SCSS

15分鐘入門

SCSS(Sassy CSS，時髦的CSS)是SASS中的一種新式語法，SASS則是一種針對CSS的腳本程式語言，藉由提供程式語言的特性，例如變數、巢狀結構、混合、函式與擴充(繼承)等等，可以預先對CSS進行結構化的工作，最後再編譯的網頁上可使用的CSS語法。

SCSS使用和CSS一樣的區塊({})與定義結尾(;)語法，它是基於CSS語法的超集(superset)，如果你已經有一些CSS語法的基礎，學習SCSS會很容易，雖然它有一些類似程式語言的語法，但都相當簡單易用，並沒有太複雜的語法。為了與之前的SASS語法作區分，檔案的副檔名使用.scss。(舊語法則使用.sass)

SCSS本身需要使用命令列工具進行編譯，這些工具可以與現有的程式開發或網頁編輯程式作搭配使用，支援很多也很廣泛，也有網頁版的線上編輯工具如SassMeister或CodePen，或是專用圖形化介面應用程式。要在自己的電腦上使用SCSS通常需要使用兩種工具，一個是用於編譯SCSS檔案為CSS檔案的"**編譯工具**"，另一個則是對SCSS程式碼的"**語法檢查工具**"。

> 以下的SCSS範例，你可以先使用SassMeister來複製貼上看看編譯的成果，後面的內容有附工具安裝的說明。

變數與運算

可以用錢號($)來定義變數，用冒號(:)來指定變數的值，這與CSS定義屬性值很相似。在檔案中的其他地方就使用這些變數值。

變數的命名通常使用在CSS中常見的屬性名稱寫法，也就是用連接號(-)與全小寫英文，例如 `$second-color` ，變數在編譯後並不會出現在最後輸出的CSS檔案中。

此外，變數也可以指定給其他變數使用，變數的值也可以是一個字串值，不過解譯時有特別的語法。

```
//scss
$color-white: #fff;
$color-pink: #ee11ab;
$title-font: normal 24px/1.5 'Open Sans', sans-serif;

$primary-color: $color-pink;

a {
  background-color: $color-white;
  color: $color-pink;
}
```

> 撰寫時的建議: 對於會套用到整個網站的每個顏色、字體建議都先定義成變數。

> 撰寫時的建議: 不要使用像 `red` 、 `blue` 的英文字詞來指定顏色，而是要用16進位碼來定義，例如 `red` 應該寫成 `#ff0000` 。

> 撰寫時的建議: 在CSS類別中定義時，按照英文字元A-Za-z來排列其中的CSS樣式定義。

變數可以再進行作加減乘除餘(`+-*/%`)運算，最特別的是字串與顏色色碼也可以進行運算。

```
//scss
.container {
  width: 100%;
}

article[role="main"] {
  float: left;
  width: 600px / 960px * 100%;
}

aside[role="complementary"] {
  float: right;
  width: 300px / 960px * 100%;
}
```

> 撰寫時的建議: 進行運算的運算符號前後要加上空格，不要黏在一團。

巢狀(Nesting)

巢狀語法可以使用明確的階層定義，這種結構在CSS中很常見，例如像下面的選單或清單項目的風格定義:

```
// scss
ul {
  list-style: none;

  li {
    display: inline-block;
    padding: 15px;

    a {
      color: #444;
      font-size: 16px;
      text-decoration: none;
    }
  }
}
```

上面的內容會編譯為下面的CSS內容:

```
/* css */
ul {
  list-style: none;
}

ul li {
  display: inline-block;
  padding: 15px;
}

ul li a {
  color: #444;
  font-size: 16px;
  text-decoration: none;
}
```

因為CSS中的巢狀結構會有兩種情況，一種是指包含在某個DOM元素之內的，例如上面範例的 `ul li a` 。而另一種則是指同個DOM元素的但不同類別，例如 `ul.mylist` 或 `a:hover` (偽類)。有空格就代表是第一種巢狀的結構，也就是上面的原本的巢狀定義方式。

第二種巢狀結構是沒有空格，接著在一起的CSS定義，這時在巢狀結構語法裡，就需要加入與符號(&)來定義，代表是黏在一起中間沒有空格的，例如以下的範例:

```
// scss
.parent{
  &:hover{}
  &.other-class{}
}
```

上面的內容會編譯為下面的CSS內容:

```
/* CSS */
.parent:hover{}
.parent.other-class{}
```

> 風格建議: 如果你有像 `h1.title` 或 `li.mylist` 之類的CSS宣告，不論裡面的樣式有多簡單，都應該在SCSS檔案中都要列成巢狀結構。

匯入(Import)

匯入其他的SCSS檔案，最後編譯時會一併包含進來編譯。要被匯入的通常檔案名稱前會加下底線(_)作區分，這樣編譯工具在編譯時會略過這些檔案，而只會去編譯那些沒下底線(_)的檔案，例如 `_reset.scss` 。

例如 base.scss 中要匯入 _reset.scss ， base.scss 的檔案內容會如下，要注意的是只需要寫 @import 'reset' ，不用加副檔名或下底線(_)，編譯程式會自動尋找對應的檔案:

```
@import 'reset';

body {
  background-color: #efefef;
  font: 100% Helvetica, sans-serif;
}
```

> 註: CSS中也有 @import 的定義，也可以加入其他的CSS檔案，語法有點小差異，詳見MDN這裡的說明。

混合(Mixin)

混合(Mixin)有點像是個函式或是一群值的組合，也可以輸入一個值然後套用這個值的整串結果，經常用於需要相符不同瀏覽器品牌的CSS3宣告上，來解決供應商前綴字的問題，例如以下的範例:

```
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}

.box { @include border-radius(10px); }
```

要記住的是 @minix 標記需要對應到 @include 標記，上面的 @include border-radius(10px) 可以像變數一樣，加在scss檔案中的任一個定義中。

這個 mixin 中的定義傳入值，是可以加上預設值的，例如以下的範例:

```
@mixin label($text: 'Code', $background: $yellow, $color: rgba(black, 0.5)) {
  position: relative;
  &:before {
    background: $background;
    color: $color;
    content: $text;
    display: inline-block;
    font-size: .6rem;
  }
}
```

混合(Mixin)經常被使用，在大部份時候如果你需要整個群組或整個類別的CSS套用，主要都是使用它，因為CSS3很多新的定義都會有供應商前綴字的問題，網路上也有很多現成的混合庫可以使用，例如以下三種常見的函式庫，但這裡面也不只有混合(Mixin)而已，還有很多其他的內容，直接使用可以節省時間，三種基本上都使用gem來安裝:

- Bourbon: bourbon.io
- Compass: compass-style.org
- Susy: susy.oddbird.net

擴充/繼承(Extend)

擴充(Extend)是可以擴充原有的CSS類別定義，你可以再加上不同的定義或覆蓋原有的定義，例如下面的範例:

```
.message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  @extend .message;
  border-color: green;
}
```

```
.error {
  @extend .message;
  border-color: red;
}
```

轉成CSS會像下面這樣，你可能會注意到 .message 仍然被保留著:

```
.message, .success, .error {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  border-color: green;
}

.error {
  border-color: red;
}
```

當不需要那個只被用來的繼承的那個CSS類別時，可以使用佔位符(%)來定義這個只被繼承之用的CSS類別，像 %message 這樣使用就行了:

```
%message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  @extend %message;
  border-color: green;
}

.error {
  @extend %message;
  border-color: red;
}
```

擴充(Extend)的用法相當簡單，相較於混合(Mixin)它不能傳入值，而且會合併相同的定義在同一個類別之中。

函式(Functions)

SCSS中也有 @function 可以定義自訂的函式，也有一些簡單的程式語法，例如 @return 、 @if 、 @for 、 @each 等流程控制的特性，以及簡單的運算能力。

```
@function calculate-width ($col-span) {
  @return 100% / $col-span;
}

.span-two {
  width: calculate-width(2); // spans 2 columns, width = 50%
}

.span-three {
  width: calculate-width(3); // spans 3 columns, width = 33.3%
}
```

SCSS中也內建了許多工具函式，詳見內建函式的清單，常用的例如按比例加亮顏色的 lighten 與按比例加深顏色的 darken ，裡面有關於顏色、透明度、字串、數字的函式很多。以下為範例:

```
$awesome-blue: #2196f3;

a {
  background-color: $awesome-blue;
```

```
    padding: 10px 15px;
  }

  a:hover {
    background-color: darken($awesome-blue,10%);
  }
```

註解(Comment)

用類似程式碼寫法的兩條斜線(//)就可以加入註解。CSS的話是規定要用 `/*...*/` 的註解記號，相較之下會簡單些。

~~15分鐘分隔線~~以下的內容將超過15分鐘。

安裝

由於大部份的基礎工具一開始都是使用Ruby語言開發，所以相關的編譯工具或檢查工具，或是幾套知名的圖形化介面的應用程式，都是要先在電腦上安裝RubyGem才能安裝， Ruby 1.9版本後都有附RubyGem。(安裝方式: Windows裝RubyInstaller，Mac OS X用homebrew裝ruby)。不過這幾套知名的圖形化介面應用程式都只有支援Mac OS X。

另一種新的選擇則是使用node平台的npm套件管理程式來安裝所需的工具，這些工具已經很完整的移植到node平台上，雖然還不是100%的與原來的工具相容，但可用程度也很高了。現在很多編輯程式也支援這些node上面的工具程式。node平台(npm)的方式也是支援各種作業系統的，安裝node.js就附有npm套件管理程式。

編譯器 & 檢查工具

原始的Sass編譯工具是使用Ruby語言寫的，後來的則有新的LibSass專案則是以C/C++寫的函式庫，可以很方便地移植到不同的程式語言平台中。

你可以先安裝Ruby後用gem指令安裝 `gem install sass` ，而另一個scss-lint工具則是另一個用Ruby寫的檢查工具，安裝方式類似。

另一個選擇是當如果你有使用node進行開發的話，可以用npm來安裝node-sass編譯工具套件，再搭配sass-lint的檢查套件。這兩個工具與上面的那種功能是相同的，視需求選擇其中一種即可，像我個人平常是有在開發JavaScript應用程式，所以是直接選擇用node-sass與sass-lint工具。

> 註: scss-lint與sass-lint這兩個套件沒事名字實在取得太像，很容易搞混。前面是RubyGem的套件，後面是npm的套件，看你要用那種平台，不要裝錯了。

檢查工具的規則

配合檢查工具所使用的檢查規則是在專案裡使用一個名稱為 `.scss-lint.yml` (RubyGem套件)或 `.sass-lint.yml` (npm套件)的檔案來設定，通常要有這個檔案在專案中，才會開始進行檢查。

基本的規則可以到scss-lint專案或sass-lint專案複製。

不過 `.scss-lint.yml` 與 `sass-lint.yml` 格式有點差異，如果需要轉換可以使用make-sass-lint-config工具轉換。

所謂的檢查並不光只是對語法錯誤的即時檢查，比較多的情況都是一些撰寫風格上的建議與提醒訊息。檢查工具的部份我會建議你使用與程式碼編輯工具搭配的套件，會比較容易使用，用命令列指引來檢查實在滿麻煩的。

命令列編譯工具的使用

與所有的命令列工具一樣，提供很多的參數可以在編譯時選擇使用，一般常用的編譯指令有下面幾個，因為原本的sass工具與node-sass有一些不同，所以分別列出來。實務上會直接用到指令的情況不多，也同樣會使用程式碼編輯程式中的外掛輔助功能，這裡只是提供一些簡單的參考。

單個檔案編譯

以sass為例，語法參數參考在這裡:

```
sass input.scss output.css
```

以node-sass為例，語法參數參考在這裡:

```
node-sass src/style.scss dest/style.css
```

對目錄中所有檔案編譯

如果要編譯整個目錄中的scss檔案到另一個目錄中時:

以sass為例，要使用 `--update` 參數

```
sass --update scss:css
```

以node-sass為例，需要加上 `--output` 指定輸出目錄:

```
node-sass scss/ --output css/
```

監控目錄中有修改的檔案進行編譯

這兩種工具都提供監控(--watch)的參數，意思是當檔案(.scss)有更動時，自動編譯有更動的檔案。監控不僅可以用在單一檔案上，也可以針對目錄中的所有檔案。不過這個指令一輸入就要一直開著命令列視窗才有效。

以sass為例:

```
sass --watch scss:css
```

以node-sass為例:

```
node-sass --watch scss/ -o css/
```

其他的參數

有兩種選項可以在編譯時提供更多的彈性，一種是輸出的風格。sass工具使用的是 `--style` 參數，而node-sass使用的是 `--output-style` 參數。風格都有以下四種:

- nested: 一般的CSS樣式，帶有以空白隔出的巢狀結構。
- expanded: 一般的CSS樣式。
- compact: 一般的CSS樣式，緊湊的排列，會把每個類別中的所有定義寫在同一行中。
- compressed: 經過壓縮的CSS樣式。

另一個選項是輸出原始碼地圖(sourcemap)檔案或內容的選項，原始碼地圖(sourcemap)是用來進行除錯使用的。sass工具使用的是 `--sourcemap` ，而node-sass除了 `--source-map` 參數還有其他的幾種選項。

常見編輯程式的外掛

Atom相關外掛

- sass-autocompile: 存檔時自動或是用快捷鍵編譯，支援.scss或.sass副檔名的檔案。(需額外安裝npm套件node-sass在全域中)
- css-snippets: CSS, SCSS, Sass, Less 程式碼片段或函式快速輸入。表列。
- atom-beautify: 漂亮格式化工具。
- autocomplete-sass: 給SASS用的CSS屬性名稱與數值自動完成(提示建議)
- 檢查工具(擇一):
    - linter-sass-lint: 檢查工具外掛，搭配node sass-lint(npm套件)

    - linter-scss-lint: 檢查工具外掛，搭配scss-lint(RubyGem套件)

Sublime Text 3

- Sass: 高亮度顯示與語法自動完成(提示建議)
- SassBuilder: 編譯輔助工具。需要先安裝sass編譯工具，而且需要有設定檔，參考說明文件。
- SassBeautify: 漂亮格式化工具。
- SASS Snippets: 程式碼片段或函式快速輸入。

Webstorm

內建基本的編輯高亮度顯示支援，可搭配外部的編譯工具進行編譯，請見這篇官網的說明文件

Visual Studio Code(VSC)

內建基本的編輯高亮度顯示支援，可搭配外部的編譯工具進行編譯，請見這篇官網的說明文件。其他的套件例如:

- Sass: 高亮度顯示與語法自動完成(提示建議)
- Sass Lint: 搭配sass-lint(npm套件)的檢查工具
- Stylesheet Formatter: 漂亮格式化工具。

Visual Studio 2015

- Web Compiler: 用於編譯SASS檔案的擴充。

Dreamweaver

Dreamweaver 2015/CC不支援，要使用另外的工具搭配。不過正在測試中的新版本已有加入支援，請見這篇2016.8日文的新版本特性說明，內容中可以下載測試版本。

Brackets

內建基本的編輯高亮度顯示支援，也有套件可以配合檢查工具。

結語

看到這裡記得開始安裝與使用吧，坐而言不如起而行。