**SASS**

Syntactically Awesome Stylesheet is a CSS pre-processor, which helps to reduce repetition with CSS and saves time. It is more stable and powerful CSS extension language that describes the style of document structurally.

sass --watch C:\ruby\lib\sass\style.scss:style.css

SASS supports two syntaxes

1. SCSS: Sassy CSS files use abc.scss
2. Indented syntax: Older syntax and SASS files use abc.sass

For example: SCSS

.myclass {

color = red;

font-size = 0.2em;

}

.myclass

:color red

:font-size 0.2em

.class1,

.class2{

color:red;

}

**@import**

**In SCSS,**

@import "themes/blackforest";

@import "style.sass";

**In SASS,**

@import themes/blackforest

@import fontstyle.sass

**Mixin Directives**

@mixin and @include.

Instead of @mixin and @include you can use = and + characters

**For instance:**

=myclass

font-size: 12px;

p

+myclass

**The above given code is the same as −**

@mixin myclass

font-size: 12px;

p

@include myclass

**Deprecated Syntax**

= => It was used instead of : when setting variables and properties to values of SassScript.

||= => It was used instead of : whenever you are assigning default value of a variable.

! => Instead of $, ! was used as variable prefix. Functionality will not be changed when it is used instead of $.

**SASS in three different ways**

1. As a command line tool
2. As a Ruby module
3. As a plugin for Rack enable framework

sass input.scss output.css => It is used to run the SASS code from the command line.

sass --watch input.scss:output.css => It informs SASS to watch the file and update the CSS whenever SASS file changes.

sass --watch app/sass:public/stylesheets => It is used to watch the entire directory, if SASS contains many files in a directory.

**Rack/Rails/Merb Plugin**

**Rack** is a web server interface, which is used for developing web applications in Ruby. Enable the SASS in the Rails 3 version using the environment.rb file present under the config folder -

**config.gem "sass"**

You can use the following line to the Gemfile for the Rails 3 - **gem "sass"**

**Rails** is an open-source web framework that uses web standards such as JSON, HTML, CSS and JavaScript for displaying user interface. Enable the SASS in Rack application, add the following lines to the config.ru file, which is present in the app's root directory −

**require 'sass/plugin/rack'**

**use Sass::Plugin::Rack**

**Merb** is a web application framework, which provides speed and modularity to Rails. Enable theSASS in Merb by adding the following line to the config/dependencies.rb file −

**dependency "merb-haml"**

**Options**

Set the options in the environment.rb file of Rails or config.ru file of Rack application by using the following line –

Sass::Plugin.options[:style] = :compact

Also, set options in the init.rb file of Merb by using the following line −

Merb::Plugin.config[:sass][:style] = :compact

**CSS Extension & Description**

**Nested Rules =>** It is a way of combining multiple CSS rules within one another.

**Referencing Parent Selectors: & =>** It is the process of selecting parent selector by using the & character.

**Nested Properties =>** It allows nesting of properties into other properties which leads to grouping of another related code.

**Placeholder Selectors =>** Sass supports placeholder selector using class or id selector by making use of @extend directive.

**Comments**

Multiline comments − /\* and \*/

Single line comments − //

**Sass – Interpolation in Multiline Comments**

Interpolation within the multiline comments are resolved in the resulting CSS. You can specify variables or property names within the curly braces.

**Syntax**

$var : "value";

/\* multiline comments #{$var} \*/

**Watch**

sass --watch C:\ruby\lib\sass\style.scss:style.css

**SASS SCRIPT:**

|  |
| --- |
| **CSS Extension & Description** |
| [Interactive Shell](https://www.tutorialspoint.com/sass/interactive_shell.htm)  It evaluates SassScript expression using command line.  You can work easily with SassScript by using the interactive shell. You can run the shell with the SASS command line along with the i option.  Syntax: $ sass -i  Example:  C:/Ruby>sass -i  >> #666 + #666  #ccccccc |
| [Variables](https://www.tutorialspoint.com/sass/variables.htm)  It represents the data such as numeric values, characters or memory addresses.  Syntax:  $variable\_name : some value; |
| [DataTypes](https://www.tutorialspoint.com/sass/datatypes.htm)  It declares data type for every data object. |
| [Operations](https://www.tutorialspoint.com/sass/operations.htm)  It provides operations such as number, color, string, boolean and list operations. |
| [Parentheses](https://www.tutorialspoint.com/sass/parentheses.htm)  It is pair of signs which are usually marked off by round brackets ( ) or square brackets []. |
| [Functions](https://www.tutorialspoint.com/sass/functions.htm)  It supports for the use of functions by providing some keyword arguments. |
| [Interpolation](https://www.tutorialspoint.com/sass/interpolation.htm)  It provides SassScript variables and property names using **#{ }** syntax.  Syntax: #{$name}  Example:  p:after {  content: "I have #{8 + 2} books on SASS!";  } |
| [& in SassScript](https://www.tutorialspoint.com/sass/&_sassscript.htm)  It allows nesting of properties into another properties which leads to group of another related code. |
| [Variable Defaults](https://www.tutorialspoint.com/sass/var_defaults.htm)  It allows nesting of properties into another properties which leads to group of another related code. |

|  |
| --- |
| **Directives & Description** |
| [@import](https://www.tutorialspoint.com/sass/directives_import.htm)  It imports the SASS or SCSS files, it directly takes the filename to import.  @import "style.css";  @import "http://tutorialspoint.com/bar";  @import url(style);  @import "style" screen; |
| [@media](https://www.tutorialspoint.com/sass/directives_media.htm)  It sets the style rule to different media types. |
| [@extend](https://www.tutorialspoint.com/sass/directives_extend.htm)  @extend directive is used to share rules and relationships between selectors. |
| [@at-root](https://www.tutorialspoint.com/sass/directives_atroot.htm)  @at-root directive is a collection of nested rules, which is able to make style block at root of the document.  For example:  @media print {  .style {  height: 8px;  @at-root (without: media) {  color: #808000;;  }  }  } |
| [@debug](https://www.tutorialspoint.com/sass/directives_debug.htm)  @debug directive detects the errors and displays the SassScript expression values to the standard error output stream.  Example:  $font-sizes: 10px + 20px;  $style: (  color: #bdc3c7  );  .container {  @debug $style;  @debug $font-sizes;  } |
| [@warn](https://www.tutorialspoint.com/sass/directives_warn.htm)  @warn directive is used to give cautionary advice about the problem; it displays the SassScript expression values to the standard error output stream.  $main-color: #bdc3c7;  @warn "Darker: " darken($main-color, 30%); |
| [@error](https://www.tutorialspoint.com/sass/directives_error.htm)  @error directive displays the SassScript expression value as fatal error. |

**warn.scss**

$colors: (

blue: #c0392b,

black: #2980b9,

);

@function style-variation($style) {

@if map-has-key($colors, $style) {

@return map-get($colors, $style);

}

@error "Invalid color: '#{$style}'.";

}

.container {

style: style-variation(white);

}

You can tell SASS to watch the file and update the CSS whenever SASS file changes, by using the following command − sass --watch C:\ruby\lib\sass\warn.scss:warn.css

|  |
| --- |
| **Control Directive & Expression with Description** |
| [if()](https://www.tutorialspoint.com/sass/if_function.htm) – this returns only one result from two possible outcomes.  Syntax: if( expression, value1, value2 )  Example:- h2 { color: if( 1 + 1 == 2 , green , red); } |
| [@if](https://www.tutorialspoint.com/sass/if_directive.htm)  The *@if* directive accepts SassScript expressions and uses the nested styles whenever the result of the expression is anything other than *false* or *null*.  @Syntax: if expression { //CSS codes are written here }  Example:  p {  @if 10 + 10 == 20 { border: 1px dotted; }  @if 7 < 2 { border: 2px solid; }  @if null { border: 3px double; }  } |
| @else if  Syntax:  @if expression {  // CSS codes  } @else if condition {  // CSS codes  } @else {  // CSS codes  }  Example:  $type: audi;  p {  @if $type == benz {  color: red;  } @else if $type == mahindra {  color: blue;  } @else if $type == audi {  color: green;  } @else {  color: black;  }  } |
| [@for](https://www.tutorialspoint.com/sass/for_directive.htm)  The *@for* directive allows you to generate styles in a loop.  **Sass - @for to Keyword**  Syntax: @for $var from <start> to <end>  Example:  @for $i from 1 to 4 {  .p#{$i} { padding-left : $i \* 10px; }  }  **Sass - @for through Keyword**  Syntax: @for $var from <start> through <end>  Example:  @for $i from 1 through 4 {  .p#{$i} { padding-left : $i \* 10px; }  } |
| [@each](https://www.tutorialspoint.com/sass/each_directive.htm)  In *@each* directive, a variable is defined which contains the value of each item in a list.  **Sass - @each Directive**  Syntax: @each $var in <list or map>  Example:  @each $color in red, green, yellow, blue {  .p\_#{$color} {  background-color: #{$color};  }  }  **Sass - @each Multiple Assignment**  Syntax: @each $var1, $var2, $var3 … in <list>  Example:  @each $color, $border in (aqua, dotted), (red, solid), (green, double){  .#{$color} {  background-color : $color;  border: $border;  }  }  **Sass - @each Multiple Assignment with Maps**  Syntax: @each $var1, $var2 in <map>  Example:  @each $header, $color in (h1: red, h2: green, h3: blue) {  #{$header} {  color: $color;  }  } |
| [@while](https://www.tutorialspoint.com/sass/while_directive.htm)  It takes SassScript expressions and untill the statement evaluates to false it iteratively outputs nested styles.  Syntax:  while(condition) { // CSS codes }  Example:  $i: 50;  @while $i > 0 {  .paddding-#{$i} { padding-left: 1px \* $i; }  $i: $i - 10;  } |

|  |
| --- |
| **Directive & Description** |
| [Defining a Mixin](https://www.tutorialspoint.com/sass/defining_a_mixin.htm)  *@mixin* directive is used to define the mixin.  Syntax:  @mixin style {  .cont{  color: #77C1EF;  }  }  @include style; |
| [Including a Mixin](https://www.tutorialspoint.com/sass/including_a_mixin.htm)  *@include* directive is used to include the mixins in the document. |
| [Arguments](https://www.tutorialspoint.com/sass/arguments.htm)  The SassScript values can be taken as arguments in mixins, which is given when mixin is included and available as variable within the mixin.  @mixin bordered($color, $width: 2px) {  color: #77C1EF;  border: $width solid black;  width: 450px;  }  .style {  @include bordered($color:#77C1EF, $width: 2px);  } |
| [Passing Content Blocks to a Mixin](https://www.tutorialspoint.com/sass/passing_content_blocks_to_mixin.htm)  Block of styles are passed to the mixin.  @mixin element {  @content;  }  @include element {  .block {  color: green;  }  } |

Function Directive:

$first-width: 5px;

$second-width: 5px;

@function adjust\_width($n) {

@return $n \* $first-width + ($n - 1) \* $second-width;

}

#set\_width { padding-left: adjust\_width(10); }

Defining Custom SASS Functions

You can define your own SASS functions while using Ruby API. You can add your custom functions by adding them to Ruby methods as shown in the following code −

module Sass::Script::Functions

def reverse(string)

assert\_type string, :String

Sass::Script::Value::String.new(string.value.reverse)

end

declare :reverse, [:string]

end

|  |  |
| --- | --- |
| **S. No.** | **Operation & Description** |
| 1 | [Number Operations](https://www.tutorialspoint.com/sass/number_operations.htm)  It allows mathematical operations such as addition, subtraction, multiplication and division |
| 2 | [Color Operations](https://www.tutorialspoint.com/sass/color_operations.htm)  It allows using color components along with the arithmetic operations. |
| 3 | [String Operations](https://www.tutorialspoint.com/sass/string_operations.htm)  It uses + operation to concatenate strings. |
| 4 | [Boolean Operations](https://www.tutorialspoint.com/sass/boolean_operations.htm)  You can perform Boolean operations on SASS script by using *and*, *or* and *not* operators. |
| 5 | **List Operations**  Lists represent series of values, which are separated using commas or space. For information about lists, see the lists section under [data types](https://www.tutorialspoint.com/sass/datatypes.htm) section. |

**Data Types**

Data Type is a type of information, which requires declaring data type for every data object. The following table shows various data types supported by SassScript −

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Data Type & Description** | **Example** |
| 1 | **Numbers**  It represents integer types. | 2, 10.5 |
| 2 | **Strings**  It is sequence of characters defined within single or double quotes. | 'Tutorialspoint', "Tutorialspoint" |
| 3 | **Colors**  It is used for defining color value. | red, #008000, rgb(25,255,204) |
| 4 | **Booleans**  It returns true or false boolean types. | 10 > 9 specifies true |
| 5 | **Nulls**  It specifies null value which is unknown data. | if(val==null) {//statements} |
| 6 | **Space and Comma**  Represents the values which are separated by spaces or commas. | 1px solid #eeeeee, 0 0 0 1px |
| 7 | **Mapping**  It maps from one value to another value. | FirsyKey: frstvalue, SecondKey: secvalue |