan selanjutnya. Sengaja penulis potong lagi supaya tidak memenuhi isi buku.

Menambah Properti Utama

Menjelang *final*. Mari kita sempurnakan lagi, dengan `margin` dan `padding`, tanpa *sub-property*.

Penamaan *CSS Class* yang ingin dicapai adalah sebagai berikut:

```

.#{$name}-#{$number}
.#{$name}-#{$subname}-#{$number}

```

Masih dengan meletakkan dua *iterator* secara bersarang, yaitu `@each` ke dalam perulangan `@while`. Kerangkanya sedikit berubah.

```

@while $cursor <= $loop-end {

    @each $prop, $name in $properties {
        ...

        @each $side, $subname in $sides {
            ...
        }
    }

    $cursor: $cursor + $interval;
}

```

Selanjutnya seperti biasa, yaitu kode sumber *SASS* selengkapnya.

Sass

```

// variable initialization
$loop-begin: 5 !default;
$loop-end: 25 !default;
$interval: 5 !default;

// sub-property: abbreviation
$sides: (top: t, bottom: b, left: l, right: r);
$properties: (margin: m, padding: p);

// loop
$cursor: $loop-begin;

```

Kita lanjutkan di halaman berikutnya, karena kode-nya panjang.



```
@while $cursor <= $loop-end {
  @each $prop, $name in $properties {
    .#{$name}-#{$cursor} {
      #{$prop}: #{$cursor}px !important;
    }

    @each $side, $subname in $sides {
      .#{$name}-#{$subname}-#{$cursor} {
        #{$prop}-#{$side}: #{$cursor}px !important;
      }
    }
  }

  $cursor: $cursor + $interval;
}
```

Perhatikan, bahwa kita juga menerapkan aturan *rule*, yaitu `!important` ke dalam value dari CSS di masing-masing kelas. Sehingga hasilnya sebagai berikut:



```
.m-5 {
  margin: 5px !important;
}

.m-t-5 {
  margin-top: 5px !important;
}

...

.p-5 {
  padding: 5px !important;
}

...
```

Dan seterusnya. Kita hampir dapat menggunakan *custom spacing class* ini secara lengkap untuk digunakan di suatu website ataupun proyek pribadi.

Penerapan di Dunia Nyata

Untuk alasan praktis, ada beberapa hal yang perlu diperhatikan.

- ▶ Menggunakan format `.sass`, alih-alih `.scss`, supaya ada contoh bagi pembaca.
- ▶ Memulai dari angka `0` (nol), alih-alih dari `5` (lima), supaya tersedia pengaturan tanpa *margin* dan tanpa *padding*.
- ▶ Menggunakan `X`, dan `Y`, supaya mirip `Bootstrap`. Hal ini memudahkan saat migrasi dari `bootstrap` ke *custom class*.

Hasil naskah jadi, yang ingin kita dapatkan adalah sebagai berikut.



```
.m-y-5 {  
  margin-top: 5px !important;  
  margin-bottom: 5px !important;  
}  
  
.m-x-5 {  
  margin-left: 5px !important;  
  margin-right: 5px !important;  
}
```

Semakin lengkap belum tentu semakin baik. Kekurangan dari cara ini adalah, ukuran *stylesheet* menjadi gemuk, karena ukurannya besar.

Seperti telah disebutkan sebelumnya, `SASS` memperbolehkan pasangan nilai di dalam deklarasi *list*, sehingga sekarang kita memiliki *sub-property* sebagaimana berikut:



```
$sides: (top: t, bottom: b, left: l, right: r)  
$sidesy: (top, bottom)  
$sidesx: (left, right)  
$properties: (margin: m, padding: p)
```

Perhatikan, kalau kode-nya tidak memakai titik-koma dan nantinya juga tidak memakai kurung kurawal. Yang dipakai adalah indentasi.

Kemudian mari kita lihat kerangkanya, perulangan yang dibikin menjadi juga membutuhkan perubahan.



```
// loop
$cursor: $loop-begin

@while $cursor <= $loop-end
  @each $prop, $name in $properties

    .#{$name}-y-#{$cursor}
      @each $side in $sidesy
        #{$prop}-#{$side}: #{$cursor}px !important

    .#{$name}-x-#{$cursor}
      @each $side in $sidesx
        #{$prop}-#{$side}: #{$cursor}px !important

$cursor: $cursor + $interval
```

Berikut kode sumber [SASS](#) selengkapnya, untuk keperluan sehari-hari.



```
// variable initialization
$loop-begin: 0 !default
$loop-end: 25 !default
$interval: 5 !default

// sub-property: abbreviation
$sides: (top: t, bottom: b, left: l, right: r)
$sidesy: (top, bottom)
$sidesx: (left, right)
$properties: (margin: m, padding: p)

// loop
$cursor: $loop-begin

@while $cursor <= $loop-end
  @each $prop, $name in $properties
    .#{$name}-#{$cursor}
      #{$prop}: #{$cursor}px !important

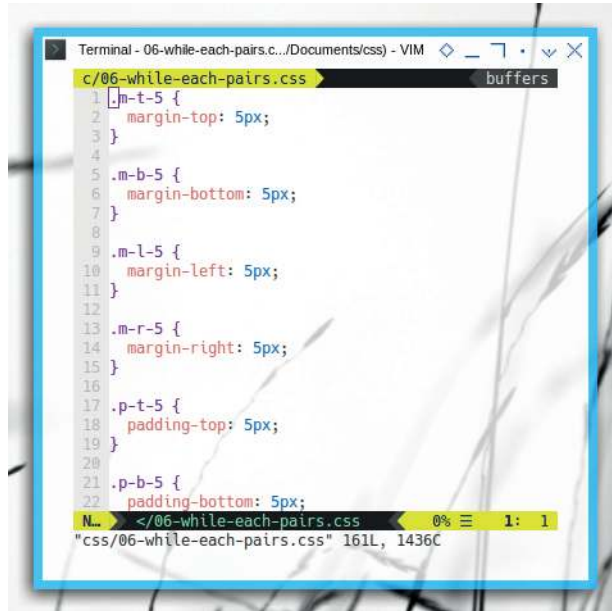
  @each $side, $subname in $sides
    .#{$name}-#{$subname}-#{$cursor}
      #{$prop}-#{$side}: #{$cursor}px !important

  .#{$name}-y-#{$cursor}
    @each $side in $sidesy
      #{$prop}-#{$side}: #{$cursor}px !important

  .#{$name}-x-#{$cursor}
    @each $side in $sidesx
      #{$prop}-#{$side}: #{$cursor}px !important

$cursor: $cursor + $interval
```

Akhirnya *custom spacing classes* ini siap dipakai. Mengenai hasilnya bagaimana, dapat dicoba sendiri sebagai latihan.



```

Terminal - 06-while-each-pairs.c.../Documents/css) - VIM
c/06-while-each-pairs.css buffers
1 .m-t-5 {
2   margin-top: 5px;
3 }
4
5 .m-b-5 {
6   margin-bottom: 5px;
7 }
8
9 .m-l-5 {
10  margin-left: 5px;
11 }
12
13 .m-r-5 {
14  margin-right: 5px;
15 }
16
17 .p-t-5 {
18  padding-top: 5px;
19 }
20
21 .p-b-5 {
22  padding-bottom: 5px;
23 }
N... </06-while-each-pairs.css 0% 1: 1
"css/06-while-each-pairs.css" 161L, 1436C
  
```

Bagaimana menurut sohib pembaca?

Bab Ketiga

Menggunakan LESS

► ses.Pendahuluan

`Less` adalah salah satu contoh keluarga *CSS Preprocessor*, yang patut dimasukkan ke buku ini, karena beberapa alasan:

1. Pertama karena `Less` ini punya nama lengkap `Less.js`, yang berarti satu-satunya implementasi `Less` adalah di `NodeJS`.
2. Kedua, adalah `Less.js` tidak mendukung perulangan (loop), namun memakai fungsi rekursi untuk mendapatkan hasil yang sama.

Perbedaan ini yang menarik untuk dikaji. Kita memang perlu memahami keberagaman supaya tidak terpatok bahwa *CSS Preprocessor* harus seperti implementasi tertentu.

Penyusunan

Bab ini dibagi dua bagian, yaitu:

- *Looping* untuk menghasilkan *spacing classes*.
- *Conditional* untuk menghasillkan *color classes*.

Penerapan Less.js

`Less.js` adalah *CSS Preprocessor* yang dipakai di dalam *CSS Framework* bernama *Semantic UI*. Selain itu tidak banyak penggunaan `Less.js` ini. *Semantic UI* adalah *CSS Framework* yang bagus, namun kurang dalam beberapa hal, salah satunya adalah *spacing class*, yang kita jadikan contoh di buku ini. Karena sifatnya hanya contoh maka tidak diterangkan lagi perihal instalasi pemasangannya. Silahkan cari di situs resminya.

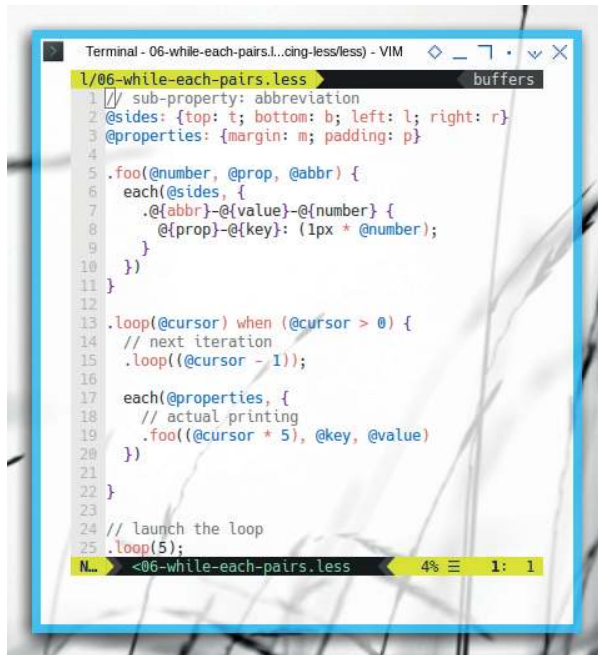
Seperti yang telah penulis katakan di bab sebelumnya, pembikinan theme, membutuhkan *custom CSS*. *spacing class* ini akan kita bikin menggunakan `Less.js`.

Seperti sebelumnya, panduan ini akan menghemat banyak waktu *coding*, karena sudah diberikan contoh. Dan tentunya juga menghemat banyaknya ketikan, karena dengan otomasi maka tidak perlu mengetik *CSS* secara *manual*.

Untuk mempermudah pembelajaran, semua kode di bab ini adalah saduran langsung (*direct port*) dari bab sebelumnya mengenai *SASS*.

Spacing Class dengan LESS

- Tujuan: Menghasilkan spacing class dengan cara rekursif di [Less.js](#).



```

Terminal - 06-while-each-pairs...cing-less/less - VIM
1 // sub-property: abbreviation
2 @sides: {top: t; bottom: b; left: l; right: r};
3 @properties: {margin: m; padding: p};
4
5 .foo(@number, @prop, @abbr) {
6   each(@sides, {
7     .@{abbr}-@{value}-@{number} {
8       @{prop}-@{key}: (1px * @number);
9     }
10  })
11 }
12
13 .loop(@cursor) when (@cursor > 0) {
14   // next iteration
15   .loop((@cursor - 1));
16
17   each(@properties, {
18     // actual printing
19     .foo((@cursor * 5), @key, @value)
20   })
21 }
22
23
24 // launch the loop
25 .loop(5);
  
```

1: Perulangan Sederhana dengan Rekursi

Bayangkan sohib pembaca membutuhkan beberapa kelas (*class*) yang berurut untuk menangani *margin*, sebagaimana *CSS* berikut:

```

3 .m-1 {
4   margin: 1px;
5 }
  
```

Hasil jadi dari naskah di atas dapat dicapai oleh [Less.js](#) dengan fungsi rekursi, lebih rumit dari *syntax* dari *SASS*, sebagaimana berikut di bawah:

```

.loop(@cursor) when (@cursor > 0) {
  .loop((@cursor - 1)); // next iteration
  ...
}
  
```

Penamaan *CSS Class* yang ingin dicapai adalah sebagai berikut:

```

.m-@{number}
  
```

Supaya nyaman maka perlu jelas, mari menentukan nilai awal perulangan.

```
{less} .loop(@cursor) when (@cursor > 0) {  
    // next iteration  
    .loop((@cursor - 1));  
  
    // code for each iteration  
    .m-@{cursor}{  
        margin: (1px * @cursor);  
    }  
}  
  
// launch the loop  
.loop(3);
```

Pahami dahulu, supaya tidak bingung nantinya.

Di dalam fungsi rekursi tersebut kita menjumpai interpolasi variabel yaitu `@{...}`, yang digunakan untuk meng-ekstrak suatu nilai ke dalam CSS.

► <http://lesscss.org/functions/#list-functions>

Dengan `@cursor` menangani isi dari perulangan, maka kita mendapatkan hasil sebagaimana berikut:



```
.m-1 {  
    margin: 1px;  
}  
  
.m-2 {  
    margin: 2px;  
}  
  
.m-3 {  
    margin: 3px;  
}
```

Yang menjadi keterbatasan dari `.loop(@cursor)` adalah kita tidak dapat menggunakan *interval*. Sebagai contoh adalah, kita dapat menghasilkan nilai berurut yaitu [1, 2, 3, 4, 5], namun kita tidak dapat menghasilkan nilai berurut berupa [5, 10, 15, 20, 25].

2: Trik Perulangan Lebih Lanjut

Untuk menyelesaikan permasalahan nilai berurut [5, 10, 15, 20, 25] maka kita dapat menggunakan fungsi yaitu `.foo(@number)`, yang dilakukan di dalam fungsi rekursi `.loop(@cursor)`. Pencetakan CSS yang sebenarnya dilakukan oleh `.foo(@number)`, sebagaimana di bawah ini:

```
{less} .loop(@cursor) when (@cursor > 0) {
    // next iteration
    .loop((@cursor - 1));

    // actual printing
    .foo((@cursor * 5))
}
```

Penamaan *CSS Class* yang ingin dicapai masih sama, yaitu sebagai berikut:

```
.m-@{number}
```

Maka sekarang kita susun, yaitu kode sumber dari perulangan dengan fungsi rekursi `.loop(@cursor)`-nya, tampak sebagaimana berikut di bawah:

```
{less} .foo(@number) {
    .m-@{number}{
        margin: (1px * @number);
    }
}

.loop(@cursor) when (@cursor > 0) {
    // next iteration
    .loop((@cursor - 1));

    // actual printing
    .foo((@cursor * 5))
}

// launch the loop
.loop(5);
```

Sehingga kita mendapatkan hasil naskah CSS seperti tampak di bawah:



```
.m-5 {  
    margin: 5px;  
}  
.m-10 {  
    margin: 10px;  
}  
.m-15 {  
    margin: 15px;  
}  
.m-20 {  
    margin: 20px;  
}  
.m-25 {  
    margin: 25px;  
}
```

3: Each

Sebelum melanjutkan ke situasi yang lebih rumit, kerjakan contoh yang sederhana terlebih dahulu. Tantangannya adalah persis seperti bab sebelumnya, yaitu membikin *margin property* yang memiliki beberapa jenis varian, sebagaimana berikut:

- ▶ *margin*,
- ▶ *margin-top*,
- ▶ *margin-bottom*,
- ▶ *margin-left*,
- ▶ *margin-right*.

Untuk kesederhanaan, maka penulis tidak mengikutsertakan varian *margin*.

Penamaan *CSS Class* yang ingin dicapai berbeda, yaitu sebagai berikut:

```
.m-@{side}-5
```

Variabel *Less.js* dapat menangani deklarasi *list*, yaitu:

```
@sides: top, bottom, left, right;
```


Sedangkan daftar list tersebut dapat di-akses dengan *iterator*, yaitu `@each`:

```
each(@sides, {
  ...@{value}...
});
```

Mari kita lihat bagaimana cara menangani variabel `@sides` ini:

```
{less} // property
@sides: top, bottom, left, right;

each(@sides, {
  .m-@{value}-5 {
    margin-@{value}: 5px;
  }
});
```

Walaupun tampaknya, perbedaan antara `Less.js` dengan `SASS`, hanya di *syntax* saja. Perhatikan `@{value}` sebagai hasil ekstraksi dari `@sides`.

Sekarang kita dapat melihat naskah hasil jadinya sebagaimana di bawah:



```
.m-top-5 {
  margin-top: 5px;
}
.m-bottom-5 {
  margin-bottom: 5px;
}
.m-left-5 {
  margin-left: 5px;
}
.m-right-5 {
  margin-right: 5px;
}
```

Ternyata naskah hasil jadinya masih tidak sesuai dengan harapan, karena yang kita inginkan adalah berbentuk `.m-t-5`, alih-alih berbentuk `.m-top-5`. Sama seperti bab sebelumnya.

4: Each untuk Pasangan Nilai

Less.js juga memiliki solusi untuk permasalahan ini.

Penamaan *CSS Class* yang ingin dicapai adalah menggunakan singkatan, yaitu sebagai berikut:

```
.m-@{abbreviation}-5.
```

Untungnya Less.js juga memperbolehkan kita menulis pasangan *pairs* dalam daftar *list*. Sehingga kita mendapatkan *sub-property* sebagaimana berikut:

```
@sides: {top: t; bottom: b; left: l; right: r}
```

Mari kita tulis ulang kode sebelumnya menjadi sebagai berikut:

```
{less} // property: abbreviation
@sides: {top: t; bottom: b; left: l; right: r}

each(@sides, {
  .m-@{value}-5 {
    margin-@{key}: 5px;
  }
})
```

Perhatikan hasil ekstraksi dari `@sides`, berupa `@{key}` dan `@{value}`. Dan jangan lupa kurung tutup fungsi yang bukan kurung kurawal.

Sekarang kita mendapatkan naskah hasil jadi yang kita inginkan.



```
.m-t-5 {
  margin-top: 5px;
}
.m-b-5 {
  margin-bottom: 5px;
}
.m-l-5 {
  margin-left: 5px;
}
.m-r-5 {
  margin-right: 5px;
}
```

Sebagaimana sebelumnya, penulis menggunakan `.m-t-5`, alih-alih `.mt-5`, untuk membedakan dengan spacing classes yang ada di salah satu *CSS Frameworks* yaitu *Bootstrap*.

5: Each di Perulangan

Pertimbangkan untuk meningkatkan kode untuk situasi yang lebih rumit. Kita membutuhkan varian dari kelas-kelas tadi untuk beberapa jarak angka yang berbeda secara berurut.

Penamaan *CSS Class* yang ingin dicapai adalah sebagai berikut:

```
.m-@{subname}-@{number}
```

Perhatikan kerangka kode-nya. Kita meletakkan *iterator*, yaitu `@each` ke dalam perulangan `@while`.

```
.foo(@number) {
  each(@sides, {
    ...
  })
}
```

Maka kode sumber *Less.js* selengkapnya adalah sebagai berikut:

```
{less} // sub-property: abbreviation
@sides: {top: t; bottom: b; left: l; right: r}

.foo(@number) {
  each(@sides, {
    .m-@{value}-@{number} {
      margin-@{key}: (1px * @number);
    }
  })
}
```

Perhatikan bahwa di sini kita sudah mendefinisikan fungsi `.foo(@number)`.

Karena panjang, kita lanjutkan di halaman berikutnya.

```
{less} .loop(@cursor) when (@cursor > 0) {  
    // next iteration  
    .loop((@cursor - 1));  
  
    // actual printing  
    .foo((@cursor * 5))  
}  
  
// launch the loop  
.loop(5);
```

Sekarang mulai dapat kita nikmati hasilnya secara perlahan yaitu beberapa kelas secara berurut:



```
.m-t-5 {  
    margin-top: 5px;  
}  
.m-b-5 {  
    margin-bottom: 5px;  
}  
.m-l-5 {  
    margin-left: 5px;  
}  
.m-r-5 {  
    margin-right: 5px;  
}  
.m-t-10 {  
    margin-top: 10px;  
}  
.m-b-10 {  
    margin-bottom: 10px;  
}
```



```
.m-l-10 {  
    margin-left: 10px;  
}  
.m-r-10 {  
    margin-right: 10px;  
}  
  
...  
  
...  
  
.m-t-25 {  
    margin-top: 25px;  
}  
.m-b-25 {  
    margin-bottom: 25px;  
}  
.m-l-25 {  
    margin-left: 25px;  
}  
.m-r-25 {  
    margin-right: 25px;  
}
```

Sengaja penulis potong di bagian tengah supaya tidak memenuhi isi buku.

6: Each Bersarang Di Dalam Perulangan

Mari kita tambah kesulitannya.

Penamaan *CSS Class* yang ingin dicapai adalah sebagai berikut:

```
.@{name}-@{subname}-@{number}
```

Kita tentukan dahulu deklarasi *list* untuk tiap-tiap *property* yang akan dihasilkan, yaitu:

- *Property*: *margin* dan *padding*.
- Masing-masing memiliki *sub-property*: *top*, *bottom*, *left*, *right*.

Tepatnya disajikan sebagai berikut:

```
@sides: {top: t; bottom: b; left: l; right: r}
@properties: {margin: m; padding: p}
```

Perhatikan kerangka kode-nya. Kita meletakkan dua *iterator* secara bersarang, yaitu *each* ke dalam perulangan *.loop(@cursor)*. Namun perlu diperhatikan kalau ada *outer loop* dan ada *inner loop*. Perulangan yang di dalam (*Inner loop*) berada di dalam *.foo(@number, @prop, @abbr)*.

```
.foo(@number, @prop, @abbr) {
  each(@sides, {
    ...
  })
}

.loop(@cursor) when (@cursor > 0) {
  ...

  each(@properties, {
    .foo((@cursor * 5), @key, @value)
  })
}
```

Selanjutnya seperti biasa, yaitu kode sumber `Less.js` selengkapnya sebagai berikut:

```
{less}    // sub-property: abbreviation
@sizes: {top: t; bottom: b; left: l; right: r}
@properties: {margin: m; padding: p}

.foo(@number, @prop, @abbr) {
  each(@sizes, {
    .@{abbr}-@{value}-@{number} {
      @{prop}-@{key}: (1px * @number);
    }
  })
}

.loop(@cursor) when (@cursor > 0) {
  // next iteration
  .loop((@cursor - 1));

  each(@properties, {
    // actual printing
    .foo((@cursor * 5), @key, @value)
  })
}

// launch the loop
.loop(5);
```

Hasilnya bukan hanya `margin`, namun juga `padding`.



```
.m-t-5 {  
    margin-top: 5px;  
}  
.m-b-5 {  
    margin-bottom: 5px;  
}  
.m-l-5 {  
    margin-left: 5px;  
}  
.m-r-5 {  
    margin-right: 5px;  
}  
.p-t-5 {  
    padding-top: 5px;  
}  
.p-b-5 {  
    padding-bottom: 5px;  
}  
.p-l-5 {  
    padding-left: 5px;  
}  
.p-r-5 {  
    padding-right: 5px;  
}  
.m-t-10 {  
    margin-top: 10px;  
}  
.m-b-10 {  
    margin-bottom: 10px;  
}  
...
```

Dan selanjutnya. Sengaja penulis potong lagi supaya tidak memenuhi isi buku.

Menambah Properti Utama

Menjelang *final*. Mari kita sempurnakan lagi, dengan `margin` dan `padding`, tanpa *sub-property*.

Penamaan *CSS Class* yang ingin dicapai adalah sebagai berikut:

```
.@{name}-@{number}
.@{name}-@{subname}-@{number}
```

Kita masukkan beberapa hal tambahan di dalam `.foo(@number, @prop, @abbr)`. Masih dengan meletakkan dua *iterator* secara bersarang, yaitu `each` baik *outer loop* dan ada *inner loop* ke dalam perulangan `.loop(@cursor)`. Kerangkanya sedikit berubah.

```
{less} .foo(@number, @prop, @abbr) {
    .@{abbr}-@{number} {
        @{prop}: (1px * @number) !important;
    }

    each(@sides, {
        .@{abbr}-@{value}-@{number} {
            @{prop}-@{key}: (1px * @number) !important;
        }
    })
}
```

Selanjutnya seperti biasa, yaitu kode sumber `Less.js` selengkapnya.

```
{less} // sub-property: abbreviation
@properties: {margin: m; padding: p}
@sides: {top: t; bottom: b; left: l; right: r}

.foo(@number, @prop, @abbr) {
  .@{abbr}-@{number} {
    @{prop}: (1px * @number) !important;
  }

  each(@sides, {
    .@{abbr}-@{value}-@{number} {
      @{prop}-@{key}: (1px * @number) !important;
    }
  })
}

.loop(@cursor) when (@cursor > 0) {
  // next iteration
  .loop((@cursor - 1));

  each(@properties, {
    // actual printing
    .foo((@cursor * 5), @key, @value)
  })
}

// launch the loop
.loop(5);
```

Perhatikan, bahwa kita juga menerapkan aturan *rule*, yaitu `!important` ke dalam value dari CSS di masing-masing kelas. Sehingga hasilnya sebagai berikut:



```
.m-5 {
  margin: 5px !important;
}

.m-t-5 {
  margin-top: 5px !important;
}

...

.p-5 {
  padding: 5px !important;
}

...
```

Dan seterusnya. Kita hampir dapat menggunakan *custom spacing class* ini secara lengkap untuk digunakan di suatu website ataupun proyek pribadi.

Penerapan di Dunia Nyata

Untuk alasan praktis, ada beberapa hal yang perlu diperhatikan.

- ▶ Memulai dari angka 0 (nol), alih-alih dari 5 (lima), supaya tersedia pengaturan tanpa *margin* dan tanpa *padding*.
- ▶ Menggunakan X, dan Y, supaya mirip *Bootstrap*. Hal ini memudahkan saat migrasi dari *bootstrap* ke *custom class*.

Hasil naskah jadi, yang ingin kita dapatkan adalah sebagai berikut.



```
.m-y-5 {
  margin-top: 5px !important;
  margin-bottom: 5px !important;
}

.m-x-5 {
  margin-left: 5px !important;
  margin-right: 5px !important;
}
```

Semakin lengkap belum tentu semakin baik. Kekurangan dari cara ini adalah, ukuran *stylesheet* menjadi gemuk, karena ukurannya besar.

Seperti telah disebutkan sebelumnya, `Less.js` memperbolehkan pasangan nilai di dalam deklarasi *list*, sehingga sekarang kita memiliki *sub-property* sebagaimana berikut:

```
@properties: {margin: m; padding: p}
@sides: {top: t; bottom: b; left: l; right: r}
@sidesy: top, bottom;
@sidesx: left, right;
```

Kemudian mari kita lihat kerangkanya, perulangan yang dibikin menjadi juga membutuhkan perubahan. Demikian juga fungsi `foo`, ada sedikit perubahan.

```
{less} .foo(@number, @prop, @abbr) {
  .@{abbr}-@{number} {
    @{prop}: (1px * @number) !important;
  }

  each(@sides, {
    .@{abbr}-@{value}-@{number} {
      @{prop}-@{key}: (1px * @number) !important;
    }
  })

  .@{abbr}-y-@{number} {
    each(@sidesy, {
      @{prop}-@{value}: (1px * @number) !important;
    })
  }

  .@{abbr}-x-@{number} {
    each(@sidesx, {
      @{prop}-@{value}: (1px * @number) !important;
    })
  }
}
```

Berikut kode sumber `Less.js` selengkapnya, untuk keperluan sehari-hari.

```
{less}    // sub-property: abbreviation
@properties: {margin: m; padding: p}
@sides: {top: t; bottom: b; left: l; right: r}
@sidesy: top, bottom;
@sidesx: left, right;

.foo(@number, @prop, @abbr) {
  .@{abbr}-@{number} {
    @{prop}: (1px * @number) !important;
  }

  each(@sides, {
    .@{abbr}-@{value}-@{number} {
      @{prop}-@{key}: (1px * @number) !important;
    }
  })

  .@{abbr}-y-@{number} {
    each(@sidesy, {
      @{prop}-@{value}: (1px * @number) !important;
    })
  }

  .@{abbr}-x-@{number} {
    each(@sidesx, {
      @{prop}-@{value}: (1px * @number) !important;
    })
  }
}
```

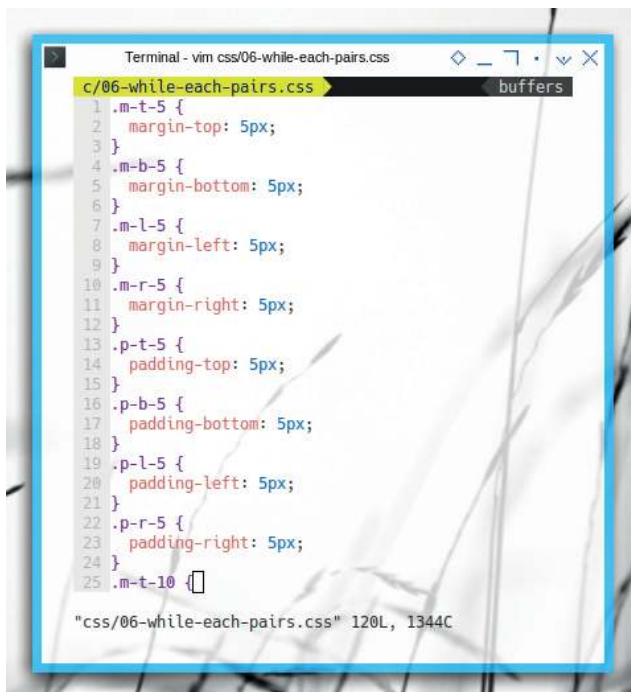
Kita lanjutkan di halaman berikutnya, karena kode-nya panjang.

```
{less}
.loop(@cursor) when (@cursor > 0) {
  // next iteration
  .loop((@cursor - 1));

  each(@properties, {
    // actual printing
    .foo((@cursor * 5 - 5), @key, @value)
  })
}

// launch the loop
.loop(6);
```

Akhirnya *custom spacing classes* ini siap dipakai. Mengenai hasilnya bagaimana, dapat dicoba sendiri sebagai latihan. Setelah penjelasan mengenai perulangan (*loop*) ini berakhir, kita akan melanjutkan ke penjelasan mengenai *conditional*.

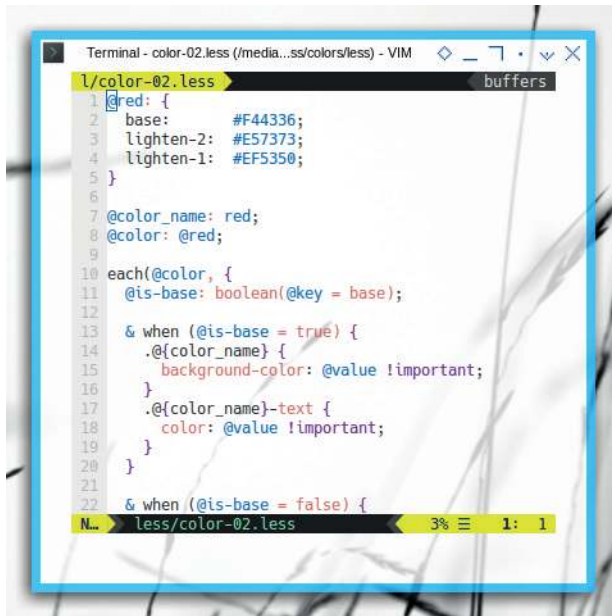


```
Terminal - vim css/06-while-each-pairs.css
c/06-while-each-pairs.css buffers
1 .m-t-5 {
2   margin-top: 5px;
3 }
4 .m-b-5 {
5   margin-bottom: 5px;
6 }
7 .m-l-5 {
8   margin-left: 5px;
9 }
10 .m-r-5 {
11   margin-right: 5px;
12 }
13 .p-t-5 {
14   padding-top: 5px;
15 }
16 .p-b-5 {
17   padding-bottom: 5px;
18 }
19 .p-l-5 {
20   padding-left: 5px;
21 }
22 .p-r-5 {
23   padding-right: 5px;
24 }
25 .m-t-10 {}

"css/06-while-each-pairs.css" 120L, 1344C
```

Color Class dengan LESS

- Tujuan: Menghasilkan color class dengan guard di `Less.js`.



```

Terminal - color-02.less (/media...ss/colors/less) - VIM
l/color-02.less
1 @red: {
2   base:          #F44336;
3   lighten-2:     #E57373;
4   lighten-1:     #EF5350;
5 }
6
7 @color_name: red;
8 @color: @red;
9
10 each(@color, {
11   @is-base: boolean(@key = base);
12
13   & when (@is-base = true) {
14     .@{color_name} {
15       background-color: @value !important;
16     }
17     .@{color_name}-text {
18       color: @value !important;
19     }
20   }
21
22   & when (@is-base = false) {
23     less/color-02.less
24   }
25 }
  
```

Di `less.js`, *conditional* berupa *if-then-else*, dapat dilakukan dengan memakai *guard*.

Contoh Kasus

- GMC: dari `Sass` ke `Less`.

Jauh sebelum penulis pernah *CSS Frameworks* bernama `Materialize CSS`, penulis sudah memakai tata warna palet dari `Google Material Design`. Biasanya penulis menyingkat dengan nama `GMC` (*google material color*).

Selanjutnya penulis menemukan `SASS` untuk mengolah warna palet yang bagus ini ke hasil jadi naskah `CSS` di dalam `Materialize CSS`. Namun penulis tidak membahas `SASS` tersebut karena bukan hasil karya penulis.

Bagian ini akan membahas saduran langsung (*direct port*) dari `SASS` untuk *google material color* tersebut ke `Less.js`.

1: Warna Dasar

Mari kita mulai dengan contoh kasus yang sangat dasar. Yaitu tiga warna [hitam, putih, transparan] yang akan kita olah menjadi warna *foreground* (text) dan warna *background*. Prosesnya sangat sederhana karena hanya membutuhkan *each*. Mari kita lihat bentuk kode *SASS* berikut:

Sass

```
$shades: (
  "black":      #000000,
  "white":      #FFFFFF,
  "transparent": transparent
);

// Shade classes
@each $color, $color_value in $shades {
  .#{$color} {
    background-color: $color_value !important;
  }
  .#{$color}-text {
    color: $color_value !important;
  }
}
```

Berikutnya dapat kita lihat saduran kode *Less.js*, yang juga sederhana.

{less}

```
@shades: {
  black:      #000000;
  white:      #FFFFFF;
  transparent: transparent
}

each(@shades, {
  .@{key}{
    background-color: @value !important;
  }
  .@{key}-text {
    color: @value !important;
  }
})
```


Sebelum melihat hasilnya kita pahami dahulu, yaitu bagaimana **Materialize CSS** melakukan pengaturan pada *stylesheet*-nya. **Materialize CSS** menggunakan dua *selectors*, yaitu:

- *Background*, menggunakan **color** sebagai selector, dan
- *Foreground*, menggunakan **color-text** sebagai selector,.

Dari sini kita mendapatkan hasil sebagaimana berikut:



```
.black {
  background-color: #000000 !important;
}
.black-text {
  color: #000000 !important;
}
.white {
  background-color: #FFFFFF !important;
}
.white-text {
  color: #FFFFFF !important;
}
.transparent {
  background-color: transparent !important;
}
.transparent-text {
  color: transparent !important;
}
```

Maka dari tiga warna tadi, kita mendapatkan enam *CSS rules*.

2: Sederhana: Satu Warna

Sekarang gunakan warna sungguhan, misalnya merah. Proses-nya sangat mirip dengan skrip sebelumnya.

Penulis menyederhanakan [SASS](#) yang rumit dari *materialize CSS* menjadi berikut di bawah:

Sass

```
$red: (  
  "base":      #F44336,  
  "lighten-2": #E57373,  
  "lighten-1": #EF5350,  
);  
  
$color_name: "red";  
$color: $red;  
  
@each $color_type, $color_value in $color {  
  .#{$color_name}.#{$color_type} {  
    background-color: $color_value !important;  
  }  
  .#{$color_name}-text.text-#{$color_type} {  
    color: $color_value !important;  
  }  
}
```

Berikut saduran kode [Less.js](#)-nya.

{less}

```
@red: {  
  base:      #F44336;  
  lighten-2: #E57373;  
  lighten-1: #EF5350;  
}  
  
@color_name: red;  
@color: @red;
```

Kita lanjutkan di halaman berikutnya:

```
{less}
  each(@color, {
    .@{color_name}.@{key} {
      background-color: @value !important;
    }
    .@{color_name}-text.text-@{key} {
      color: @value !important;
    }
  })
```

Maka dari tiga warna tadi, kita juga mendapatkan enam *CSS rules*.



```
.red.base {
  background-color: #F44336 !important;
}
.red-text.text-base {
  color: #F44336 !important;
}
.red.lighten-2 {
  background-color: #E57373 !important;
}
.red-text.text-lighten-2 {
  color: #E57373 !important;
}
.red.lighten-1 {
  background-color: #EF5350 !important;
}
.red-text.text-lighten-1 {
  color: #EF5350 !important;
}
```

3: Kondisional: Satu Warna

Sekarang saatnya memasuki bagian yang susah, yaitu

- ▶ Kita akan mencetak hasil berupa `.red {...}`
- ▶ Dan bukan mencetak hasil berupa `.red.base {...}`
- ▶ Masalah: `Less.JS` tidak memiliki perangkat *if-then-else*.

Mari kita lihat terlebih dahulu kode dari `SASS`, supaya sohib pembaca paham apa maksudnya.

Sass

```
$red: (
  "base":          #F44336,
  "lighten-2":     #E57373,
  "lighten-1":     #EF5350,
);

$color_name: "red";
$color: $red;

@each $color_type, $color_value in $color {
  @if $color_type == "base" {
    .#{$color_name} {
      background-color: $color_value !important;
    }
    .#{$color_name}-text {
      color: $color_value !important;
    }
  }
  @else {
    .#{$color_name}.#{$color_type} {
      background-color: $color_value !important;
    }
    .#{$color_name}-text.text-#{$color_type} {
      color: $color_value !important;
    }
  }
}
```

Untungnya, `Less.js` memiliki *guard*.

```
{less}
  each(@color, {
    @is-base: boolean(@key = base);

    & when (@is-base = true) {
      ...
    }

    & when (@is-base = false) {
      ...
    }
  })
```

Perhatikan bahwa, terlebih dahulu harus ditentukan apa yang benar.

► `@is-base: boolean(@key = base)`

`Less.js` membutuhkan langkah, lebih banyak dari `SASS`. Namun setidaknya dapat berjalan dengan baik.

Sekarang mari kita lihat saduran kode ke dalam bentuk `Less.js` lengkap dengan `guard`.

```
{less} @red: {  
    base:      #F44336;  
    lighten-2: #E57373;  
    lighten-1: #EF5350;  
}  
  
@color_name: red;  
@color: @red;  
  
each(@color, {  
    @is-base: boolean(@key = base);  
  
    & when (@is-base = true) {  
        .@{color_name} {  
            background-color: @value !important;  
        }  
        .@{color_name}-text {  
            color: @value !important;  
        }  
    }  
  
    & when (@is-base = false) {  
        .@{color_name}.@{key} {  
            background-color: @value !important;  
        }  
        .@{color_name}-text.text-@{key} {  
            color: @value !important;  
        }  
    }  
})
```

Sekarang mulai dapat kita nikmati hasilnya secara perlahan yaitu menggunakan `.red {...}`, alih-alih `.red.base {...}`.



```
.red {  
    background-color: #F44336 !important;  
}  
.red-text {  
    color: #F44336 !important;  
}  
.red.lighten-2 {  
    background-color: #E57373 !important;  
}  
.red-text.text-lighten-2 {  
    color: #E57373 !important;  
}  
.red.lighten-1 {  
    background-color: #EF5350 !important;  
}  
.red-text.text-lighten-1 {  
    color: #EF5350 !important;  
}
```

Tujuan terpenuhi, mari kita lanjutkan untuk menuntaskan secara lengkap.

4: Banyak Warna

Each bersarang dapat menjadi masalah di [Less.js](#), karena variabel yang digunakan memakai nama sama yaitu `@key` dan `@value`. Namun ini dapat diselesaikan dengan memisahkan bagian dalam dari fungsi.

Sekarang mari kita lihat dahulu, bagian kode [SCSS](#) dari [Materialize CSS](#). Kita ambil warna seperlunya saja.

Scss

```
$red: (  
  "base":      #F44336,  
  "lighten-2": #E57373,  
  "lighten-1": #EF5350,  
);  
  
$blue: (  
  "base":      #2196F3,  
  "lighten-2": #64B5F6,  
  "lighten-1": #42A5F5,  
);  
  
$colors: (  
  "red": $red,  
  "blue": $blue  
);
```


Berikut kode asli SCSS dari [Materialize CSS](#). Yang untuk selengkapnya dapat dilihat dari repository dari berkas aslinya.



```
@each $color_name, $color in $colors {
  @each $color_type, $color_value in $color {
    @if $color_type == "base" {
      .#{$color_name} {
        background-color: $color_value !important;
      }
      .#{$color_name}-text {
        color: $color_value !important;
      }
    }
    @else if $color_name != "shades" {
      .#{$color_name}.#{$color_type} {
        background-color: $color_value !important;
      }
      .#{$color_name}-text.text-#{$color_type} {
        color: $color_value !important;
      }
    }
  }
}
```

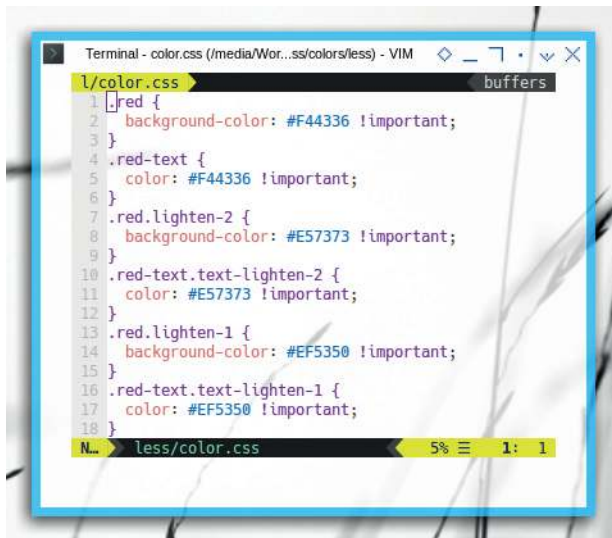
Sebagai gambaran, kerangka berkas [Less.js](#)-nya adalah sebagai berikut:

```
{less} .foo_color(@color_name, @color) {
  each(@color, {
    ...
  })
}

each(@colors, {
  .foo_color(@key, @value)
})
```

Untuk pengaturan warna di `Less.js`, tidak terlalu jauh berbeda.

```
{less}  
  
@red: {  
    base:      #F44336;  
    lighten-2: #E57373;  
    lighten-1: #EF5350;  
}  
  
@blue: {  
    base:      #2196F3;  
    lighten-2: #64B5F6;  
    lighten-1: #42A5F5;  
}  
  
@colors: {  
    red: @red;  
    blue: @blue;  
}
```



Namun dalam kenyataannya, kode `Less.js`-nya sedikit lebih panjang.

```
{less}
    .foo_color(@color_name, @color) {
        each(@color, {
            @is-base: boolean(@key = base);

            & when (@is-base = true) {
                .@{color_name} {
                    background-color: @value !important;
                }
                .@{color_name}-text {
                    color: @value !important;
                }
            }

            & when (@is-base = false) {
                .@{color_name}.@{key} {
                    background-color: @value !important;
                }
                .@{color_name}-text.text-@{key} {
                    color: @value !important;
                }
            }
        })
    }

    each(@colors, {
        .foo_color(@key, @value)
    })
}
```

Hasil akhirnya adalah sebagai berikut:



```
.red {  
  background-color: #F44336 !important;  
}  
.red-text {  
  color: #F44336 !important;  
}  
.red.lighten-2 {  
  background-color: #E57373 !important;  
}  
.red-text.text-lighten-2 {  
  color: #E57373 !important;  
}  
.red.lighten-1 {  
  background-color: #EF5350 !important;  
}  
.red-text.text-lighten-1 {  
  color: #EF5350 !important;  
}  
.blue {  
  background-color: #2196F3 !important;  
}  
.blue-text {  
  color: #2196F3 !important;  
}  
.blue.lighten-2 {  
  background-color: #64B5F6 !important;  
}  
.blue-text.text-lighten-2 {  
  color: #64B5F6 !important;  
}  
.blue.lighten-1 {  
  background-color: #42A5F5 !important;  
}  
.blue-text.text-lighten-1 {  
  color: #42A5F5 !important;  
}
```

Bagaimana menurut sohib pembaca?

Bab Keempat

Menggunakan PostCSS

Pendahuluan

`PostCSS` ini memiliki banyak fungsi dan dapat diprogram sesuai kebutuhan. Karena itu menjadi sangat luwes sehingga menjadi pilihan banyak pengembang *web*. `PostCSS` ini adalah salah satu *tools* yang perlu dikenal oleh *web developer*.

Salah satu kemampuannya adalah sebagai *CSS Preprocessor*, dengan bantuan `PreCSS`. Mengapa membutuhkan peralatan tambahan yaitu `PreCSS`? Karena `PostCSS` bukanlah *CSS Preprocessor*.

Seperti biasa, kita akan membahas dengan contoh berupa perulangan *looping* di *CSS Preprocessor*, yang kali ini menggunakan `PostCSS`.

Penyusunan

Bab ini dibagi tiga bagian, yaitu:

- ▶ Konfigurasi `PostCSS`, yaitu persiapan untuk menghasilkan *spacing classes*.
- ▶ Perulangan (*Looping*), yaitu proses *coding* menghasilkan *spacing classes*.
- ▶ `SugarSS`, yaitu *format* awal *coding*-nya berbentuk indentasi, sebagai alternatif pengganti dari `PreCSS`.

*Langkah demi langkah, unjuk kemampuan dari `PostCSS`,
untuk menghasilkan margin class maupun padding class.*

Konfigurasi PostCSS

Ada banyak cara untuk melakukan konfigurasi dengan PostCSS, misalnya:

- ▶ Menggunakan `postcss.config.js`.
- ▶ Menggunakan *task runner*, misalnya dengan `Grunt` ataupun `Gulp`.
- ▶ Menggunakan *bundler*, misalnya `postcss.config.js` bersama `webpack.config.js`.

1: Konfigurasi PostCSS secara langsung

Cara ini yang paling mudah karena dapat langsung dijalankan di *terminal*. Cara ini juga tepat guna, dengan asumsi sohib pembaca tidak perlu bekerja dengan peralatan lain misalnya `grunt`, `gulp` atau `webpack`.

Kebutuhan *requirement*, sangat beragam untuk tiap kasus. Ada pengembang yang menggunakan `PreCSS`, dan pengembang lain mungkin lebih memilih `SugarSS`. Demikian pula, rincian fitur tiap proyek dapat saja berbeda, sehingga tidak mesti sama dengan contoh yang diberikan di sini.

Untuk contoh *spacing class* berikut, `PreCSS` adalah yang kita butuhkan. Lalu untuk tiga alasan bagus lain, akan kita tambahkan beberapa fitur:

- ▶ `prettier` atau `prettify`: untuk merapikan hasil keluaran `CSS`, supaya tampil *menarik* di buku.
- ▶ `postcss-strip-inline-comments`: supaya bisa ditambahkan baris komentar sebagaimana `SASS`, dan
- ▶ `postcss-each`: untuk mengatasi perintah `each` yang rumit, dengan variabel yang lebih dari satu.

Sebagai contoh di sini, maka kita pakai konfigurasi `postcss.config.js` sebagai berikut:

```
module.exports = {
  syntax: 'postcss-scss',
  plugins: [
    require('postcss-strip-inline-comments'),
    require('postcss-each'),
    require('precss'),
    require('postcss-prettify'),
  ],
}
```

Supaya aplikasi `NodeJS` ini dapat dijalankan, maka kita membutuhkan `package.json`, sebagaimana berikut di bawah ini:

```
{
  ...
  "devDependencies": {
    "postcss": "^7.0.29",
    "postcss-cli": "^7.1.1",
    "postcss-each": "^0.10.0",
    "postcss-prettify": "^0.3.4",
    "postcss-scss": "^2.0.0",
    "postcss-strip-inline-comments": "^0.1.5",
    "precss": "^4.0.0"
  }
}
```

Jangan lupa memasang `NPM`.

```
$ npm install
```

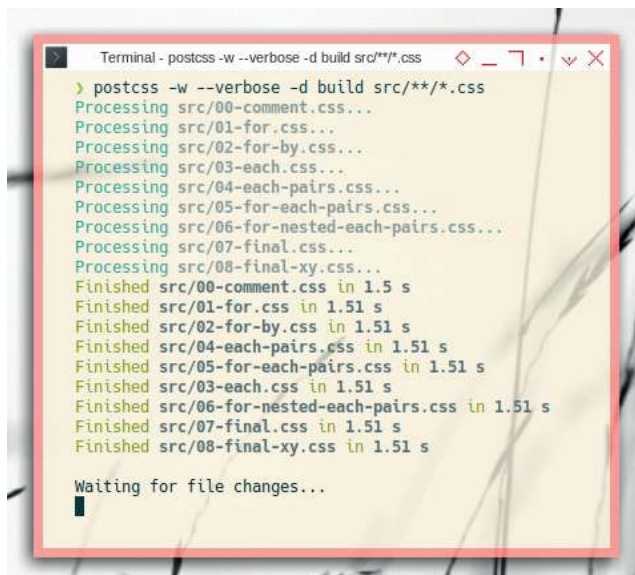
Sekarang kita dapat menjalankan perintah ini, untuk mengkompilasi berkas satu per satu:

```
$ postcss -d build src/00-comment.css
```


Atau mengamati (*watch*) secara lengkap (*verbose*), keseluruhan *folder*.

```
$ postcss -w --verbose -d build src/**/*.css
Processing src/00-comment.css...
Processing src/01-for.css...
Processing src/02-for-by.css...
Processing src/03-each.css...
Processing src/04-each-pairs.css...
Processing src/05-for-each-pairs.css...
Processing src/06-for-nested-each-pairs.css...
Processing src/07-final.css...
Processing src/08-final-xy.css...
Finished src/00-comment.css in 1.5 s
Finished src/01-for.css in 1.51 s
Finished src/02-for-by.css in 1.51 s
Finished src/04-each-pairs.css in 1.51 s
Finished src/05-for-each-pairs.css in 1.51 s
Finished src/03-each.css in 1.51 s
Finished src/06-for-nested-each-pairs.css in 1.51 s
Finished src/07-final.css in 1.51 s
Finished src/08-final-xy.css in 1.51 s

Waiting for file changes...
```



```
Terminal - postcss -w --verbose -d build src/**/*.css

> postcss -w --verbose -d build src/**/*.css
Processing src/00-comment.css...
Processing src/01-for.css...
Processing src/02-for-by.css...
Processing src/03-each.css...
Processing src/04-each-pairs.css...
Processing src/05-for-each-pairs.css...
Processing src/06-for-nested-each-pairs.css...
Processing src/07-final.css...
Processing src/08-final-xy.css...
Finished src/00-comment.css in 1.5 s
Finished src/01-for.css in 1.51 s
Finished src/02-for-by.css in 1.51 s
Finished src/04-each-pairs.css in 1.51 s
Finished src/05-for-each-pairs.css in 1.51 s
Finished src/03-each.css in 1.51 s
Finished src/06-for-nested-each-pairs.css in 1.51 s
Finished src/07-final.css in 1.51 s
Finished src/08-final-xy.css in 1.51 s

Waiting for file changes...
```

2: Alternatif: Gulp

Ada beberapa cara untuk melakukan konfigurasi [PostCSS](#), salah satunya adalah dengan memakai [Gulp](#). [Gulp](#) ini peralatan lawas yang sebetulnya sudah tidak dipakai lagi, tergantikan dengan peralatan-peralatan yang lebih modern. Namun [Gulp](#) tetap dibahas di sini, karena cukup sederhana, sementara peralatan-peralatan yang lebih modern cukup rumit untuk dijelaskan kepada pemula.

Konfigurasi dasar [PostCSS](#) dengan menggunakan [PreCSS](#) di [Gulp](#) adalah sebagai berikut.

```
'use strict';

var gulp = require('gulp');

gulp.task('css', () => {
  const postcss = require('gulp-postcss')

  return gulp.src('src/**/*.css')
    .pipe( postcss([ require('precss') ]) )
    .pipe( gulp.dest('dest/') )
})

gulp.task('default', gulp.series('css'));
```

Konfigurasi [Gulp](#) selengkapnya, adalah sebagai berikut:

```
'use strict';

var gulp = require('gulp');
```

Karena panjang kita lanjutkan di halaman selanjutnya.

```

gulp.task('css', () => {
  const postcss = require('gulp-postcss')
  const prettier = require('gulp-prettier')
  const scss = require('postcss-scss');

  return gulp.src('src/**/*.css')
    .pipe( postcss([
      require('postcss-strip-inline-comments'),
      require('postcss-each'),
      require('precss')
    ], {syntax: scss})
    )
    .pipe( prettier() )
    .pipe( gulp.dest('dest/') )
  })

  gulp.task('default', gulp.series('css'));

```

Supaya aplikasi **NodeJS** ini dapat dijalankan, lengkap dengan tiga fitur tambahan, maka kita membutuhkan **package.json** berikut di bawah ini:

```

{
  ...
  "devDependencies": {
    "gulp": "^4.0.2",
    "gulp-postcss": "^8.0.0",
    "gulp-prettier": "^2.3.0",
    "precss": "^4.0.0",
    "postcss-scss": "^2.0.0",
    "postcss-strip-inline-comments": "^0.1.5",
    "postcss-each": "^0.10.0"
  }
}

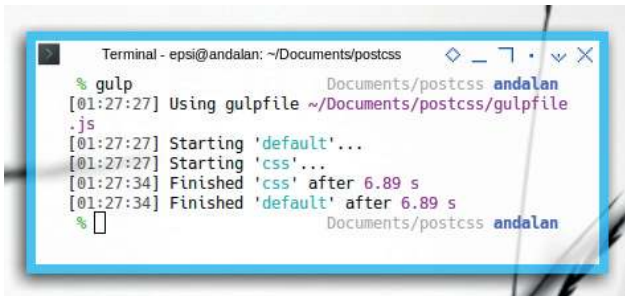
```

Sekarang kita dapat memasang **NPM**.

```
$ npm install
```

Berikutnya adalah, menjalankan **Gulp**.

```
$ gulp
[01:27:27] Using gulpfile ~/Documents/postcss/gulpfile.js
[01:27:27] Starting 'default'...
[01:27:27] Starting 'css'...
[01:27:34] Finished 'css' after 6.89 s
[01:27:34] Finished 'default' after 6.89 s
```



3: Alternatif: Grunt

Walaupun merupakan peralatan lawas, namun **Grunt** masih cocok dipakai untuk kebutuhan sehari-hari, sambil menunggu waktunya sampai jarang dipakai lagi sebagaimana **Gulp**. Konfigurasi **Grunt** cukup panjang, namun jelas alurnya.

Menggunakan **Grunt** membutuhkan sedikit kejelian, karena **PostCSS** di **Grunt** akan menimpa berkas aslinya. Untungnya ada jawaban yang tepat, dan juga lengkap, di [stackoverflow](https://stackoverflow.com/questions/36376645/how-to-use-autoprefixer-with-postcss-and-grunt-on-more-than-one-file#36442427). Yaitu dengan menambahkan **grunt-contrib-copy**.

- <https://stackoverflow.com/questions/36376645/how-to-use-autoprefixer-with-postcss-and-grunt-on-more-than-one-file#36442427>

Percobaan pertama penulis, dari konfigurasi dasar [PostCSS](#) dengan menggunakan [PreCSS](#) di [Grunt](#) adalah sebagai berikut.

```
module.exports = function(grunt) {
  // configure the tasks
  let config = {

    postcss: {
      options: {
        map: true, // inline sourcemaps
        syntax: require('postcss-scss'),
        processors: [
          require('postcss-strip-inline-comments'),
          require('postcss-each'),
          require('precss'),
          require('postcss-prettify'),
        ]
      },
      dist: {
        src: 'src/*.css'
      }
    }
  };

  grunt.initConfig(config);

  // load the tasks
  grunt.loadNpmTasks('grunt-postcss');

  // define the tasks
  grunt.registerTask('default', [
    'postcss'
  ] );

};
```

Yang tentu saja, konfigurasi ini akan menimpa semua berkas asli yang terletak di `src/*.css`.

Konfigurasi [Grunt](#) selengkapnya setelah membaca [stackoverflow](#) di atas, adalah sebagai berikut:

```
module.exports = function(grunt) {  
  // configure the tasks  
  let config = {  
  
    postcss: {  
      options: {  
        map: true, // inline sourcemaps  
        syntax: require('postcss-scss'),  
        processors: [  
          require('postcss-strip-inline-comments'),  
          require('postcss-each'),  
          require('precss'),  
          require('postcss-prettify'),  
        ]  
      },  
      dist: {  
        files: [  
          {  
            src: 'dest/*.css'  
          }  
        ]  
      }  
    },  
  },  
};
```

Karena panjang kita lanjutkan di halaman selanjutnya. Di sini kita menggunakan perintah `copy`.

```
copy: {
  postcss: {
    files: [
      {
        src: 'src/*.css',
        dest: './dest/',
        expand: true,
        flatten: true
      }
    ]
  }
},

watch: {
  postcss: {
    files: ['src/*'],
    tasks: ['postcss'],
    options: {
      interrupt: false,
      spawn: false
    }
  },
}

};
```

Sekali lagi, karena panjang kita lanjutkan di halaman selanjutnya.

```
grunt.initConfig(config);

// load the tasks
grunt.loadNpmTasks('grunt-postcss');
grunt.loadNpmTasks('grunt-contrib-copy');
grunt.loadNpmTasks('grunt-contrib-watch');

grunt.registerTask('css', ['copy:postcss', 'postcss']);

// define the tasks
grunt.registerTask('default', [
  'css', 'watch'
] );

};
```

Supaya aplikasi [NodeJS](#) ini dapat dijalankan, lengkap dengan semua fitur selengkapannya, maka kita membutuhkan [package.json](#) berikut di bawah ini:

```
{
  ...
  "devDependencies": {
    "grunt": "^1.0.1",
    "grunt-contrib-copy": "^1.0.0",
    "grunt-contrib-watch": "^1.1.0",
    "grunt-postcss": "^0.9.0",
    "postcss-each": "^0.10.0",
    "postcss-prettify": "^0.3.4",
    "postcss-scss": "^2.0.0",
    "postcss-strip-inline-comments": "^0.1.5",
    "precss": "^4.0.0"
  }
}
```


Sekarang kita dapat memasang NPM.

```
$ npm install
```

Berikutnya adalah, menjalankan Grunt.

```
$ grunt
```

```
Running "copy:postcss" (copy) task
```

```
Copied 9 files
```

```
Running "postcss:dist" (postcss) task
```

```
>> 9 processed stylesheets created.
```

```
Running "watch" task
```

```
Waiting...
```

