# RESUMEN SASS

### Sass compilando

### Comentarios

* **Comentarios multilínea**: Estos se escriben usando / \* y \* /. Los comentarios de varias líneas se conservan en la salida CSS.
* **Comentarios de una sola línea:** se escriben usando **//** seguido de comentarios. Los comentarios de una sola línea no persisten en la salida de CSS.

### Uso de variables

EJEMPLO

SCSS

$color\_botones: #F00;

.boton{

background-color: $color\_botones;

}

CSS resultante

.boton{

background-color: #F00;

}

### Nesting (anidación)

EJEMPLO

SCSS

nav{

ul{

margin: 0;

padding: 0;

li{

a{

color: blue;

}

}

}

}

CSS resultante

nav ul {

margin: 0;

padding: 0;

}

nav ul li a{

color: blue;

}

También se puede usar @media de forma anidada. Es decir, podemos crear media querys, para cada regla, manteniendo todo el código ordenado.

SCSS

.box{

background-color: #000;

@media (min-width: 1024px){

width: 960px;

margin: 0 auto;

}

@media (max-width: 1024px){

width: 96%;

margin: 0 auto;

}

}

CSS resultante:

.box{

background-color: #000;

}

@media (min-width: 1024px){

.box{

width: 960px;

margin: 0 auto;

}

}

@media (max-width: 1024px){

.box{

width: 96%;

margin: 0 auto;

}

}

### Selector Padre => &

EJEMPLO: SCSS

a{

font-weight: bold;

text-decoration: none;

&:hover{

text-decoration: underline;

}

li &{

color: red;

}

}

CSS resultante:

a{

font-weight: bold;

text-decoration: none;

}

a:hover{

text-decoration: underline;

}

li a{

color: red;

}

EJEMPLO: SCSS

#main {

color: black;

&-sidebar { border: 1px solid; }

}

CSS resultante:

#main {

color: black;

}

#main-sidebar {

border: 1px solid;

}

### Propiedades anidadas

SCSS

.font-definition{

font:{

family: Lato;

size: 24px;

weight: bold;

}

border :{

style :

radius :

}

}

CSS resultante

.font-definition{

font-family: Lato;

font-size: 24px;

font-weight: bold;

}

### Importar con @import

Se utiliza cuando se trabaja con un archivo scss para cada módulo, se divide cada sección para que cada una tenga su propio fichero scss, manteniendo una mejor organización.

Con SASS, sí que se puede hacer, ya que posteriormente al compilar, se generará un sólo archivo CSS con todo el contenido de los archivos. Por tanto podemos tener un archivo principal «base» que llame al resto, con la directiva:

@import ‘nombre\_archivo’

El resultado, será el contenido de este «nombre\_archivo» junto al resto del CSS.

### Hojas parciales

Para importar estilos escritos en archivos scss, pero que no queremos compilar en el css (no se generarán hojas css diferentes)

* El nombre de los archivos llevará un \_NombreArchivo al inicio del nombre
* Llamarlas desde un archivo que se compilará en css: @import NombreArchivo (sin \_ inicial)

### Mixins con @mixin

Nos permite definir estilos que posteriormente se pueden reutilizar. Nos permiten agrupar valores CSS para mejorar su eficiencia. Son reglas fijas que pueden invocarse en la hoja de estilo cuando sea necesario sin tener que volver a insertar el código completo. Esto ayuda a **trabajar más rápido y a mantener el código más ligero.**

Para trabajar eficazmente con mixins, se necesitan dos directivas:

* *@mixin :* con la primera se crea el modelo
* @include: con la segunda se inserta el bloque de código

Pueden llevar variables para tener más dinamismo.

Uno de los usos más extendidos, es crear mixins para gestionar los prefijos de los navegadores. Si queremos un botón con esquinas redondas, podemos crear el siguiente mixins

EJEMPLO

SCSS

Definición:

@mixin border-radius($radius){

-webkit-border-radius: $radius;

-moz-border-radius: $radius;

-ms-border-radius: $radius;

border-radius: $radius;

}

Llamada:

.mi-boton{

@include border-radius(10px);

}

CSS resultante

.mi-boton{

-webkit-border-radius: 10px;

-moz-border-radius: 10px;

-ms-border-radius: 10px;

border-radius: 10px;

}

EJEMPLO

SCSS

@mixin texto-grande{

font:{

family: Roboto;

size: 40px;

weight: bold;

}

}

h1{

@include texto-grande;

color: blue;

}

CSS resultante

h1{

font-family: Roboto;

font-size: 40px;

font-weight: bold;

color: blue;

}

EJEMPLO

SCSS

@mixin flex ($direction){

display :flex ;

justify-content :center ;

align-items : center ;

flex-direction : $direction ;

}

.container{

@include flex(column)

}

### Extend, la herencia de SASS

## La directiva garantiza que todas las propiedades de una clase se transfieran a las de otra. Usando @extend se evita tener que redefinirlo todo.

## La directiva también funciona como una cadena. Una clase definida por @extend puede a su vez formar parte de una tercera clase.

## Se puede heredar de varios selectores.

EJEMPLO

SCSS

.mensaje{

border: 3px solid black;

padding: 20px;

color: white;

}

.exito{

@extend .mensaje{

background-color: green;

}

}

.error{

@extend .mensaje{

background-color: red;

}

}

CSS resultante:

.mensaje{

border: 3px solid black;

padding: 20px;

color: white;

}

.exito{

border: 3px solid black;

padding: 20px;

color: white;

background-color: green;

}

.error{

border: 3px solid black;

padding: 20px;

color: white;

background-color: red;

}

También puede extenderse de una clase ya extendida:

SCSS

.mensaje{

border: 3px solid black;

padding: 20px;

color: white;

}

.error{

@extend .mensaje{

background-color: red;

}

}

.error-serio{

@extend .error{

font-size: 30px;

}

}

Donde error-serio, tendrá todos los atributos de error, y a su vez de mensaje.

### %

Se utiliza un signo de porcentaje (%) para identificar a una clase que se crea únicamente con el fin de utilizarla en otras clases.

Si al diseñar el sitio web no se importa un selector de este tipo, SASS tampoco lo compilará en CSS:

$bg-color:green;

$grey: #D3D3D3 ;

%module {

margin: 20px;

padding: 10px;

color: $bg-color;

}

%barranav{

Background-color: blue;

Border: 2px 2px;

}

.post {

@extend %module;

color: $grey;

}

En el código CSS final, la clase *module* ya no aparece. Sus propiedades se transfieren directamente a la clase *post*.

EJEMPLO

SCSS

.post {

margin: 20px;

padding: 10px;

color: green;

}

.post {

color: #D3D3D3;

}

### etiqueta *!optional*

Si escribes una extensión de una clase que no existe, SASS generará un error durante la compilación.

Con !optional se evita, porque SASS ignorará entonces el comando si no encuentra una clase adecuada

### Maps

Listas de valores que se pueden recuperar. Los valores son clave: valor y separados por comas (excepto el último de la lista).

Definir la lista y después recuperar un valor con map-get (clave).

Para recorrer la lista entera, utilizar @each

EJEMPLO

SCSS

$font-weights: (

“light”: 300,

“regular”: 400,

“bold”: 700

);

h1{

font-weight: map-get ($font-weights, light)

}

EJEMPLO

## Hacer una acción para cada valor del mapa. Recorrerla con each (variable1: guardará la clave, variable2: guardará el valor, variable3: nombre de la lista)

SCSS

$icons:(

abacus: “\f640”,

address-book: “\f2b9”,

bell:”\f0f3”,

igloo: “\f7ae”

)

@each $clave, $icon in $icons{

.fa-#{$clave}:before{ content: $icon;}

}

.fa-abacus:before{content: “\f640”;

### Interpolacion #

Para que el parámetro pasado elimine las comillas.

SCSS

$ruta-imagenes: '../imagenes';

body{

background: url(#{$ruta-imagenes}/fondo.png);

}

### Funciones

Las funciones nos permiten construir operaciones complejas y reutilizarlas en el código. Para definir una función utiliza **@function** seguido del nombre de tu función y las variables. Una función también debe contener el valor que retornará.

Sass trae definidas algunas funciones: <https://www.tutorialsteacher.com/sass/sass-functions>

Las funciones se crean con la directiva ***@function***, aunque en realidad necesitan siempre dos directivas: además de la inicial **@function**, se necesita un **@return** anidado con el cual se define el valor de salida:

EJEMPLO1

SCSS

Change-color: función de Sass que reemplaza el color por el indicado

mix: función de Sass que mezcla colores

invert: invierte los componentes RGB individualmente

<https://www.tutorialsteacher.com/sass/sass-color-functions>

DEFINICIÓN DE LA FUNCIÓN

@function invert($color, $amount:100%){

$inverse: change-color($color, $hue:hue($color) + 180);

@return mix ($inverse, $color, $amount);

}

LLAMADA

$primary-color: rgb(245, 3, 48);

.header{

background-color: invert($primary-color, 80%);

}

EJEMPLO2

SCSS

$column-count: 12;

@function column-width($num) {

@return $num \* 100% / $column-count;

}

.three-columns {

width: column-width(3);

}

### Bucles (loops)

Los bucles proporcionan al lenguaje de hojas de estilo el atractivo de un lenguaje de programación real. En SASS se utilizan para crear bloques de instrucciones que se repiten **hasta que tiene lugar una condición determinada**. Existen tres directivas diferentes para crear bucles:

* @for
* @while
* @each
  1. **@for** es el **loop estándar en programación**: comienza al inicio y repite la orden hasta que se alcanza un estado de salida y con ello el final. En SASS, esta directiva se presenta en dos variantes diferentes: o bien el último ciclo se ejecuta una vez más cuando se alcanza el objetivo o se abandona el bucle antes.

Con through (-le) se indica que la cuarta repetición también debe realizarse, mientras que con to (-lt) el bucle se detiene después de la tercera vuelta.

Si para el inicio se define un valor superior al elegido para el final, SASS cuenta hacia atrás.

#{$i}: la variable se incrementará en 1 con cada vuelta.

EJEMPLO

SCSS

@for $i from 1 through 4 {

.width-#{$i} { width: 10em + $i; }

}

@for $i from 1 to 4 {

.height-#{$i} { height: 25em \* $i; }

}

* 1. **La directiva@while** tiene una mecánica muy similar a la de @for pero, mientras que este último tiene puntos fijos de inicio y final, un bucle **@while** contiene una consulta de tipo lógico: mientras un estado siga siendo verdadero, las órdenes deberán repetirse.

Con este tipo de bucle, primero se debe **asignar un valor a las variables**

También se ha de integrar un comando en el bucle que haga que $i aumente con cada vuelta

EJEMPLO

SCSS

$i: 1;

@while $i < 5 {

.width-#{$i} { width: 10em + $i; }

$i: $i + 1;

}

* 1. **La directiva *@each* funciona de manera diferente.**

**Este bucle se basa en una lista de datos** especificada por el usuario que el bucle recorrerá en cada vuelta. Para cada entrada, @each hace una repetición diferente. De este modo, indicando una lista con los valores 1, 2, 3 y 4 sería posible generar el mismo resultado que con los otros bucles. La verdadera ventaja de este bucle, sin embargo, es que también puede introducirse otra información en la lista exceptuando valores numéricos (con @each, por ejemplo, se insertan varias imágenes diferentes en el diseño). Puedes introducir los datos directamente en la directiva o introducir la lista en una variable y luego llamarla.

Necesitas una variable. Esta asume el nombre de una de las entradas de la lista en cada vuelta.

EJEMPLO

SCSS

$list: dog cat bird dolphin;

@each $i in $list {

.image-#{$i} { background-image: url('/images/#{$i}.png'); }

}

### Ramificación

Con la directiva @if una orden solo se ejecuta si tiene lugar un cierto evento; en caso contrario, se ejecuta otro comando. Además de esta directiva, también hay una **función if()**. No guardan relación, pero pueden aparecer juntas.

1. La función es fácil de explicar. Contiene tres parámetros: la condición y dos salidas diferentes.

La primera salida se emite si el primer parámetro es verdadero, de lo contrario la función reproduce el tercer parámetro.

if( expression, value1, value2 )

EJEMPLO

SCSS

$black: #000000;

$white: #ffffff;

$text-color: $black;

body {

background-color: if($text-color == $black, $white, $black);

}

La función verifica si la variable $text-color está configurada en negro (black). Si este es el caso, el fondo se mostrará blanco. En cualquier otro caso, el CSS mostraría el fondo en negro.

1. Para requisitos más complejos, conviene emplear la **directiva @if**, que cuenta con la ventaja de poder distinguir más de dos casos entre sí:

EJEMPLO

SCSS

$black: #000000;

$white: #ffffff;

$lightgrey: #d3d3d3;

$darkgrey: #545454;

@mixin text-color($color) {

@if ($color == $black) {

background-color: $white;

}

@else if ($color == $white) {

background-color: $black;

}

@else if ($color == $lightgrey) {

background-color: $black;

}

@else {

background-color: $white;

}

}

p {

@include text-color($lightgrey);

}