**Препроцесор**

CSS препроцесор – це програма, яка компілює написаний код(з використанням спец. синтаксису) в чистий CSS код. При цьому, препроцесор надає розробнику нові можливості та функції.

**Початок роботи з Sass**

На жаль *Sass* файли не можуть бути інтерпретовані браузером, тому вони потребують компіляції з **CSS** перш ніж вони будуть готові відображатися в браузері. Тому для початку вам потрібен інструмент, який допоможе перевести **.scss**(або **.sass**) файл в **.css**.

Крім того, саме час сказати про те, що Sass пропонує два різних синтаксиси - Sass і SCSS. Вони виконують однакові функції, просто виглядають по-різному. SCSS новіший і, як правило, вважається кращим, тому ми будемо використовувати його.

**Змінні**

Змінні в Sass працюють таким же чином, як і в будь-якій іншій мові програмування, мають ті ж самі принципи: типи даних і області застосування. Коли ми ініціалізуємо змінну, ми зберігаємо всередині неї певне значення, яке зазвичай часто повторюється в CSS, наприклад кольори, розміри, шрифти та інші значення.

**SCSS**

$title-font: 'Open Sans', sans-serif;  
$color-red: #F44336;  
$bg-white: #fff;  
  
h1.*title* {  
 font: $title-font;  
 color: $color-red;  
}  
  
div.*container* {  
 background-color: $bg-white;  
 width: 90%;  
 min-height: 100vh;  
 margin: 0 auto;  
}

**CSS**

h1.*title* {  
 font: "Open Sans", sans-serif;  
 color: #F44336;  
}  
  
div.*container* {  
 background-color: #fff;  
 width: 90%;  
 min-height: 100vh;  
 margin: 0 auto;  
}

Ідея всього цього в тому, що ми можемо в подальшому повторно використовувати одні і ті ж значення більш швидко, або якщо потрібно щось змінити, ми можемо це зробити в одному місці (визначення змінної), замість того щоб змінювати всюди це вручну, ми можемо використовувати цю властивість.

**Mixins**

Вважайте, що *міксини* (домішки) - це спрощений варіант класів в мовах програмування - ви можете захопити цілу групу CSS налаштувань і повторно використовувати на будь-який елемент, котрому ви забажаєте такий самий набір стилів. Домішки можуть приймати аргументи з можливістю установки значень за умовчуванням. У наведеному нижче прикладі ми оголошуємо *міксин* для квадрату і створюємо декілька різних за розміром і кольором елементів.

**SCSS**

*@mixin* square($size, $color) {  
 width: $size;  
 height: $size;  
 background-color: $color;  
}  
  
.*small-blue-square* {  
 *@include* square(20px, rgb(0,0,255));  
}  
  
.*big-red-square* {  
 *@include* square(300px, rgb(255,0,0));  
}

**CSS**

.*small-blue-square* {  
 width: 20px;  
 height: 20px;  
 background-color: blue;  
}  
  
.*big-red-square* {  
 width: 300px;  
 height: 300px;  
 background-color: red;  
}

**Наслідування**

Наступна особливість, яку ми розглянемо це @extend: вона дозволяє успадкувати властивості CSS одного селектора до іншого. Вона працює аналогічно *міксинам*, але наслідування краще в тому випадку, коли ми хочемо створити логічний зв'язок між елементами на сторінці.

Наслідування слід використовувати коли нам потрібно аналогічним чином оформити елементи, які відрізняються в деяких деталях. Наприклад, давайте створимо дві кнопки - одна для підтвердження, а інша для скасування діалогу.

**SCSS**

.*button* {  
 box-sizing: border-box;  
 color: #ffffff;  
 box-shadow: 0 1px 1px 0 rgba(0, 0, 0, 0.12);  
 padding: 12px 40px;  
 cursor: pointer;  
}  
  
.*confirm* {  
 *@extend* .*button*;  
 background-color: #28a745;  
 border-color: #28a745;  
}  
  
.*cancel* {  
 *@extend* .*button*;  
 background-color: #dc3545;  
 border-color: #dc3545;  
}

**CSS**

.*button*, .*cancel*, .*confirm* {  
 box-sizing: border-box;  
 color: #ffffff;  
 box-shadow: 0 1px 1px 0 rgba(0, 0, 0, 0.12);  
 padding: 12px 40px;  
 cursor: pointer;  
}  
  
.*confirm* {  
 background-color: #28a745;  
 border-color: #28a745;  
}  
  
.*cancel* {  
 background-color: #dc3545;  
 border-color: #dc3545;  
}

**Вкладеність**

В HTML слід дотримуватися сурової вкладеності структури, тоді як в CSS зазвичай повний хаос. Тож не дарма ми встановлювали препроцесор. Sass допоможе організувати таблицю стилів таким чином, щоб вона була якнайбільше подібна до вашої HTML розмітки, тим самим знижуючи ймовірність конфліктів CSS. Приклад зі список, що містить ряд посилань.

**SCSS**

ul {  
 list-style: none;  
 li {  
 padding: 15px;  
 display: inline-block;  
 a {  
 text-decoration: none;  
 font-size: 16px;  
 color: #444;  
 }  
 }  
}

**CSS**

ul {  
 list-style: none;  
}  
ul li {  
 padding: 15px;  
 display: inline-block;  
}  
ul li a {  
 text-decoration: none;  
 font-size: 16px;  
 color: #444;  
}

**Операції**

Використовувати математику в CSS дуже корисно. Sass має кілька стандартних математичних операторів, таких як +, -, \*, / і%. У нашому прикладі ми здійснюємо прості математичні обчислення для розрахунку ширини aside і article.

**SCSS**

.*container* {  
 width: 100%;  
 display: flex;  
 justify-content: space-around;  
}  
  
article {  
 width: 600px / 960px \* 100%;  
}  
  
aside {  
 width: 300px / 960px \* 100%;  
}

**CSS**

.*container* {  
 width: 100%;  
 display: flex;  
 justify-content: space-around;  
}  
  
article {  
 width: 62.5%;  
}  
  
aside {  
 width: 31.25%;  
}

Хоча і звичайний CSS тепер також пропонує цю функцію у вигляді **calc()**, але все ж таки Sass швидший для написання, має операцію по модулю (%) і може бути застосований до більш широкого діапазону типів даних (наприклад, кольору і рядків)

Додаткові джерела:

<http://sass-lang.com>

<http://sass-scss.ru>

<https://webref.ru/layout/sass>