CSS pre-processors

\ What are they and should we use them now? /

Mary and Crystal 2022.02.16

CSS pre-processor

A program that compiles unique syntax to regular CSS.

Compared with CSS, the syntaxes of CSS pre-processors have abundant features, such as nesting selectors and functions.

These features make the code more readable and easy to maintain.

Most common CSS pre-processors

A few CSS pre-processors are mainly used by some developers these days:

Sass, LESS, Stylus, etc.

These are open source which means we can get them for free, and the source code is public for further development.

Each CSS pre-processor has its own syntax rules.

SASS: Syntactically Awesome Style Sheets

- created in **2007** year by American programmer Hampton Catlin
- The most mature CSS pre-processor
- Two different syntaxes

SASS

```
$font-stack: Helvetica, sans-serif
$primary-color: #333

body
  font: 100% $font-stack
  color: $primary-color
```

SCSS

```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

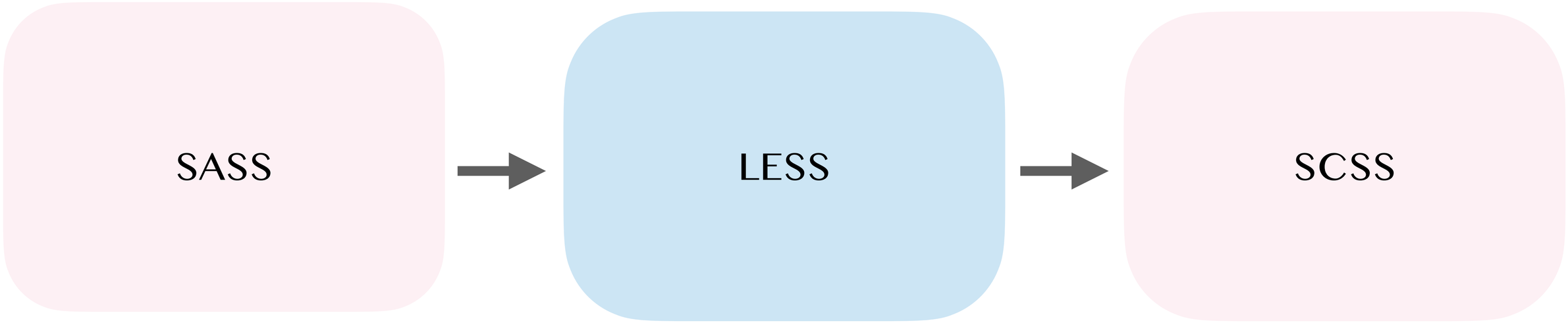
The first version of the SASS syntax uses .sass as the file extension and is indent-based. The code can be shorter because it uses indentation for nesting and does not use curly braces and semicolons. It is a strict syntax checking and easy to get errors.

The SCSS syntax was introduced later, and it uses .scss as the file extension. It supports the original CSS syntax with curly braces and semicolons. Developers can adapt SCSS syntax quickly. Therefore, the SCSS syntax made Sass became the most popular CSS pre-processor.

LESS: Leaner Style Sheets

- created in **2009** year by Alexis Seller
- The first version was written in Ruby
  the later version, replaced by JavaScript
- As it is a JavaScript-based CSS pre-processor, most browsers can run the compiler and read Less code directly. Developers can skip the installation and simply link to the .less file as the stylesheet and then call the compiler JavaScript file from the official provided Content Delivery Network (CDN).

Evolution of CSS pre-processors



Sass is the earliest CSS pre-processor created, and Less was inspired by Sass and created two years later. Less was designed as close as CSS and so became more popular than Sass at that time. The newer versions of Sass also introduced a CSS-like syntax called SCSS. And later on, it became more popular than Less and is used by many developers.

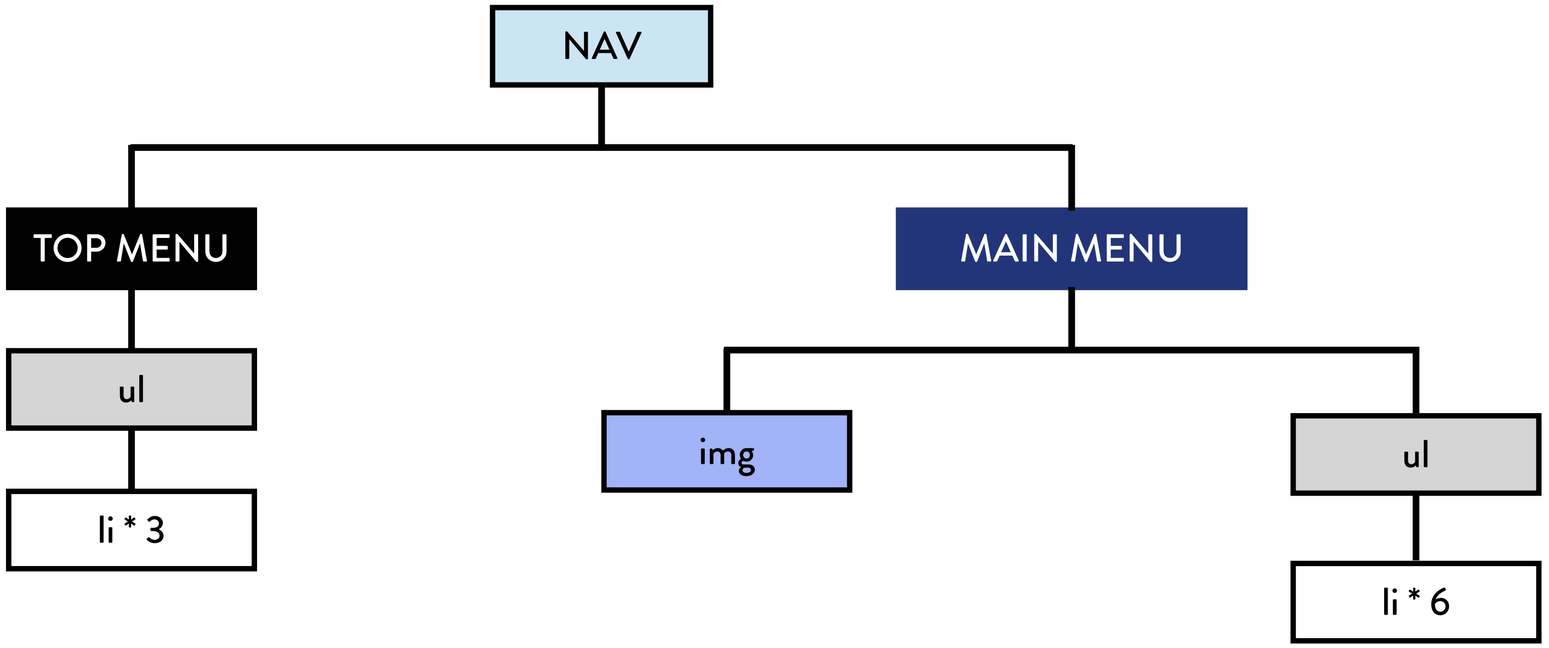
Search Faculties Portal

UNIVERSITY of GREENWICH

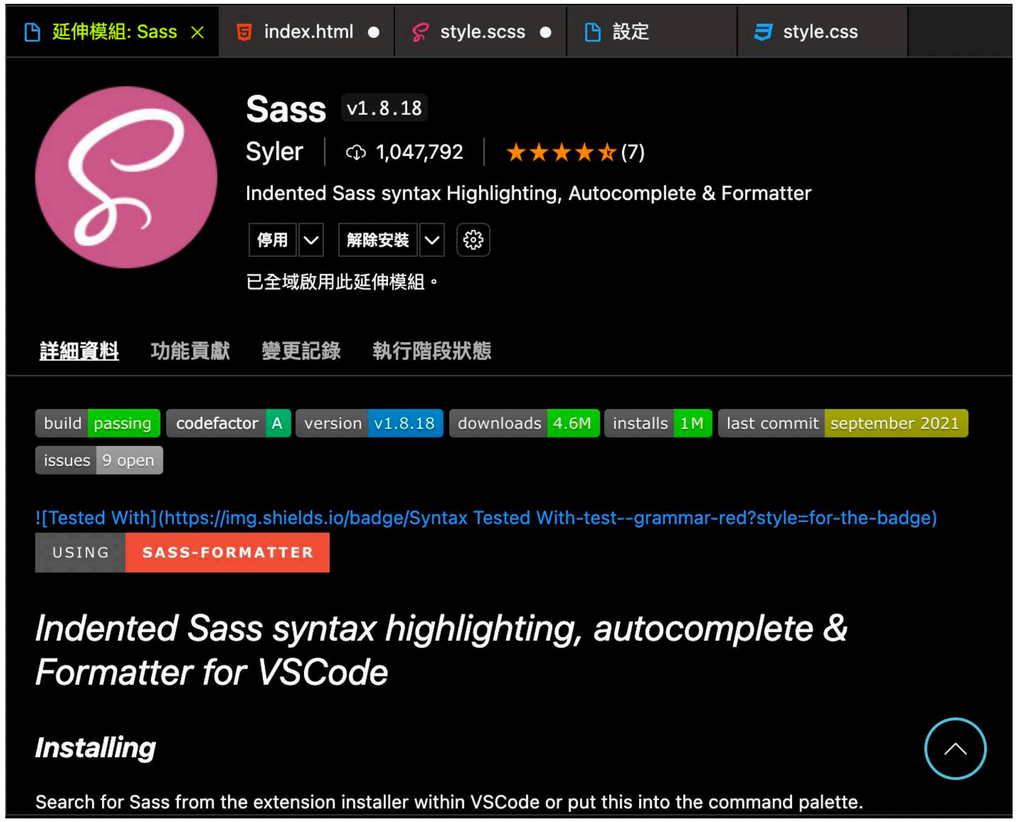
STUDY HERE SUBJECTS INTERNATIONAL RESEARCH BUSINESS & ENTERPRISE ALUMNI

Among those CSS pre-processors, I picked SCSS syntax as the example to introduce some basic functions. I used it to copy the navigation bar of the website of the University of Greenwich (UoG). There are two menus, including two unordered lists and an image.

Demo page



Install the SASS extension on Visual Studio Code



Before we start coding, we need to install the compiler package, and there are a few ways to install it. We can either install it using the command line or download Ruby and then run a command line from PowerShell. It's a bit complicated process for us to use the command line. The easiest way is to use Visual Studio Code editor. We just need to install the Sass extension simply.

1. create a SCSS file
2. start coding in SCSS syntax
3. When you finish coding or check the result, you just need to click the “Watch Sass” button.
4. It will be automatically compiled the code to a CSS file.
5. link that CSS file on our HTML file as usual.

Please be remember that don’t link to the .scss file as the browser does not support it.

Demo for introducing SASS’s features

Here is the demo page, which was written in SCSS syntax

This navbar alone uses Sass’s five common features and functions:

- Nesting
- Variables
- Lighten and Darken
- @mixin and @includes
- @function.

Nesting

SCSS

```scss
.gre-logo {
  margin-right: auto;
  img {
    width: 250px;
    height: auto;
  }
}
```

CSS

```css
.gre-logo {
  margin-right: auto;
}

.gre-logo img {
  width: 250px;
  height: auto;
}
```

In SCSS syntax, if the CSS selectors are in the same hierarchy of HTML code, we can nest the code.

In this case, a div with a class named `.gre-logo` and its image, we can write `img {}` inside `.gre-logo {}`.
This feature is called Nesting.

It makes our code has a clear nested and visual hierarchy. It is easier to read and maintain our code than CSS syntax.

After being compiled to CSS, it will become the usual way we write on CSS.

Variables

SCSS

```scss
$colour-gre-blue: #2e3e80;
$colour-dark: #1a2861;
$colour-white: #ffffff;
$width-content: 1200px;
```

```scss
.second-row {
  a {
    padding: divide(1em, 2) 1em;
    &:link,
    &:visited {
      color: $colour-gre-blue;
    }
    &:hover {
      background-color: $colour-gre-blue;
      color: $colour-white;
    }
```

CSS

```css
.second-row a:link,
.second-row a:visited {
  color: #2e3e80;
}

.second-row a:hover {
  background-color: #2e3e80;
  color: #ffffff;
}
```

It is more straightforward to set variables on Sass.
Start with a $ sign, followed by the name of the variable. Then we can assign the value we want.

I used it for the colours and the width of contents in our case. When I need to use the blue of UoG, I just need to assign `$colour-gre-blue` to those attributes. It will become the defined value after being compiled into a CSS file.

One day, if we need to change the colour, we just need to change the value of that variable.

Lighten & Darken

SCSS

```scss
a.gre-highlight {
  background-color: lighten($colour-gre-blue, 19%);
  &:hover {
    background-color: lighten($colour-gre-blue, 45%);
    color: darken($colour-gre-blue, 50%);
  }
}
```

CSS

```css
a.gre-highlight {
  background-color: #4f65c0;
}

a.gre-highlight:hover {
  background-color: #b1bae3;
  color: black;
}
```

One more feature I've used for colour is **Lighten** and **Darken**.
In our case, when we need to highlight the link to the Portal in a lighter tone of the blue of UoG as its background colour, we don't need to pick another colour with same tone on colour wheel.
We can use Lighten and then give it two parameters: the colour and the percentage of lightness.
Sass will help us calculate the required colour and show it on the compiled CSS file.
And it is the same way to darken colours.

@mixin & @include

SCSS

```scss
@mixin flexbox {
  display: flex;
  align-items: center;
}

.first-row {
  width: $width-content;
  margin: 0 auto;
  @include flexbox;
  justify-content: flex-end;
```

CSS

```css
.first-row {
  width: 1200px;
  margin: 0 auto;
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-align: center;
  -ms-flex-align: center;
  align-items: center;
  -webkit-box-pack: end;
  -ms-flex-pack: end;
  justify-content: flex-end;
}
```

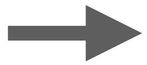
Except for setting a variable to a value,
we can use `@mixin` to define a set of reusable styles.
In our case, we use `@mixin` and then specify a name for the style, which is flexbox here. Then assign `display: flex` and `align-items: center` to it. When we need to use Flexbox, use `@include` and its defined name to call it.
This function allows us to reduce repeated code entry for the same style.
After being compiled to CSS, we can see that Sass gave us more rows named `-webkit-box` and `-ms-flexbox`.
They are used for increasing the compatibility with more browsers.

@function (e.g. Calculation)

SCSS

```scss
@function divide($a, $b) {
  @return $a / $b;
}

.second-row {
  a {
    padding: divide(1em, 2) 1em;
```



CSS

```css
.second-row a {
  padding: 0.5em 1em;
}
```

We can also do calculations on Sass. Here is just an example to see if it is working.
Let's use `@function` to make a division formula. A divided by B. Then give it a name and two parameters `($a, $b)`, and use `@return` to return the calculated value at the place called this function.
In this example, I tried to calculate 1em divided by 2, and it will return the answer, which is 0.5em, after being compiled to CSS.

LOOPS // MATHS // ARGUMENTS ...

Many more features and functions on Sass seem powerful.
You can go to the Documentation page on the Sass website to learn more if you are interested,
but it requires some basic understanding of programming languages to understand.

SASS: Documentation for more

Should we learn / use them now?

As a newbie of web design

As a newbie in web design who has only learned CSS for four months, my answer is NO.

Not that it’s bad, just that I’m not planning to start using it right away at this stage.

Consolidate the fundament of CSS first

CSS pre-processors make it easy to come up with clean code, but I would rather be able to write beautiful code by myself. Make sure my code is lean, well-formatted, and made good use of inheritability.

We all need a solid and sound foundation of vanilla CSS for ourselves, whether for work or to guide others in the future.

Easy to confuse the functions between CSS and CSS pre-processor

Since the syntax of CSS and CSS pre-processor are different but similar, we may feel confused or even misuse them when we are not yet familiar with them.

For example, setting variables in CSS and Sass are written differently. When not familiar with setting variables in CSS, we may use Sass one on CSS. As a result, it fails and then takes effort to debug. It's best not to learn similar languages simultaneously.

Lack of programming fundamentals to harness the power of CSS pre-processors

All CSS pre-processors have many extension functions containing programming concepts. For example, the conditional statement of if-else and loops on JavaScript. So that users can make calculations on styles.

However, for those who are just starting to learn JavaScript, it's like making things more complicated. It feels like CSS pre-processors are programs created by programmers for programmers. So I think **CSS pre-processors can be used more effectively after mastering at least one programming language.**

Self-assessment

Should you use them now?
You should use it if you answer “YES” for all questions.

1. **Are you confident in using CSS?**
2. **Do you have basic knowledge of any programming language?**
3. **Are you building a big project?**
4. **Are you able to debug by yourself when there is something wrong with your code?**

References

Sass-lang.com. (2019). Sass: Syntactically Awesome Style Sheets. [online] Available at: https://sass-lang.com/.

Wikipedia. (2022). Sass (stylesheet language). [online] Available at: https://en.wikipedia.org/wiki/Sass_(stylesheet_language) [Accessed 8 Mar. 2022].

lesscss.org. (n.d.). Getting started | Less.js. [online] Available at: https://lesscss.org/.

Wikipedia. (2022). Less (stylesheet language). [online] Available at: https://en.wikipedia.org/wiki/Less_(stylesheet_language) [Accessed 8 Mar. 2022].

Udemy. (n.d.). Advanced CSS and Sass: Flexbox, Grid, Animations and More. [online] Available at: https://www.udemy.com/course/advanced-css-and-sass/ [Accessed 8 Mar. 2022].

LambdaTest. (2019). CSS Preprocessors – Sass vs Less vs Stylus (With Examples). [online] Available at: https://www.lambdatest.com/blog/css-preprocessors-sass-vs-less-vs-stylus-with-examples/.

University of Greenwich. (n.d.). Welcome to the University of Greenwich, London. [online] Available at: https://www.gre.ac.uk/.