SaSS: syntactically awesome style sheets

Extension to CSS language

Add feature which were not there in old basic CSS syntax.

Makes CSS development easier and improve the maintainability.

Sass Extends Css syntax

So we can define variable, nest CSS rules import other Sass inside other Sass files. With this we can write simple maintainable css files that are easy to read.

Two syntaxes used:

1. SCSS (Sassy css). File
2. SaSS (with extension .sass) uses indentation rather than brackets for nesting of style sheet new line rather than semicolon to separate properties

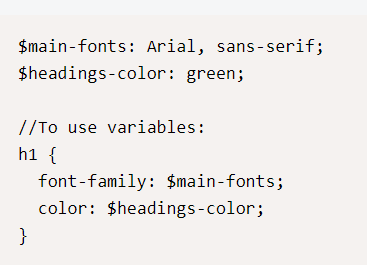
In inline style sheet we define the style tag type as type=”text/SCSS”

<style type='text/scss'>

Features:

1. Variables to store data and use it after inside selectors.

$variable name: value;



Css nesting

Old way of writing it is define Css at each level

<nav>

<ul>

<li></li>

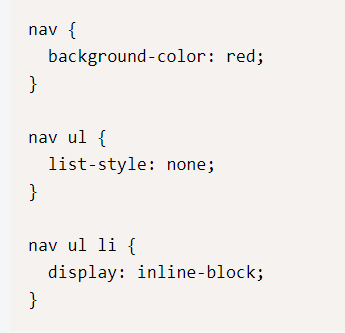
</ul>

<li>

Nav {}

Nav ul{}

Nal ul li{}



With Sassy way

Nav {

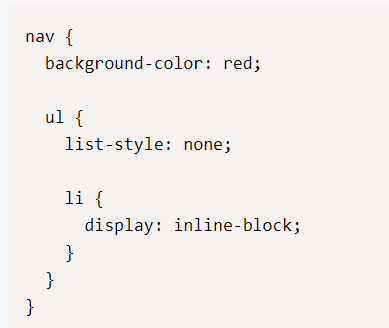
Ul{

Li{

}

}

}



3.Mixin:

Group of css declaration re used through the style sheet.

Eg:

New css features added will take some time to fully support in almost all the browser. Because the vendors of the browser new to upgrade their browser to support this.

Box shadow:

div {

-webkit -box-shadow: 0px 0px 4px #fff;

-moz-box-shadow: 0px 0px 4px #fff;

-ms-box-shadow: 0px 0px 4px #fff;

box-shadow: 0px 0px 4px #fff;

}

Its lot of css typing for all the boxes that uses similar box shadow Css

With mixin we can have a generic declaration to which we can pass value and generate the styles.

@mixin name($parametre1, $parameter2, $parameter3)

{

-webkit-rule:$ parametre1,$ parametre2, parametre3,

- moz -rule:$ parametre1,$ parametre2, parametre3,

}

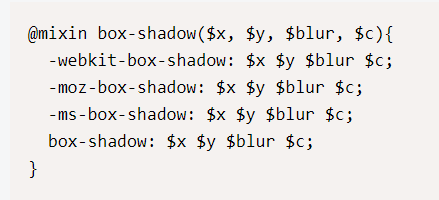
Use

Div{

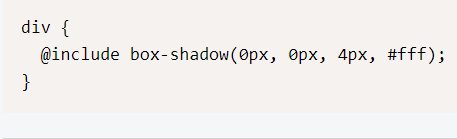
@include rule(value1, value2, value3);

}

define



use



4.Conditional statements inside lex mixin

@mixin css-class($param1)

{

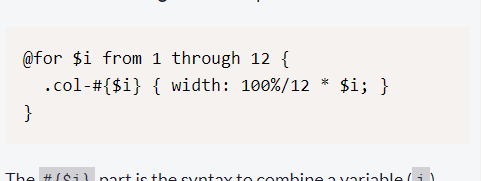
@if $param1==’’{}

@else if $param1==’’{}

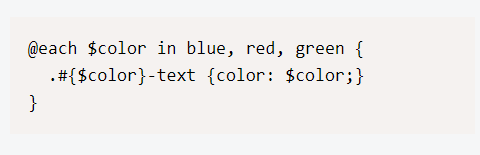
@else {}

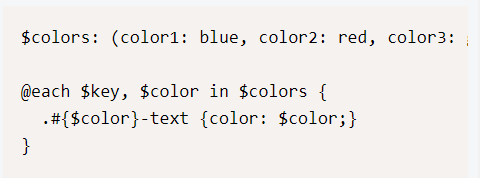
}

5.For looping

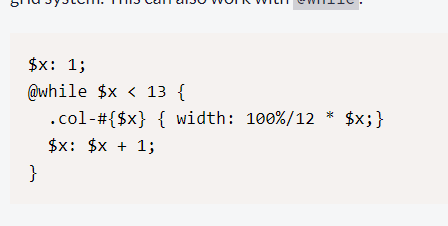


For each implementation:





While looping:



Module based development:

Instead of putting css in single file we can distribute the definition across multiple file called modules and its called module based development.

Then we can use

@import statement to import the css in another files.

All partial definition by convention begins with the \_as prefix

\_Mixin.scss

Main.scss

@import ‘mixin’

Extending the styles so that it can be shared among multiple elements.

.style1{

Css rules

}

.style2{

@extends .style1;

Css rules;



}