

Esta clase va a ser

- grabada

Clase 14. DESARROLLO WEB

SASS II

Temario

13

SASS I

- ✓ Preprocesadores CSS
- ✓ Nesting, import y vars

14

SASS II

- ✓ [Operadores, condicionales y bucles](#)
- ✓ [Maps](#)

15

Servidores y SEO para tu sitio

- ✓ Servidores
- ✓ Buenas prácticas, optimizando imágenes
- ✓ SEO: conceptos básicos

Objetivos de la clase

- **Agregar** operaciones y decisiones con SASS.
- **Usar** reglas SASS.

Glosario

SASS: es un preprocesador de CSS que te permite escribir un código, el cual luego se transforma (compila) en un archivo de CSS puro.

Esto genera un código más limpio y sencillo de mantener y editar, a través de una estructura ordenada, usando un lenguaje de estilos.

- ✓ **Sintaxis:** en Sass cuentas con dos diferentes tipos de sintaxis: SCSS y SASS. La primera y más popular, es conocida como SCSS (Sassy CSS). Es muy similar a la sintaxis nativa de CSS, tanto así que te permite importar hojas de estilos CSS (copiar y pegar) directamente en un archivo SCSS, y obtener un resultado válido.

Nesting o anidación: con la anidación de SASS, puedes organizar tu hoja de estilo de una manera que se asemeja a la de HTML, lo que reduce la posibilidad de conflictos en el CSS.

Import: te permite incluir la fuente de tus archivos individuales en una hoja de estilo maestra.

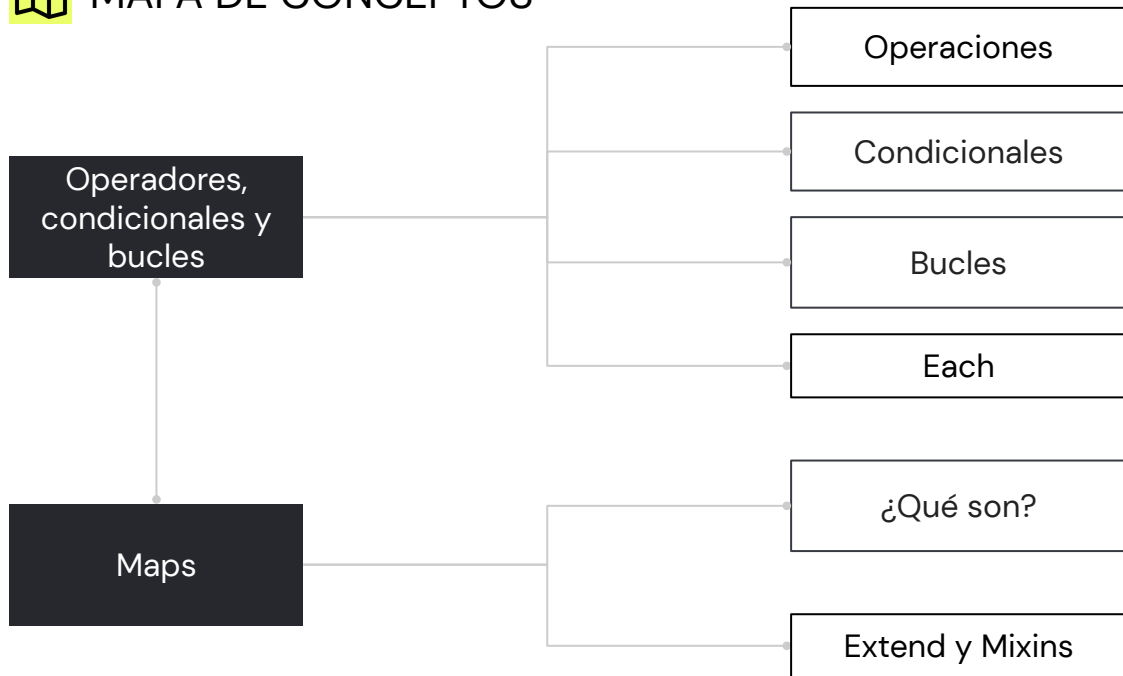
CLASE N°10

Glosario

Vars (variables): son una manera de guardar información que necesites reutilizar en tus hojas de estilos: colores, dimensiones, fuentes o cualquier otro valor. SASS utiliza el símbolo dólar (\$) al principio de la palabra clave para crear una variable.



MAPA DE CONCEPTOS



Vamos a iniciar el procesador CSS

Extend

Extend

A menudo, al diseñar una página **una clase debe tener todos los estilos de otra clase**, así como **sus propios estilos específicos**. En esos casos usamos **@extend**, para traer los estilos de otra clase.

Por ejemplo, la metodología **BEM** fomenta las clases modificadoras que van en los mismos elementos que las clases de bloque o elemento. Pero esto puede crear HTML desordenado, es propenso a errores al olvidar incluir ambas clases, y puede traer problemas de estilo no semántico a su marcado.

Extend

```
<div class="redsocial  
redsocial--nueva">  
  ¡Esta es una nueva red social!  
</div>
```

HTML

```

.boton {
    width:200px;
    height:50px;
    background-color:white;
    text-align:center;
    color:black;
    border-radius:20px;
    border:1px solid black;
}

.boton--aceptar{
    @extend .boton;
    background-color:green;
    color:yellow;
}

.boton--cancelar{
    @extend .boton;
    background-color:red;
    color:yellow;
}

```

SCSS

```

.boton, .boton--aceptar,
.boton--cancelar {
width:200px;
    height:50px;
    background-color:white;
    text-align:center;
    color:black;
    border-radius:20px;
    border:1px solid black;
}

.boton--aceptar{
    background-color:green;
    color:yellow;
}

.boton--cancelar{
    background-color:red;
    color:yellow;
}

```

CSS

Ejemplo de extend en Bootstrap

```
_popover.scss x
110 }
111 }
112
113 .bs-popover-left {
114   margin-right: $popover-arrow-height;
115
116   > .arrow {
117     right: subtract(-$popover-arrow-height, $popover-border-width);
118     width: $popover-arrow-height;
119     height: $popover-arrow-width;
120     margin: $popover-border-radius 0; // make sure the arrow does not touch the popover's rounded corners
121
122     &::before {
123       right: 0;
124       border-width: ($popover-arrow-width / 2) 0 ($popover-arrow-width / 2) $popover-arrow-height;
125       border-left-color: $popover-arrow-outer-color;
126     }
127
128     &::after {
129       right: $popover-border-width;
130       border-width: ($popover-arrow-width / 2) 0 ($popover-arrow-width / 2) $popover-arrow-height;
131       border-left-color: $popover-arrow-color;
132     }
133   }
134 }
135
136 .bs-popover-auto {
137   &[x-placement="top"] {
138     @extend .bs-popover-top;
139   }
140   &[x-placement="right"] {
141     @extend .bs-popover-right;
142   }
143   &[x-placement="bottom"] {
144     @extend .bs-popover-bottom;
145   }
146   &[x-placement="left"] {
147     @extend .bs-popover-left;
148   }
149 }
150
151
```

Mixins

Mixins

Te permiten definir estilos que pueden ser reutilizados en tu proyecto. Una de las mayores diferencias con los Extend, es que los Mixins pueden recibir argumentos, los cuales nos permitirán producir una gran variedad de estilos con unas simples líneas.

Mixins sin parámetros

```
@mixin flex-column-center{  
    display:flex;  
    flex-direction:column;  
    justify-content:center;  
}  
.box {  
    @include flex-column-center;  
}
```

SCSS

```
.box {  
    Display:flex;  
    flex-direction:row;  
    justify-content:center;  
}
```

CSS

Mixins con parámetros

```
@mixin sizes($width, $height)
{
    width: $width;
    height: $height;
}

.box {
    @include sizes(500px,
50px);
}
```

SCSS

```
.box {
    height:
50px;
    width:
500px;
}
```

CSS

Ejemplo de mixins en Bootstrap

```
1 // stylelint-disable property-blacklist
2 // Single side border-radius
3
4 @mixin border-radius($radius: $border-radius, $fallback-border-radius: false) {
5   @if $enable-rounded {
6     border-radius: $radius;
7   }
8   @else if $fallback-border-radius != false {
9     border-radius: $fallback-border-radius;
10  }
11 }
12
13 @mixin border-top-radius($radius) {
14   @if $enable-rounded {
15     border-top-left-radius: $radius;
16     border-top-right-radius: $radius;
17   }
18 }
19
20 @mixin border-right-radius($radius) {
21   @if $enable-rounded {
22     border-top-right-radius: $radius;
23     border-bottom-right-radius: $radius;
24   }
25 }
```

Mixins

Ya tenemos un poco más claro cuales son las diferencias entre estas importantes características de SASS. Recuerden que utilizaremos:

Extends

Para compartir fragmentos de estilos idénticos entre componentes.

Mixins

Para reutilizar fragmentos de estilos que puedan tener un resultado diferente en cada lugar donde los declaremos.

Operadores, condicionales y **bucles**

Operaciones

Operaciones

Con SASS puedes realizar operaciones matemáticas básicas en la misma hoja de estilo, y es tan sencillo como poner el símbolo aritmético adecuado.

[Enlace de interés](https://programacion.net/)

Fuente: <https://programacion.net/>



Operaciones

```
$ancho: 720px;
$blue: #4285F4;
$green: #33D374;

.box_uno {
    background-color: $blue;
    width: $ancho/2; /* Ancho de
360*/
}
.box_dos {
    background-color: $green;
    width: ($ancho/2)-50;
}
```

SCSS

```
.box_uno {
    background-color:
#4285f4;
    width: 360px;
}

.box_dos {
    background-color:
#33d374;
    width: 310px;
}
```

CSS

Condicionales

Condicionales

Permiten establecer reglas para validar si se aplica o no una acción, cambio o asignación en el atributo de un elemento. Estas condiciones podrán incluir comparadores típicos (`==`, `!=`, `<`, `>`) entre variables, constantes o cualquier expresión intermedia.

If: (Si condicional)

Sólo en caso de cumplirse la condición, se ejecutará la generación de código del bloque asociado.

Condicionales

```
$fiesta: navidad;
```

```
body {
```

```
    @if $fiesta == pascuas {
```

```
        background-color: pink;
```

```
    } @else if $fiesta == padre {
```

```
        background-color: blue;
```

```
    } @else if $fiesta == navidad {
```

```
        background-color: red;
```

```
    } @else {
```

```
        background-color: $bg-body;
```

```
        font-style: $fuente-titulos;
```

```
        color: $letra-clara;
```

```
    }
```

```
}
```

EJEMPLO CONDICIONAL EN BOOTSTRAP

```
109 }
110 }
111 .carousel-control-prev {
112   left: 0;
113   @if $enable-gradients {
114     background-image: linear-gradient(90deg, rgba($black, .25), rgba($black, .001));
115   }
116 }
117 .carousel-control-next {
118   right: 0;
119   @if $enable-gradients {
120     background-image: linear-gradient(270deg, rgba($black, .25), rgba($black, .001));
121   }
122 }
123 }
```

EJEMPLO CONDICIONAL EN BOOTSTRAP

```
109 }  
110 }  
111 .carousel-control-prev {  
112   left: 0;  
113   @if $enable-gradients {  
114     background-image: linear-gradient(90deg, rgba($black, .25), rgba($black, .001));  
115   }  
116 }  
117 .carousel-control-next {  
118   right: 0;  
119   @if $enable-gradients {  
120     background-image: linear-gradient(270deg, rgba($black, .25), rgba($black, .001));  
121   }  
122 }  
123 }
```

Bucles

Bucles

Un bucle es una secuencia que repite más de una vez una porción de código, dada cierta condición. Cuando la misma deja de cumplirse, el bucle finaliza.

For: (Para)

```
@for $var from [to|through] {  
  //Bloque de reglas donde podrás utilizar $var  
  mediante interpolación  
}
```

\$var será el nombre de la variable que queramos utilizar en nuestro bloque.

Tanto **<start>** como **<end>** tendrán que ser expresiones SassScript válidas, que devuelvan números enteros. Por último, si indicamos **'through'** se tendrán en cuenta los valores **<start>** y **<end>** dentro del bucle; si utilizamos **'to'**, no se tendrá en cuenta el valor **<end>** dentro del bucle.

Bucles

```
@for $i from 1 through 10 {  
  .col-#{ $i }{  
    width:10% * $i;  
  }  
}
```

SCSS

```
.col-1{width:10%;}  
.col-2{width:20%;}  
.col-3{width:30%;}  
.col-4{width:40%;}  
.col-5{width:50%;}  
.col-6{width:60%;}  
.col-7{width:70%;}  
.col-8{width:80%;}  
.col-9{width:90%;}  
.col-10{width:100%;}
```

CSS

Ejemplo de bucles en Bootstrap

```
_grid-framework.scss x
// Framework grid generation
//
// Used only by Bootstrap to generate the correct number of grid classes given
// any value of `$grid-columns`.

@mixin make-grid-columns($columns: $grid-columns, $gutter: $grid-gutter-width, $breakpoint: $grid-breakpoints) {
  // Common properties for all breakpoints
  %grid-column {
    position: relative;
    width: 100%;
    padding-right: $gutter / 2;
    padding-left: $gutter / 2;
  }

  @each $breakpoint in map-keys($breakpoints) {
    $infix: breakpoint-infix($breakpoint, $breakpoints);

    // Allow columns to stretch full width below their breakpoints
    @for $i from 1 through $columns {
      .col#{$infix}-#{$i} {
        @extend %grid-column;
      }
    }
    .col#{$infix},
    .col#{$infix}-auto {
      @extend %grid-column;
    }
  }
}
```


Each

Each

La regla `@each` facilita la emisión de estilos, o la evaluación del código para cada elemento de una lista, o cada par en un mapa.

Es ideal para estilos repetitivos que sólo tienen algunas variaciones entre ellos ya que, de cumplirse una característica, realiza dicha acción.

Each

Es posible definir una estructura @each de la siguiente manera:

```
@each $var in {  
  //Bloque de reglas donde podremos utilizar $var  
  mediante interpolación  
}
```

En este caso, <list> será cualquier expresión que devuelva una lista de elementos SassScript válida, es decir, una sucesión de elementos separados por comas.

Each

```
@each $color in blue, yellow,  
black {  
    #{$color}{  
        color:#{$color};  
    }  
}
```

SCSS

```
.blue {  
    color:blue;  
}  
  
.yellow{  
    color:yellow;  
}  
  
.black {  
    color:black;  
}
```

CSS

Ejemplo each en Bootstrap

```
40 }  
41  
42  
43 // Alternate styles  
44 //  
45 // Generate contextual modifier classes for colorizing the alert.  
46  
47 @each $color, $value in $theme-colors {  
48   .alert-#{$color} {  
49     @include alert-variant(theme-color-level($color, $alert-bg-level),  
50                           theme-color-level($color, $alert-border-level),  
51                           theme-color-level($color, $alert-color-level));  
52   }  
53 }  
54
```

Maps

Maps

Los mapas son variables cuyo valor es una colección de variables. Se definen con un nombre que los identifica. Las claves suelen ser cadenas o números, mientras que los valores pueden ser cualquier tipo de dato.

Ejemplo: suponte que se necesita crear una serie de botones para compartir contenido y te exigen tres de diferente color. Para no crearlos uno a uno, generas un mapa con clave "el botón" y valor "el color que tendrá".

```
$map: (key1: value1, key2: value2, key3: value3);
```

Par clave:valor

Maps

```
$redes: ( /*Declaramos nuestro mapa*/  
  twitter: #55acee,  
  facebook: #3a5795,  
  send-mail: #C25E30  
);  
/*Creamos el bucle para usar los valores  
del mapa*/  
@each $red, $color in $redes {  
  .btn--#{$red} {  
    background-color: $color;  
  }  
}
```

SCSS

```
.btn--twitter {  
  background-color: #55acee;  
}  
  
.btn--facebook {  
  background-color: #3a5795;  
}  
  
.btn--send-mail {  
  background-color: #C25E30;  
}
```

CSS

Maps-GET()

```
$estilos: (  
  color: #3673D9,  
  centro: center,  
  tFuente: 35px,  
  1rem : 1rem  
);  
  
div {  
  background-color: map-get($estilos, color);  
  text-align: map-get($estilos, centro);  
  padding: map-get($estilos, 1rem);  
  font-size: map-get($estilos, tFuente);  
}
```

SCSS

```
div {  
  background-color: #3673D9;  
  Text-align: center;  
  Padding: 1rem;  
  Font-size: 35px;  
}
```

CSS

Ejemplo de maps en Bootstrap

```
variables.scss
65     },
66     $colors
67   );
68
69   $primary:    $blue !default;
70   $secondary:  $gray-600 !default;
71   $success:    $green !default;
72   $info:       $cyan !default;
73   $warning:    $yellow !default;
74   $danger:     $red !default;
75   $light:      $gray-100 !default;
76   $dark:       $gray-800 !default;
77
78   $theme-colors: () !default;
79   // stylelint-disable-next-line scss/dollar-variable-default
80   $theme-colors: map-merge(
81     (
82       "primary":    $primary,
83       "secondary":  $secondary,
84       "success":    $success,
85       "info":       $info,
86       "warning":    $warning,
87       "danger":     $danger,
88       "light":      $light,
89       "dark":       $dark
90     ),
91     $theme-colors
92   );
```

```
buttons.scss
54   }
55
56
57   //
58   // Alternate buttons
59   //
60
61   @each $color, $value in $theme-colors {
62     .btn-#{$color} {
63       @include button-variant($value, $value);
64     }
65   }
66
67   @each $color, $value in $theme-colors {
68     .btn-outline-#{$color} {
69       @include button-outline-variant($value);
70     }
71   }
72 }
```



Break

¡10 minutos y volvemos!



Ejemplo en vivo

¡Vamos a practicar lo visto!

¿Preguntas?



Tercera pre-entrega

En la clase que viene se presentará la segunda parte del Proyecto Final, que **nuclea temas vistos entre las clases 12 y 15**.
Recuerda que tendrás 7 días para subirla en la plataforma.



Encuesta

Por encuestas de Zoom

¡Terminamos el **módulo 7: Preprocesador!**

Cuéntanos qué temas te resultaron más complejos de entender. **Puedes elegir más de uno.** Vamos a retomar aquellos temas que resultaron de mayor dificultad en el próximo AfterClass.

Resumen de la clase hoy

- ✓ Aplicación de operaciones y decisiones con SASS.

Opina y valora
esta clase

Muchas gracias.

#DemocratizandoLaEducación