

***RECUERDA PONER A GRABAR LA
CLASE***





¿DUDAS DEL ON-BOARDING?

MIRALO AQUI



Clase 12. DESARROLLO WEB

SASS II



OBJETIVOS DE LA CLASE

- Agregar operaciones y decisiones con SASS.

GLOSARIO:

Clase 11

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.
- **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.

GLOSARIO:

Clase 11

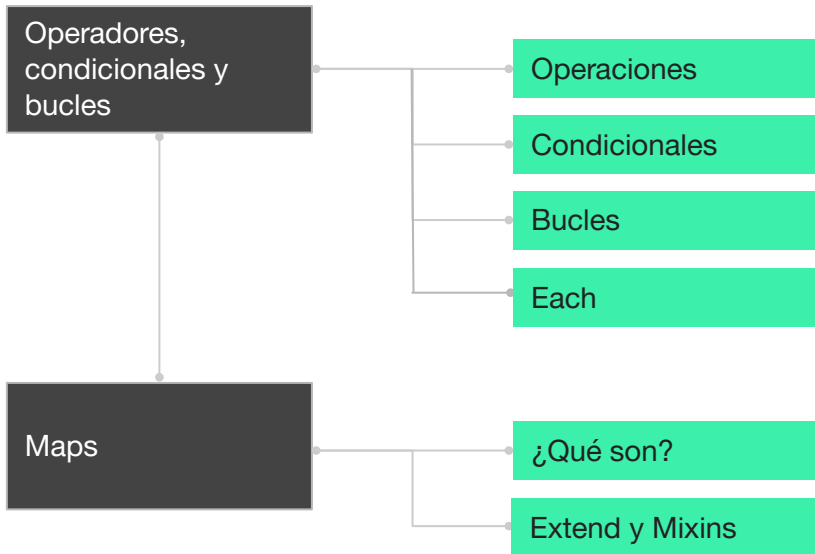
BEM: significa Modificador de Bloques de Elementos (Block Element Modifier) por sus siglas en inglés. Sugiere una manera estructurada de nombrar tus clases, basada en las propiedades del elemento en cuestión.

- **Bloque:** es un contenedor o contexto donde el elemento se encuentra presente.
- **Modificador:** permite modificar el estilo de un elemento específico.
- **Elemento:** es una de las piezas que compondrán la estructura de un bloque. El bloque es el todo, y los elementos son las piezas de este bloque.

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS CLASE 12

¡Para
recordar!



CRONOGRAMA DEL CURSO

Clase 11



SASS I



PRÁCTICAS DE LO
VISTO EN CLASE



APLICANDO SASS

Clase 12



SASS II



PRÁCTICAS DE LO
VISTO EN CLASE



APLICANDO SASS -
OPERACIONES



TERCERA ENTREGA
DEL PROYECTO FINAL

Clase 13



Servidores, Seguridad y SEO para tu sitio



APLICAR SEO A NUESTRA
PÁGINA



GUIÓN DE LA CLASE

Accede al material complementario [aquí](#).

***VAMOS A INICIAR EL
PROCESADOR CSS***

OPERADORES, CONDICIONALES Y BUCLES

OPERACIONES



[Enlace de interés](#)

OPERACIONES

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

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

```
$animal: gato; /* Mi constante */  
p {  
    @if $animal == gato {  
        color: blue;  
    } @else if $animal == perro {  
        color: red;  
    } @else if $animal == caballo {  
        color: green;  
    } @else {  
        color: black;  
    }  
}
```

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
```

BUGLES

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.

BUCLÉS

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.

BUCLÉS

```
@for $i from 1 through 3 {  
  .casitodos-#{ $i } { width: 2em * $i; }  
}
```

SCSS

```
.casitodos-1 {  
  width: 2em;  
}
```

```
.casitodos-2 {  
  width: 4em;  
}
```

```
.casitodos-3 {  
  width: 6em;  
}
```

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, $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;
    }
  }
}
```


Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE

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 $animal in puma, sea-slug, egret {  
  .#{$animal}-icon {  
    Background-image:  
    url('/images/#{$animal}.png');  
  }  
} /*animal tendrá los valores de cada uno */
```

SCSS

```
.puma-icon {  
  background-image: url("/images/puma.png");  
}  
  
.sea-slug-icon {  
  background-image: url("/images/sea-slug.png");  
}  
  
.egret-icon {  
  background-image: url("/images/egret.png");  
}
```

CSS

EJEMPLO EACH EN BOOTSTRAP

```
40 }
41
42
43 // Alternate styles
44 //
45 // Generate contextual modifier classes for coloring 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
```

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE

MAPS

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

EJEMPLO DE MAPS EN BOOTSTRAP

```
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 );
93
```

```
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
```

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE

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

```
.redsocial {  
  border: 1px #f00;  
  background-color: #fdd;  
}  
  
.redsocial--nueva {  
  @extend .redsocial;  
  border-width: 3px;  
}
```

SCSS

```
.redsocial, .redsocial--nueva {  
  border: 1px #f00;  
  background-color: #fdd;  
}  
  
.redsocial--nueva {  
  border-width: 3px;  
}
```

CSS

EJEMPLO DE EXTEND EN BOOTSTRAP

```
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

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.

MIXINS

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

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE



APLICANDO SASS - OPERACIONES

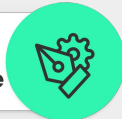
Selecciona tres o más bloques HTML de alguna de tus páginas del desafío que aún no tengan styleadas en CSS, y genera el código necesario en SASS para darle estilo.

APLICANDO SASS - OPERACIONES

Formato: archivo HTML y CSS. Debe tener el nombre "Idea+Apellido".

Sugerencia: carpeta en formato zip o rar, con el/los archivos HTML, CSS y SCSS.

Desafío
entregable



>> Consigna: agrega SASS y sus operaciones a tu Proyecto Final.

>>Aspectos a incluir en el entregable:

Selecciona tres o más bloques de alguna de tus páginas que aún no tengan estilos, y genera el código necesario en SCSS.

>>Ejemplo:

[Carpeta comprimida con SASS](#)



APLICANDO SASS II

APLICANDO SASS II

Formato: Archivo html y css

Sugerencia: carpeta en formato zip o rar con el/los archivos html y CSS.

Desafío
Complementario



>> Consigna:

Escoge un SCSS de Bootstrap como base para generar tu CSS, editando alguno de los bloques de estilo que se adapte a tu proyecto. Modifica los:

1. Bucle
2. Each
3. Mapas



TERCERA ENTREGA DEL PROYECTO FINAL

Deberás entregar **la estructura avanzada y el estilo avanzado de la web, con la adaptación al framework y las animaciones, transformaciones y transiciones**, correspondientes a la tercera entrega de tu proyecto final.

TERCERA ENTREGA DEL PROYECTO FINAL

Formato: carpeta comprimida con archivos del proyecto. Incluir apellido en el nombre del archivo (ej. "Nombre Proyecto - Apellido").

Sugerencia: activar comentarios en el archivo.

Proyecto
Final



3

Estructura avanzada de la web

>>Objetivos generales:

1. Realizar una estructura del HTML prolija, limpia, fácil de leer y que no tenga errores en sus atributos o en sus valores.

>>Objetivos específicos:

1. Agregar elementos HTML según la necesidad de armar contenedores o elementos web determinados, en base al framework elegido y la documentación del mismo.
2. Agregar transformaciones, animaciones y/o transiciones para otorgarle dinamismo a la web en elementos que tengan interacción con el usuario.

>>Se debe entregar:

- Maquetado de la web: las estructuras maquetan a la web en base al framework elegido, haciendo usos de clases utilitarias para armar grillas, elementos web y estilos propios del framework, además del HTML de contenido.
- Páginas: todas las páginas tienen el contenido estructurado y el estilo linkeado. También tiene que tener agregadas las diferentes librerías de Javascript y CSS pertinentes al framework.

CODER HOUSE



Estilo avanzado de la web

Formato: archivo CSS

>>Objetivos generales:

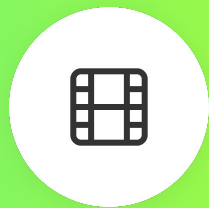
1. Crear archivos de CSS para darle estilo a la web.

>>Objetivos específicos:

1. Agregar transformaciones, animaciones y/o transiciones para otorgarle dinamismo a la web en elementos que tengan interacción con el usuario.
2. Hacer uso de selectores de CSS para poder darle estilo propio a los elementos que ya vienen con su propia identidad del framework.

>>Se debe entregar:

- Estilo avanzado: se le mejorarán los elementos interactivos con variaciones en sus diferentes estados, ya sea de la mano de transformaciones, transiciones y/o animaciones.
- Estilo del Framework: no todos los elementos del framework van a tener una estética que condice con el sitio en el que son implementados, por lo que se usará CSS para darles un estilo acorde.
- Estructura de la web: usa etiquetas no sólo para armar contenido, sino para armar los elementos que van a conformar el layout de la web, los contenedores, etc.



***¿QUIERES SABER MÁS? TE DEJAMOS
MATERIAL AMPLIADO DE LA CLASE***



- [Más información sobre BEM](#) | **BEM 101**
- [Más información sobre OOCSS](#) | **Smashing Magazine**

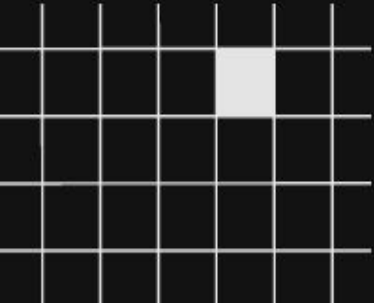
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Aplicación de operaciones y decisiones con SASS.
- 



OPINA Y VALORA ESTA CLASE