

Parte 4 de 4: Sass y Compass

DIRECTIVAS @IF Y @FOR
DEPURACIÓN DE ERRORES CON @DEBUG

Apuntes de: **Rosa María Medina Gómez**

Adaptados a Power Point por José Jesús Torregrosa García

Curso de Formación del Profesorado a distancia
Cefire Específico de FP de Cheste

Generalitat Valenciana Curso 2018 - 2019



Introducción

- En esta última sesión del curso, estudiaremos las directivas: **funtion**, **if**, **each**, **for/while** y veremos cuando usar cada una de ellas.
- Veremos también cómo utilizar el **debug** para facilitar la búsqueda de errores y Compass.
- Aparte hay un anexo de [**MasSobre.Compass.odt**](#)

Funciones

Para definir una función en Sass, la estructura a seguir es:

`@function` nombreFuncion([**arg**]){...}

Y cuando la vamos a invocar, los paréntesis **son obligatorios** aun no recibiendo ningún parámetro como entrada, ejemplo:

```
application.scss x
1 // @function nombreFuncion([arg])
2 @function fluidize(){
3     @return 35%;
4 }
```

Funciones

Si nos fijamos, la definición de una función es exactamente igual que como vimos cómo definir un **mixin**, pero en este caso tenga o no tenga por defecto parámetros de entrada **los paréntesis son obligatorios**. Dentro de la función, podemos determinar un valor de retorno.

Funciones

application.scss x

```
1  @function fluidize(){
2  |    @return 35%;
3  |  }
4  .sidebar{
5  |    width: fluidize();
6  |  }
7  // @function nombreFuncion([arg])
8  @function fluidize(){
9  |    @return 35%;
10 |  }
```

Funciones

- Otro ejemplo:

 application.scss 

```
1  @function fluidize2($target, $context){  
2    |    @return ($target/$context) * 100%;  
3  }  
4  .sidebar2{  
5    |    width: fluidize2(350px, 1000px);  
6  }
```

Sentencia If

La sentencia if corresponde a una estructura condicional. Por ejemplo, si queremos que nuestra cabecera tenga un fondo negro si el tema es dark:

```
application.scss x
1  $theme: dark;
2  header{
3      @if $theme == dark{
4          background: #000;
5      }
6  }
```



Sentencia If

Hay varios operadores para realizar una comparativa en Sass:

- Igual $\rightarrow ==$
- No es igual $\rightarrow !=$
- Mayor que $\rightarrow >$ **(solo números)**
- Mayor o igual que $\rightarrow >=$ **(solo números)**
- Menor que $\rightarrow <$ **(solo números)**
- Menor o igual que $\rightarrow <=$ **(solo números)**

@if ... @else ...

- Un if...else...

```
application.scss x
1  $theme: light;
2  header{
3      @if $theme == dark{
4          background:  #000;
5      } @else{
6          background:  #fff;
7      }
```

@if ... @else ...

- En cascada:

```
application.scss x
1  $theme: pink;
2  header{
3    @if $theme == dark{
4      background: black;
5    } @else if $theme == pink{
6      background: pink;
7    } @else {
8      background: white;
9    }
10 }
```

@each (.scss)

@each nos permite iterar una lista por cada uno de los ítems que tiene. Por ejemplo, si tenemos una variable que contiene un conjunto de autores, podemos recorrerlos:

```
application.scss x
1  $autores: rosa lucia pablo jose;
2  @each $autor in $autores {
3      .author-#{ $autor }{
4          background: url(author-#{ $autor }.jpg);
5      }
6  }
```

@each (.css)

application.css x

```
1  /* line 3, ../../sass/teoriaSesion4/application.scss */
2  .author-rosa {
3    background: url(author-rosa.jpg);
4  }
5
6  /* line 3, ../../sass/teoriaSesion4/application.scss */
7  .author-lucia {
8    background: url(author-lucia.jpg);
9  }
10
11 /* line 3, ../../sass/teoriaSesion4/application.scss */
12 .author-pablo {
13   background: url(author-pablo.jpg);
14 }
15
16 /* line 3, ../../sass/teoriaSesion4/application.scss */
17 .author-jose {
18   background: url(author-jose.jpg);
19 }
```

@for

Estas sentencias funcionan como el @each pero a diferencia del anterior se encuentra asociado por lo general a números. Por ejemplo, si queremos realizar una iteración del 1 al 3 de forma que para cada uno de los valores del bucle (1..3) lo multiplicaremos por 30píxeles de forma que cuando compilemos tengamos una lista de ítems que use los i valores , tal como aparece a continuación:

@for (en .scss)

application.scss x

```
1  .item{
2      position: absolute; right: 0;
3      @for $i from 1 through 3{
4          &.item-#{ $i }{
5              top: $i * 30px;
6          }
7      }
8  }
```

@for (desplegado en .scss)

Con esto conseguimos que para cada uno de los ítems tenga un valor de **top** diferente.

```
# application.css x
1  /* line 1, ../../sass/tec
2  .item {
3      position: absolute;
4      right: 0;
5  }
6  /* line 4, ../../sass/tec
7  .item.item-1 {
8      top: 30px;
9  }
10 /* line 4, ../../sass/tec
11 .item.item-2 {
12     top: 60px;
13 }
14 /* line 4, ../../sass/tec
15 .item.item-3 {
16     top: 90px;
17 }
18
```

@while

- Darse cuenta del uso de variables en el while:

```
application.scss x
1  $i: 1;
2  .item{
3      position: absolute;
4      right: 0;
5      @while $i < 4{
6          &.item-#{ $i }{
7              top: $i * 30px;
8          }
9          $i: $i+1;
10     }
11 }
```


Cuándo usar funciones, mixin y cuándo extends

- **Mixin**: cuando definimos propiedades similares múltiples veces con pequeñas variaciones
- **Extend**: cuando tenemos propiedades que son exactamente iguales, es decir, declaraciones copiadas una y otra vez.
- **Funciones**: cuando tenemos operaciones que necesitamos usar muchas veces y que nos devuelven un valor.

Cuándo usar funciones, mixin y cuándo extends

- **Ejemplo:** asignamos un border-radius si el mixin recibe el valor de cierto o lo toma por defecto en **\$rounded**.

🔗 application.scss ✕

```
1  @mixin button($color, $rounded: true){
2      color:$color;
3      @if $rounded == true{
4          border-radius: 4px;
5      }
6  }
```

Cuándo usar funciones, mixin y cuándo extends

Ahora lo que estamos haciendo es que, si recibimos un valor de rounded, entonces le asignamos ese valor como border radius:

```
application.scss x
1  @mixin button($color, $rounded: false){
2      color: $color;
3      @if $rounded{
4          border-radius: $rounded;
5      }
6  }
7  .btn-a{
8      @include button(#000);
9  }
10 .btn-b{
11     @include button(#000, 4px);
12 }
```

Cuándo usar funciones, mixin y cuándo extends

En este caso, a `.btn-a` no le estamos dando un border, pero en cambio a `.btn-b` sí, ya que le estamos pasando como segundo argumento un valor distinto de falso o nulo y por tanto entra en el if y le asigna el border-radius que estamos enviándole por parámetro.

Depuración de errores

Antes de comenzar a ver cómo funcionan las directivas para la depuración de errores, quiero indicar cómo podemos ir viendo los cambios que vamos realizando en nuestro proyecto y cómo poder ver si ha entrado en alguna directiva como **@debug** **@error** o **@warn**. Para poder lanzar el observer de Compass.app, desde la consola debemos **ejecutar compass watch desde el directorio donde tenemos el proyecto.**

Depuración de errores

La directiva **@debug**, nos permite imprimir en la salida estándar de errores los valores de nuestro fichero de Sass. Esta directiva es bastante útil para la depuración de ficheros. La forma de usarlo es:

```
application.scss x
1  @debug 10em + 12 em;
2  // saldrá por pantalla al debugear:
3  // @debug 10em + 12 em;
```

Depuración de errores

La etiqueta **@warn** nos imprime también por la salida estándar, esta etiqueta es bastante útil para mostrar avisos sobre funciones que estamos usando y están deprecated, o para mostrar algún error que tengamos al usar algún mixin.

Hay dos diferencias principales entre el funcionamiento de @warn y @debug:

- Podemos activar las advertencias (**@warn**) con la opción de línea de comandos o con --quiet
- Se imprime una traza junto con el mensaje para que podamos ver cuáles son los estilos que causaron el warning (**@warn**).

Depuración de errores

application.scss x

```
1  @mixin border-radius($radius) {  
2      @warn "The `border-radius()` mixin will be deprecated in version 2.0.";  
3  
4      -webkit-border-radius: $radius;  
5      -moz-border-radius: $radius;  
6      -ms-border-radius: $radius; -o-border-radius: $radius; border-radius: $radius;  
7  }
```


Depuración de errores

La directiva `@error` lanza el valor de una expresión SassScript como un **error "fatal"** incluyendo un seguimiento de la pila (**no termina de compilar el código**). Es útil para validar argumentos para mixin y funciones. Por ejemplo:

Bibliografía y/o páginas de interés

- Sass: <http://sass-lang.com/>
- Ruby: <https://www.ruby-lang.org/en/>
- Sass Wikipedia: [https://en.wikipedia.org/wiki/Sass_\(stylesheet_language\)](https://en.wikipedia.org/wiki/Sass_(stylesheet_language))
- Less wikipedia: [https://en.wikipedia.org/wiki/Less_\(stylesheet_language\)](https://en.wikipedia.org/wiki/Less_(stylesheet_language))
- Stylus wikipedia:
[https://en.wikipedia.org/wiki/Stylus_\(stylesheet_language\)](https://en.wikipedia.org/wiki/Stylus_(stylesheet_language))
- compass: <http://compass.kkbox.com/>
- scout: <http://mhs.github.io/scout-app/>
- Koala: <http://koala-app.com/>