[What's the difference between SCSS and Sass?](https://stackoverflow.com/questions/5654447/whats-the-difference-between-scss-and-sass)

Sass is a CSS pre-processor with syntax advancements. Style sheets in the advanced syntax are processed by the program, and turned into regular CSS style sheets. However, they do **not** extend the CSS standard itself.

CSS variables are supported and can be utilized but not as well as pre-processor variables.

For the difference between SCSS and Sass, this text on the [Sass documentation page](http://sass-lang.com/documentation/file.SASS_REFERENCE.html#syntax) should answer the question:

There are two syntaxes available for Sass. The first, known as **SCSS (Sassy CSS)** and used throughout this reference, is an **extension of the syntax of CSS**. This means that every valid CSS stylesheet is a valid SCSS file with the same meaning. This syntax is enhanced with the Sass features described below. Files using this syntax have the **.scss** extension.

The second and **older syntax**, known as the **indented syntax (or sometimes just “Sass”)**, provides a more concise way of writing CSS. It **uses indentation rather than brackets** to indicate nesting of selectors, and **newlines rather than semicolons** to separate properties. Files using this syntax have the **.sass** extension.

However, **all this works only with the Sass pre-compiler** which in the end creates CSS. It is not an extension to the CSS standard itself.

*Sass* (*Syntactically Awesome StyleSheets*) have two syntaxes:

* a newer: *SCSS* (*Sassy CSS*)
* and an older, original: indent syntax, which is the original *Sass* and is also called *Sass*.

So they are both part of *Sass* preprocessor with two different possible syntaxes.

The most important difference between *SCSS* and original *Sass*:

***SCSS***:

* Syntax is similar to *CSS* (so much that every regular valid *CSS3* is also valid *SCSS*, but the relationship in the other direction obviously does not happen)
* Uses braces {}
* Uses semi-colons ;
* Assignment sign is :
* To create a *mixin* it uses the @mixin directive
* To use *mixin* it precedes it with the @include directive
* Files have the *.scss* extension.

**Original *Sass***:

* Syntax is similar to *Ruby*
* No braces
* No strict indentation
* No semi-colons
* Assignment sign is = instead of :
* To create a mixin it uses the = sign
* To use *mixin* it precedes it with the + sign
* Files have the *.sass* extension.

Some prefer *Sass*, the original syntax - while others prefer *SCSS*. Either way, but it is worth noting that [*Sass’s* indented syntax has not been and will never be deprecated](https://web.archive.org/web/20121108051331/http:/thesassway.com/news/sass-is-here-to-stay) (web archive).

The Sass .sass file is visually different from .scss file, e.g.

### Example.sass - sass is the older syntax

$color: red

=my-border($color)

border: 1px solid $color

body

background: $color

+my-border(green)

### Example.scss - sassy css is the new syntax as of Sass 3

$color: red;

@mixin my-border($color) {

border: 1px solid $color;

}

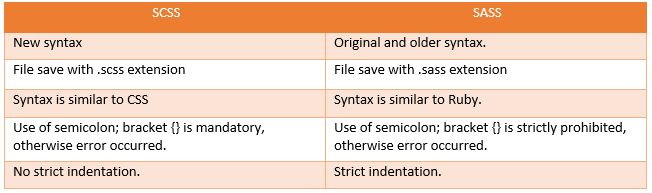
body {

background: $color;

@include my-border(green);

}

Any valid CSS document can be converted to Sassy CSS (SCSS) simply by changing the extension from .css to .scss.



An example of some SCSS and SASS syntax:

**SCSS**

$font-stack: Helvetica, sans-serif;

$primary-color: #333;

body {

font: 100% $font-stack;

color: $primary-color;

}

//Mixins

@mixin transform($property) {

-webkit-transform: $property;

-ms-transform: $property;

transform: $property;

}

.box { @include transform(rotate(30deg)); }

**SASS**

$font-stack: Helvetica, sans-serif

$primary-color: #333

body

font: 100% $font-stack

color: $primary-color

//Mixins

=transform($property)

-webkit-transform: $property

-ms-transform: $property

transform: $property

.box

+transform(rotate(30deg))

**Output CSS after Compilation(Same for Both)**

body {

font: 100% Helvetica, sans-serif;

color: #333;

}

//Mixins

.box {

-webkit-transform: rotate(30deg);

-ms-transform: rotate(30deg);

transform: rotate(30deg);

}