



Diseño de Interfaces Web

UD – 3

SASS

Contenidos



ÍNDICE

Introducción.....	3
Instalar SASS.....	3
<i>Ventajas.....</i>	<i>4</i>
<i>Desventajas.....</i>	<i>4</i>
Elementos básicos.....	5
<i>Variables.....</i>	<i>5</i>
<i>Comentarios.....</i>	<i>6</i>
<i>Listas y Mapas.....</i>	<i>6</i>
<i>Interpolación.....</i>	<i>7</i>
<i>Anidamiento.....</i>	<i>7</i>
Compilar sass.....	9
Estructuras de Control.....	10
<i>@if / @else if / @else.....</i>	<i>10</i>
<i>@while.....</i>	<i>10</i>
<i>@for.....</i>	<i>11</i>
<i>@each.....</i>	<i>11</i>
Funciones.....	12
Mixins.....	12

1. INTRODUCCIÓN

Sass es un metalenguaje de Hojas de Estilo en Cascada (CSS). Es un lenguaje de script que es traducido a CSS, SassScript es el lenguaje de script en sí mismo.

2. INSTALAR SASS

El camino más rápido para instalar sass sería utilizando:

- Si tienes **Nodejs** Instalado:

```
npm install -g sass
```

- Si tienes **Windows Chocolatey**:

```
choco install sass
```

- Si tienes **Mac Homebrew**

```
brew install sass/sass/sass
```

Otra alternativa sería descargar el código de **github**:

<https://github.com/sass/dart-sass/releases>

▼ Assets 10		
dart-sass-1.55.0-linux-arm.tar.gz	3.07 MB	13 days ago
dart-sass-1.55.0-linux-arm64.tar.gz	3.17 MB	13 days ago
dart-sass-1.55.0-linux-ia32.tar.gz	16.1 MB	13 days ago
dart-sass-1.55.0-linux-x64.tar.gz	3.34 MB	13 days ago
dart-sass-1.55.0-macos-arm64.tar.gz	2.95 MB	13 days ago
dart-sass-1.55.0-macos-x64.tar.gz	3.16 MB	13 days ago
dart-sass-1.55.0-windows-ia32.zip	15.5 MB	13 days ago
dart-sass-1.55.0-windows-x64.zip	3.53 MB	13 days ago
Source code (zip)		13 days ago
Source code (tar.gz)		13 days ago

Los pasos a seguir :

- Descargar el fichero → **dart-sass-1.55.0-linux-x64.tar.gz**
- Descomprimir el fichero
- Añadir el ejecutable sass al path, utilizando el fichero **/home/agustin/.bashrc**

```
export PATH="/home/agustin/Docencia/2022-2023/DIW/Programas/dart-sass-1.55:$PATH"
```

```
agustin@agustin-pc:~$ gedit /home/agustin/.bashrc
^C
agustin@agustin-pc:~$ source .bashrc
agustin@agustin-pc:~$
agustin@agustin-pc:~$ sass --version
1.55.0
```

2.1 Ventajas

Las **ventajas** de utilizar Sass son todas aquellas derivadas de ser un “Preprocesador CSS”, es decir, básicamente usar Sass nos facilitará el desarrollo de CSS complejos proporcionando estructuras y herramientas para trabajar de forma más óptima y organizada.

- **Reduce el tiempo** para crear y mantener mi CSS.
- Permite una **organización modular** de mis estilos (proyectos grandes).
- Proporciona **estructuras de lenguaje de programación** (variables, listas, funciones, estructuras de control).
- Permite **generar distintos tipos de salidas** (comprimida, normal, minimizada)
- Permite regenerar **salida** (CSS) de **manera automática** (modo watch)
- **Herramientas, librerías, comunidad** etc....

2.2 Desventajas

Las **desventajas** de usar Sass son las mismas que aparecen al utilizar cualquier preprocesador:

- ¿**Necesidad de aprender** una nueva herramienta? ¿Desventaja?.
- Se **consume un tiempo** en la **compilación** para obtener nuestros estilos (la transformación de SCSS a CSS).
- **Sintaxis** más **compleja** que CSS.

3. ELEMENTOS BÁSICOS

3.1 Variables

Las variables son elementos típicos de lenguajes de programación en los que “guardamos” valores para utilizarlos posteriormente:

	# DEFINICIÓN
<code>\$nombre: expresion;</code>	
<code>\$miColor: red;</code>	
<code>\$miMargin: 16px;</code>	
	# USO
<code>a {</code>	
<code> color : \$miColor;</code>	
<code> margin : \$miMargin / 2;</code>	
<code>}</code>	

En el estilo SCSS

```
$blue: #3bbfce;
$margin: 16px;

.content-navigation {
  border-color: $blue;
  color: darken($blue, 9%);
}

.border {
  padding: $margin / 2;
  margin: $margin / 2;
  border-color: $blue;
}
```

Debe compilar a:

```
.content-navigation {
  border-color: #3bbfce;
  color: #2b9eab;
}

.border {
  padding: 8px;
  margin: 8px;
  border-color: #3bbfce;
}
```

Los **tipos de datos** a utilizar en las variables :

- Números *48 48px 0.5em*
- Cadenas *'left' "left" left*
- Colores *rg(255, 0, 0) hsl(0, 100%, 50%), #f00 #ff0000*
- Booleanos *true y false*
- null
- Mapas y Listas

Debes tener precaución con el ambito de las variables:

```
//Variable global fuera de todo bloque
$logo-width: 50%;

.header {
  //Variable local
  $header-width: 50%;
}
```

3.2 Comenarios

- Comentarios de una sólo línea // ESTO ES UN COMENTARIO
- Comentarios multilínea /* ESTO TAMBIÉN ES UN COMENTARIO */

La **recomendaciones** en el uso de comentarios:

- *Comenta todo lo que no sea evidente.*
- *Es casi imposible “comentar demasiado”.*
- *Describe las funciones.*
- *Explica la agrupaciones de las reglas y sus objetivos.*

3.3 Listas y Mapas

Sass nos proporciona dos tipos de datos más complejos como son las listas y los mapas.

- **Listas** → *Colecciones de valores*
- **Mapas** → *Colecciones de valores a los que accedemos por clave*

```
$iconos : ( // Tamaño de los iconos
    40px,
    80px,
    160px,
);

$indice:1;
$valor:nth($iconos,$indice);
```

```
// Tamaños de página

$pagina : (
    'pequeño' : 576px,
    'medio' : 768px,
    'grande' : 992px
);

$clave : medio;
$valor : map-get($pagina,$clave);
```

3.4 Interpolación

La Interpolación es una herramienta que nos proporciona Sass y que nos permite, casi en cualquier sitio del documento, insertar expresiones cuyo resultado, al ser evaluadas, formará un trozo del código CSS final.

#{expresión}

El resultado de evaluar esa expresión/operación formará parte de nuestro CSS final.

Las interpolaciones pueden ser usadas en lugares como:

- Selectores.
- Nombres de propiedades.
- Comentarios
- Reglas de Sass `@import`, `@extend` y `@mixins`
- Cadenas (con o sin comillas)
- Funciones
-

// Interpolación en selectores

```
$button-type: "error";
$btn-color : #f00;

.btn-#{ $button-type } {
    background-color: $btn-color;
}
```

//Interpolación en el uso de funciones

```
$fondo : "images/fondos/default.png";

.container {
    background-image: url("#{ $fondo }");
}
```

//Interpolación de comentarios

```
$fondo : "Agustin"
/*
    Web desarrollada por #{ $autor }
*/
```

3.5 Anidamiento

El anidamiento permite crear reglas más concisas y mejor organizadas:

```
// En css
nav {...}
nav li {...}
nav li a {...}

// En SASS
nav { ...
  li {...}
  a {...}
}
```

(referencia al padre)

CSS

```
a {...}
a:hover {...}
```

SASS

```
a { ...
  &:hover {...}
}
```

```
//Anidamiento básico
a {
  color: $link-normal;
  text-decoration: none;
  &:hover {
    color: $link-hover;
  }
  &:visited {
    color: $link-visited;
  }
  &:active {
    color: $link-active;
  }
}
```

(referencia al padre & Y SUFIJO)

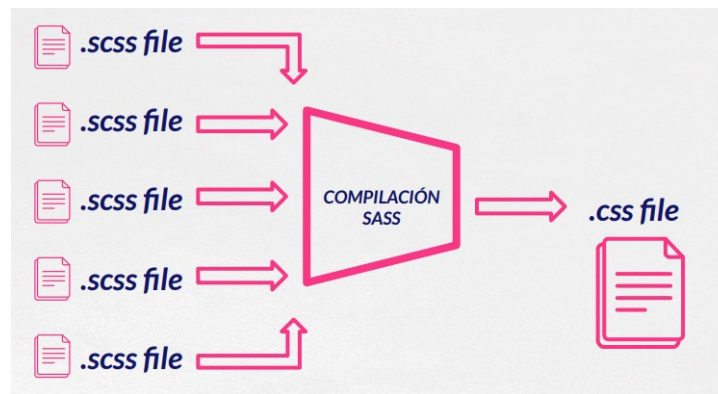
CSS

```
.btn {...}
.btn_warning {...}
```

SASS

```
.btn { ...
  &_warning {...}
}
```


4. COMPILAR SASS



SIMPLE

Sass entrada.**SCSS** salida.**CSS**

Existen diferentes tipos de Compilación:

- Simple
- Múltiple (varios ficheros)
- Expandida (por defecto)
- Comprimida
- Vigilando los cambios
-Y muchas más posibilidades incluyendo distintos flags

MULTIPLE

Sass file1.scss:output_file1.css ... fileN.scss:output_fileN.css ...

COMPRIMIDA (Quita la mayor cantidad de caracteres posible)

Sass --style=compressed file.scss output_file.css

VIGILANDO LOS CAMBIOS Y ACTUALIZANDO FICHEROS

Sass --watch file.scss output_file.css

5. ESTRUCTURAS DE CONTROL

Las estructuras de control son aquellas que nos van a permitir definir un flujo de ejecución a la hora de generar nuestras hojas de estilos. **Sass** nos proporciona las siguientes:

- **@if / @else if / @else**
- **@while**
- **@for**
- **@each**

5.1 @if / @else if / @else

Esta estructura nos permite controlar si un bloque **Sass** será evaluado o no atendiendo a una expresión o condición. De manera general tienen la siguiente estructura:

```
$light-theme: true;
$dark-theme: false;

header {
  @if $light-theme == true {
    background-color: #fff;
    color: #000;
  } @else if $dark-theme {
    background-color: #000;
    color: #fff;
  } @else { //Default theme
    background-color: #aaa;
    color: #444;
  }
}
```

Aclaración: Dependiendo de los valores de las variables *\$light-theme* y *\$dark-theme* tendremos una u otra combinación de colores y, en caso de que ambas sean falsas, un combinación de colores por defecto.

5.2 @while

Esta estructura de control nos sirve para evaluar de manera repetitiva ciertas órdenes **Sass**.

```
$num: 1;
$color-list: #0f0, #00f, orange, #ccc;

@while $num < 5 {
  td:nth-child(#{ $num }) {
    color: #f00;
    background-color: nth($color-list, $num);
  }
  $num: $num + 1;
}
```

Aclaración: Utilizamos el bucle para recorrer una lista de colores y hacer que las celdas de una tabla tengan diferentes colores dependiendo de la posición en la que se encuentran.

5.3 @for

Esta estructura de control es muy similar a la estructura @while. Las diferencias son que nosotros establecemos unos valores inicial y final para controlar el número de veces que se repite la estructura y que no tenemos que preocuparnos para la actualización de las variables de control. De manera general tiene la siguiente estructura:

```
num: 1;
$color-list: #0f0, #00f, orange, #ccc;

@for $i from 1 to 5 {
  p:nth-of-type(#{ $i }) {
    color: #000;
    background-color: nth($color-list, $i);
  }
}
```

Aclaración: Estamos dando unos colores a los 4 primeros párrafos de una página web y escogiendo esos colores de una lista de colores.

5.4 @each

Esta estructura es una estructura iterativa que utilizaremos para recorrer tanto listas como mapas.

Ejemplo de each recorriendo una lista

```
$usuarios: pepe, lola,manuel;

@each $u in $usuarios {
  .profile-#{ $e } {
    background: image-url("img/#{ $e }.png") no repeat;
  }
}
```

Ejemplo de each recorriendo un mapa

```
$mapa : pepe : pepe.png, lola: lola.png ,manuel: manuel.png;

@each $u,$v in $mapa {
  .perfil-#{ $u } {
    background: image-url("img/#{ $v }") no repeat;
  }
}
```

6. FUNCIONES

Al igual que en un lenguaje de programación en **Sass** podemos definir funciones en las que por un lado pondremos, normalmente un trozo de código que vayamos a utilizar frecuentemente y , por otro lado, nos deben devolver un valor.

```
// Definir una función
@function anchura-col($col,$total) {
  @return percentage($col/$total);
}

//Llamadas a la función

.sidebar {
  ...
  width: anchura-col(2,10);
  ...
}

.main {
  ...
  width: anchura-col(5,10);
  ...
}
```

7. MIXINS

Es una directiva de Sass que me permite definir estilos que luego puedo reutilizar a lo largo del resto mi hoja de estilos

```
@mixin box-shadow($color) {
  box-shadow: 2px 10px 24px $color;
}

.c-button {
  color: #FF0000;
  @include box-shadow(#FF0000);
}

.c-panel {
  color: #00FF00;
  @include box-shadow(#00FF00);
}
```