

---

---

# SASS

Alfonso Tienda v.2019  
atienda@iprocuratio.com

---

# ¿Quién soy?

Alfonso Tienda Braulio

Ingeniero Informático en la UPV, MBA, PRINCE2, PMP...

CEO en iProcuratio Consultores

Technology Strategy Leader en Alfatec Sistemas

...

# Ahora vosotros...

¿Qué sabéis hasta ahora?

¿Qué os interesa?

...

# ¿Como vamos a trabajar?

Esto es un taller para aprender. Yo ya me lo sé.

Parejas / tríos ..

Salis y enseñáis el trabajo..

...

---

# SASS

¿Qué es SASS?

---

# ¿Qué es SASS?

Sass, o Syntactically Awesome StyleSheets, es una extensión del lenguaje CSS. Con Sass, puedes escribir hojas de estilo limpias y flexibles, además de solucionar los clásicos problemas de repetición y mantenimiento en el CSS tradicional.

Además de ser una competencia valiosa para un desarrollador front-end, cambiar de CSS a SASS es bastante suave. Es por ello que estudiaremos la sintaxis SCSS (Sassy CSS).





# SASS

## → Variables, funciones...

Se pueden usar variables, funciones, estructuras condicionales...

## → Modulos

Se pueden dividir los scss en módulos, en proyectos grandes

## → Compilado (Traspilado)

Sass no es interpretado directamente por el navegador. Hay que traspilarlo a css, mediante un ejecutable en Ruby.

# —

# Descargando y “compilando” SASS



Instalar Ruby, SASS

... O ...

<https://prepros.io/>



—

# Anidamiento

# Anidamiento de selectores


## Nesting

### SASS

```
.parent {  
  blockquote, p {  
    font-family: Times;  
  }  
  a {  
    color: teal;  
  }  
  span {  
    background: yellow;  
  }  
}
```

### CSS

```
.parent blockquote,  
.parent p {  
  font-family: Times;  
}  
.parent a {  
  color: teal;  
}  
.parent span {  
  background: yellow;  
}
```



Anidar es el proceso de colocar selectores dentro del alcance de otro selector:

El alcance de una variable es el contexto en el que una variable está definida y disponible para su uso.

En Sass, es útil pensar en el alcance de un selector como cualquiera de los códigos entre su apertura {y el cierre} de llaves.

Los selectores que están anidados dentro del alcance de otro selector se conocen como hijos. El selector anterior se conoce como el padre. Esto es igual a la relación observada en los elementos HTML.

# Anidamiento de propiedades

Se pueden anidar propiedades con los dos puntos tras la primera parte

```
1 div {  
2   font: {  
3     family: "Times New Roman";  
4     size: 2em;  
5     style: italic;  
6     variant: small-caps;  
7     weight: bold;  
8   };  
9 }
```

```
1 div {  
2   font-family: "Times New Roman";  
3   font-size: 2em;  
4   font-style: italic;  
5   font-variant: small-caps;  
6   font-weight: bold;  
7 }
```

# Anidamiento : más

En algunos puntos se habrá de usar el símbolo & para hacer referencia al padre

```
1 a {  
2   text-decoration: none;  
3   &:hover {  
4     text-decoration: underline;  
5   }  
6 }  
7
```

```
1 a {  
2   text-decoration: none;  
3 }  
4 a:hover {  
5   text-decoration: underline;  
6 }
```

—

# Variables

## Variables.

Las variables en SCSS nos permiten asignar valores a un identificador.

Al contrario que en CSS que hay que sustituir los valores a lo largo del documento, en SCSS un cambio de la variable será reflejado en múltiples reglas.

El símbolo \$ se usa para definir una variable

# Variables

```
1 $bg-color: rgb(178,455,223,0.7);
2
3 ▾ #uno {
4     background-color: $bg-color;
5     color: red;
6 }
7
8 ▾ #dos {
9     background-color: $bg-color;
10    color:blue;
11 }
12 |
```



```
2     background-color: rgba(178, 255, 223, 0.7);
3     color: red;
4 }
5
6 ▾ #dos {
7     background-color: rgba(178, 255, 223, 0.7);
8     color: blue;
9 }
```

# Variables: tipos.

- Numéricos. 1, 2, 10, 10px. Aunque lleven una unidad asociada (px, em) se les considera numéricos
- Strings, como “Arial”, ‘blue’ o span.
- Boleanos, esto es *true* o *false*
- *null*
- Listas
  - 4px solid black;
  - Helvetica, Arial, sans-serif;
- Mapas
  - (key1: value1, key2: value2);



—

# Mixins

# Mixins



Los mixins permiten reutilizar estilos pegándolos

dart-sass v1.18.0

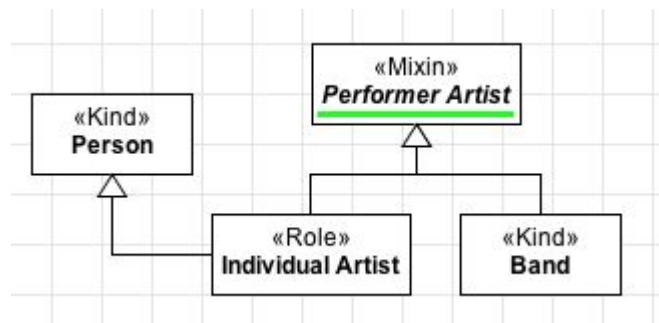
```
1 ▾ @mixin buttons {  
2   color: white;  
3   width: 30px;  
4   border-radius: 10px;  
5 }  
6 ▾ .btn-a {  
7   @include buttons;  
8   background: blue;  
9 }  
10 ▾ .btn-b {  
11   @include buttons;  
12   background: black;  
13 }
```

CSS

```
1 ▾ .btn-a {  
2   color: white;  
3   width: 30px;  
4   border-radius: 10px;  
5   background: blue;  
6 }  
7  
8 ▾ .btn-b {  
9   color: white;  
10  width: 30px;  
11  border-radius: 10px;  
12  background: black;  
13 }
```

## — Mixin: Patrón de diseño

- En los lenguajes de programación orientada a objetos, un mixin es una clase que ofrece cierta funcionalidad para ser heredada por una subclase, pero no está ideada para ser autónoma.
- Heredar de un mixin no es una forma de especialización sino más bien un medio de obtener funcionalidad.
- Una subclase puede incluso escoger heredar gran parte o el total de su funcionalidad heredando de uno o más mixins mediante herencia múltiple.



## Reglas generales de los mixins.

- Se definen con @mixin
- Se usan con @include
- Buena práctica escribir primero el @include y luego las propiedades singulares para poder sobrescribir alguna de las que nos trae el Mixin si fuera necesario.
- @mixin ha de declararse antes de usar los @include.

# Mixins: Argumentos



Los mixins permiten el uso de varios argumentos

```
1 ▾ @mixin rounded($radio) {  
2   border-radius: $radio;  
3   -moz-border-radius: $radio;  
4   -webkit-border-radius: $radio;  
5 }  
6  
7 #footer { @include rounded(5px); }  
8 #sidebar { @include rounded(8px); }
```

```
1 ▾ #footer {  
2   border-radius: 5px;  
3   -moz-border-radius: 5px;  
4   -webkit-border-radius: 5px;  
5 }  
6  
7 ▾ #sidebar {  
8   border-radius: 8px;  
9   -moz-border-radius: 8px;  
10  -webkit-border-radius: 8px;  
11 }
```

# Mixins: Argumentos



Los argumentos permiten valores por defecto

Los argumentos por defecto deben ir siempre al final de la cadena, si son varios

```
1 @mixin rounded($radio: 10px) {  
2   border-radius: $radio;  
3   -moz-border-radius: $radio;  
4   -webkit-border-radius: $radio;  
5 }  
6  
7 #navbar li { @include rounded;  
8 #footer { @include rounded(5px);  
9 #sidebar { @include rounded(8px); }
```

```
1 #navbar li {  
2   border-radius: 10px;  
3   -moz-border-radius: 10px;  
4   -webkit-border-radius: 10px;  
5 }  
6  
7 #footer {  
8   border-radius: 5px;  
9   -moz-border-radius: 5px;  
10  -webkit-border-radius: 5px;  
11 }  
12  
13 #sidebar {  
14   border-radius: 8px;  
15   -moz-border-radius: 8px;  
16   -webkit-border-radius: 8px;  
17 }
```

# Mixins: Argumentos



Se puede pasar una lista como varios argumentos, con el spread (similar a javascript)

...

```
1 ▾ @mixin roundedandcolored($color, $radio: 10px) {  
2   border-radius: $radio;  
3   -moz-border-radius: $radio;  
4   -webkit-border-radius: $radio;  
5   background-color: $color;  
6 }  
7  
8 $navbar: yellow, 40px;  
9 #footer { @include roundedandcolored(blue, 5px); }  
10 #sidebar { @include roundedandcolored(green); }  
11 #navbar { @include roundedandcolored($navbar...); }
```

```
1 ▾ #footer {  
2   border-radius: 5px;  
3   -moz-border-radius: 5px;  
4   -webkit-border-radius: 5px;  
5   background-color: blue;  
6 }  
7  
8 ▾ #sidebar {  
9   border-radius: 10px;  
10  -moz-border-radius: 10px;  
11  -webkit-border-radius: 10px;  
12  background-color: green;  
13 }  
14  
15 ▾ #navbar {  
16   border-radius: 40px;  
17   -moz-border-radius: 40px;  
18   -webkit-border-radius: 40px;  
19   background-color: yellow;  
20 }
```

# Mixins: String Interpolation



Variables dentro de un string

Con #{ }

```
1 ▾ @mixin roundedandcolored($color, $radio: 10px) {  
2   border-radius: $radio;  
3   -moz-border-radius: $radio;  
4   -webkit-border-radius: $radio;  
5   background-color: light#{ $color };  
6 }  
7  
8 $navbar: yellow, 40px;  
9 #footer { @include roundedandcolored(blue, 5px); }  
10 #sidebar { @include roundedandcolored(green); }  
11 #navbar { @include roundedandcolored($navbar...)
```

```
1 ▾ #footer {  
2   border-radius: 5px;  
3   -moz-border-radius: 5px;  
4   -webkit-border-radius: 5px;  
5   background-color: lightblue;  
6 }  
7  
8 ▾ #sidebar {  
9   border-radius: 10px;  
10  -moz-border-radius: 10px;  
11  -webkit-border-radius: 10px;  
12  background-color: lightgreen;  
13 }  
14  
15 ▾ #navbar {  
16   border-radius: 40px;  
17   -moz-border-radius: 40px;  
18   -webkit-border-radius: 40px;  
19   background-color: lightyellow;  
20 }
```



—

# Funciones

# Funciones



Puedes también definir funciones que devuelven valores

```
1 ▾ @function calculate-rem($size) {  
2   $rem-size: $size / 16px;  
3   @return #{ $rem-size }rem;  
4 }  
5  
6  
7 ▾ .my-class {  
8   font-size: calculate-rem(50px);  
9 }  
10
```

```
1 ▾ .my-class {  
2   font-size: 3.125rem;  
3 }
```



# Funciones: Built-in

- **Existen multitud de funciones preconstruidas**

<https://sass-lang.com/documentation/functions>

- **Numéricas, String, Colores...**

Para tratar tipos básicos y propiedades

- **Listas y mapas**

Tratamiento de listas y mapas.

- **Selectores**

Acceso al motor de selectores.

- **Introspección**

Funciones para acceso interno a funcionamiento de SASS.

—

# Imports


# Imports

```
// foundation/_code.scss
code {
  padding: .25em;
  line-height: 0;
}
```

```
// foundation/_lists.scss
ul, ol {
  text-align: left;

  & & {
    padding: {
      bottom: 0;
      left: 0;
    }
  }
}
```

```
// style.scss
@import 'foundation/code', 'foundation/lists';
```



Importa ficheros en el lugar que se incluye @import  
Como convención los ficheros que por sí sólo no funcionan (se compilan) se inician con \_  
Los imports respetan el anidamiento

—

# Extending

# @extend



Extiende (hereda) a otra clase

```
1 .btn-a {  
2   background: blue;  
3   color: white;  
4   width: 30px;  
5   border-radius: 10px;  
6 }  
7 .btn-b {  
8   @extend .btn-a;  
9   background: black;  
10 }
```

```
1 .btn-a, .btn-b {  
2   background: blue;  
3   color: white;  
4   width: 30px;  
5   border-radius: 10px;  
6 }  
7  
8 .btn-b {  
9   background: black;  
10 }  
11
```

# @extend



Mas opciones...

```
1 ▾ .btn {  
2   color: white;  
3   width: 30px;  
4   border-radius: 10px;  
5 }  
6  
7 ▾ .btn-a {  
8   @extend .btn;  
9   border-radius: 10px;  
10 }  
11 ▾ .btn-b {  
12   @extend .btn;  
13   background: black;  
14 }
```

```
1 ▾ .btn, .btn-a, .btn-b {  
2   color: white;  
3   width: 30px;  
4   border-radius: 10px;  
5 }  
6  
7 ▾ .btn-a {  
8   border-radius: 10px;  
9 }  
10  
11 ▾ .btn-b {  
12   background: black;  
13 }  
14
```



# @extend



## Con anidación

```
1 ▾ .contenido {  
2   background: #eaeaea;  
3   padding: 10px;  
4  
5 ▾   p {  
6     font-size: 1.5em;  
7     margin-bottom: 1em;  
8   }  
9 }  
10  
11 ▾ .lateral {  
12   @extend .contenido;  
13   background: #ddd;  
14 }
```

```
1 ▾ .contenido, .lateral {  
2   background: #eaeaea;  
3   padding: 10px;  
4 }  
5  
6 ▾ .contenido p, .lateral p {  
7   font-size: 1.5em;  
8   margin-bottom: 1em;  
9 }  
10  
11 ▾ .lateral {  
12   background: #ddd;  
13 }  
14 }
```

# @extend



Placeholder selectors: Sólo son clases para ser extendidas, no existen por sí mismas. Se definen con %

```
1 ▾ %btn {  
2     background: blue;  
3     color: white;  
4     width: 30px;  
5     border-radius: 10px;  
6 }  
7  
8 ▾ .btn-a {  
9     @extend %btn;  
10    background: blue;  
11 }  
12 ▾ .btn-b {  
13     @extend %btn;  
14    background: black;  
15 }
```

```
1 ▾ .btn-a, .btn-b {  
2     background: blue;  
3     color: white;  
4     width: 30px;  
5     border-radius: 10px;  
6 }  
7  
8 ▾ .btn-a {  
9     background: blue;  
10 }  
11  
12 ▾ .btn-b {  
13     background: black;  
14 }  
15
```

—

# Logging

@debug, @warn, @error

```
$color-blue: #1c94c6;  
$font-sizes: sm, p, bq, heading, hero;  
$colors: (  
  brand-red: #c0392b,  
  brand-blue: #2980b9,  
  text-gray: #2c3e50,  
  text-silver: #bdc3c7  
);
```

```
.element {  
  @debug $color-blue; // single value  
  @debug $font-sizes; // list  
  @debug $colors; // map  
  @debug 4em * 3; // math expression
```



Tiempo de compilación

—

# Control de flujo

# @if and @else

```
$light-background: #f2ece4;
$light-text: #036;
$dark-background: #6b717f;
$dark-text: #d2e1dd;

@mixin theme-colors($light-theme: true) {
  @if $light-theme {
    background-color: $light-background;
    color: $light-text;
  } @else {
    background-color: $dark-background;
    color: $dark-text;
  }
}

.banner {
  @include theme-colors($light-theme: true);
  body.dark & {
    @include theme-colors($light-theme: false);
  }
}
```

```
1 .banner {
2   background-color: #f2ece4;
3   color: #036;
4 }
5
6 body.dark .banner {
7   background-color: #6b717f;
8   color: #d2e1dd;
9 }
10
```

# @each

```
$sizes: 40px, 50px, 80px;
```

```
@each $size in $sizes {  
  .icon-#{$size} {  
    font-size: $size;  
    height: $size;  
    width: $size;  
  }  
}
```

```
1 ▾ .icon-40px {  
2   font-size: 40px;  
3   height: 40px;  
4   width: 40px;  
5 }  
6  
7 ▾ .icon-50px {  
8   font-size: 50px;  
9   height: 50px;  
10  width: 50px;  
11 }  
12  
13 ▾ .icon-80px {  
14   font-size: 80px;  
15   height: 80px;  
16   width: 80px;  
17 }  
18
```

# @for

```
$base-color: #036;  
  
@for $i from 1 through 3 {  
  ul:nth-child(3n + #{ $i }) {  
    background-color: lighten($base-color, $i * 5%);  
  }  
}
```

```
1 ▾ ul:nth-child(3n + 1) {  
2   background-color: #004080;  
3 }  
4  
5 ▾ ul:nth-child(3n + 2) {  
6   background-color: #004d99;  
7 }  
8  
9 ▾ ul:nth-child(3n + 3) {  
10  background-color: #0059b3;  
11 }  
12
```



# @while

```
/// Divides `$value` by `$ratio` until it's below `$  
@function scale-below($value, $base, $ratio: 1.618)  
  @while $value > $base {  
    $value: $value / $ratio;  
  }  
  @return $value;  
}  
  
$normal-font-size: 16px;  
sup {  
  font-size: scale-below(20px, 16px);  
}
```

```
1 sup {  
2   font-size: 12.36094px;  
3 }  
4
```

—

# Muchas Gracias