

UT4: Usabilidad.

Preprocesadores CSS. SASS

The Sass logo is displayed in a stylized, cursive script. It is contained within a white rectangular area that is framed by a thick magenta border. The word 'Sass' is written in a fluid, handwritten style with a magenta color.

Introducción

Un preprocesador de CSS es un programa que te permite generar CSS y agregar algunas características que no existen en el CSS puro, como variables (sin tener que usar funciones para acceder a ellas), mixins (conjunto de propiedades reutilizables), selectores anidados, herencia de selectores, etc...

Estas características hacen que la estructura del CSS sea más legible y fácil de mantener.



.sass

```
.SCSS .sass
$blue: #3bbfce
$margin: 16px

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

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

.SCSS

```
.SCSS .sass
$blue: #3bbfce;
$margin: 16px;

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

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

.CSS













```
/* CSS */

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

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

Instalación

Distinguiremos principalmente dos, como **aplicación independiente** o con un programa de gestión de paquetes.

 dart-sass-1.83.4-android-arm.tar.gz	3.78 MB	last week
 dart-sass-1.83.4-android-arm64.tar.gz	3.92 MB	last week
 dart-sass-1.83.4-android-ia32.tar.gz	17.9 MB	last week
 dart-sass-1.83.4-android-riscv64.tar.gz	4.94 MB	last week
 dart-sass-1.83.4-android-x64.tar.gz	4.05 MB	last week
 dart-sass-1.83.4-linux-arm-musl.tar.gz	4.01 MB	last week
 dart-sass-1.83.4-linux-arm.tar.gz	3.88 MB	last week
 dart-sass-1.83.4-linux-arm64-musl.tar.gz	4.15 MB	last week
 dart-sass-1.83.4-linux-arm64.tar.gz	4.02 MB	last week
 dart-sass-1.83.4-linux-ia32-musl.tar.gz	18.2 MB	last week
 Source code (zip)		last week
 Source code (tar.gz)		last week

5

Instalación

Con un programa de gestión de paquetes.

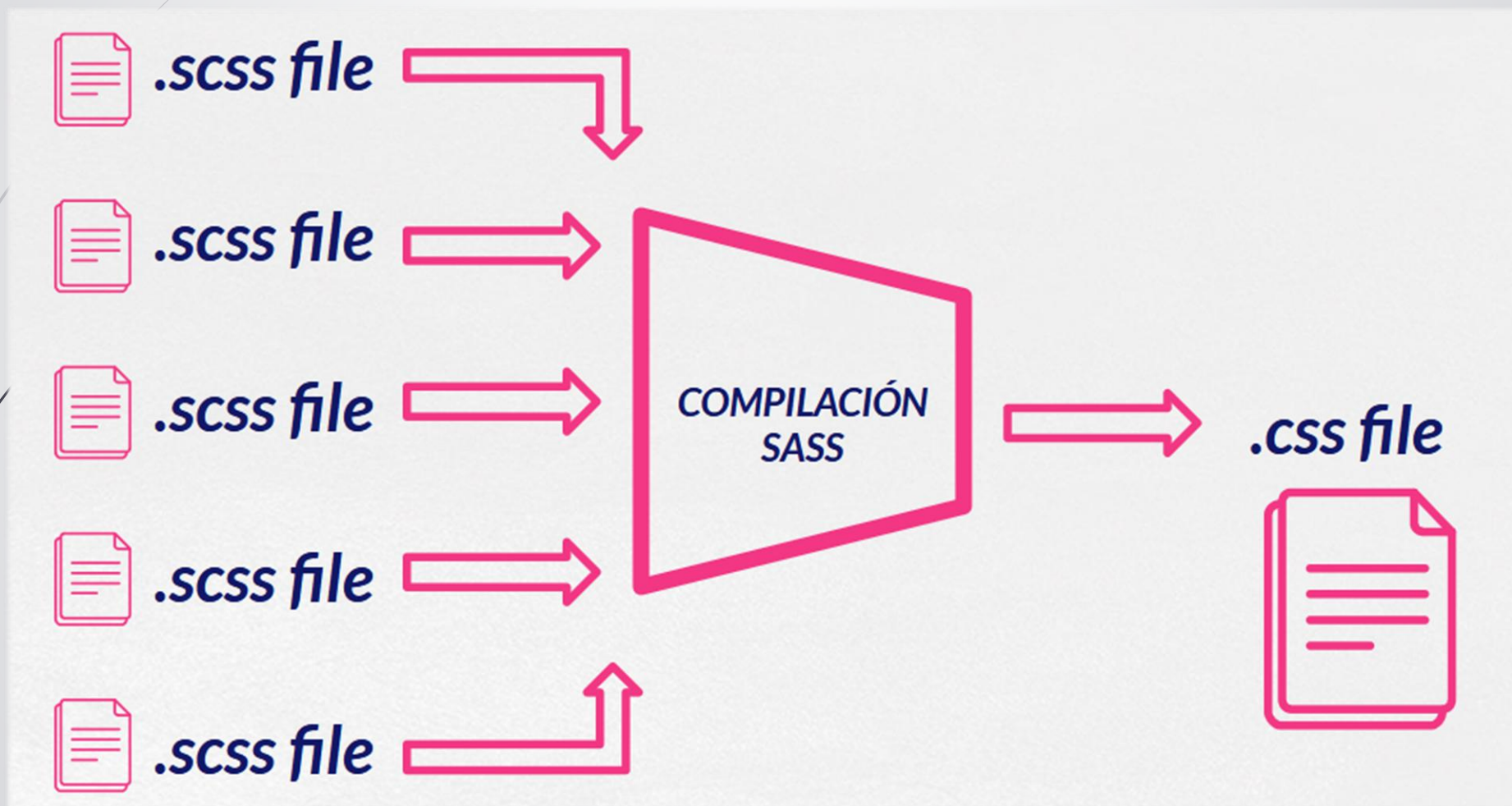


```
node -v
```

```
npm install -g sass
```

```
sass --version
```


Proceso general



Elementos básicos

► Recomendaciones

En todo proyecto profesional debemos de seguir una serie de reglas y convenciones para que el esfuerzo realizado sea reusable, escalable y mantenible.

Vamos a ver los siguientes elementos básicos:

- Variables.
- Comentarios.
- Listas y mapas.
- Interpolación.
- Anidamiento

Sass Guidelines

Elementos básicos

► Variables

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

En CSS existen las variables. La ventaja de las variables CSS vs las de SASS son:

- Se pueden cambiar en tiempo de ejecución desde JavaScript
- Pueden existir a nivel local dentro de los selectores.

```
/* Definición de la variable */
:root {
  --color-alternativo-5: #0056b8;
}

/* Uso de la variable */
.c-button {
  background-color: var(--color-alternativo-5)
}
```


Elementos básicos

► Variables

En SASS definimos una variable usando el símbolo del dólar, seguido de los dos puntos y el valor o expresión que le queremos a asignar.

Los tipos de datos básicos pueden ser:

- Numéricos: 12 12px 12em
- Cadenas: "left" 'left' left
- Colores: rgb(255, 0, 0) hsl(0, 100%, 50%) #f00 #ff0000
- Booleanos
- Null
- Mapas y Listas.

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

Elementos básicos

► Variables

Normalmente, cuando asignas un valor a una variable, si esa variable ya tenía un valor, el valor anterior se sobrescribe. Sin embargo, si estás escribiendo una biblioteca de Sass, es posible que desees permitir que los usuarios configuren las variables de tu biblioteca antes de usarlas para generar CSS.

Para hacer esto posible, Sass proporciona la bandera **!default**.

Esta asigna un valor a una variable sólo si esa variable no está definida o si su valor es nulo. De lo contrario, se usará el valor existente.

Elementos básicos

► Variables

El ámbito de las variables puede ser global, si se define fuera de todo bloque, o local:

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

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

Elementos básicos

➤ Comentarios

➤ Comentarios de una sola línea

```
// Comentario en una única línea
```

➤ Comentarios multilínea

```
/*  
Comentario  
multilínea  
*/
```

Elementos básicos

► 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

```
$variable_lista: (v1, v2, v3);  
  
$sizes: (40px, 80px, 160px);  
  
$sizes: (  
  40px,  
  80px,  
  160px,  
);
```

```
$nombre_mapa: (  
  "clave1": valor1,  
  ...  
  "claven": valorn  
);  
  
$breakpoint: (  
  'pequeño' : 576px,  
  'medio'   : 768px,  
  'grande'  : 992px  
);
```


Elementos básicos

► Listas y Mapas

- `append($list, $val, $separator: auto)`: Devuelve una copia de *\$list* con *\$val* añadido al final de la misma.
- `index($list, $value)`: Devuelve el índice de *\$value* si está en la lista o *null* en caso contrario.
- `is-bracketed($list)`: devuelve *true* si la lista tiene corchetes o *false* en caso contrario.
- `join($list1, $list2, $separator: auto, $bracketed: auto)`: devuelve una nueva lista con los valores de *\$list1* seguido de los valores de *\$list2*, con el *\$separator* indicado y entre corchetes si *\$bracketed* es *true*.
- `length($list)`: devuelve la longitud de la lista.
- `list-separator($list)`: devuelve el separador utilizado (espacio por defecto o coma).
- `nth($list, $n)`: devuelve el elemento con índice *\$n* de *\$list*, contando desde el final si *\$n* es negativo y devolviendo un error si no existe el índice.
- `set-nth($list, $n, $value)`: Devuelve una copia de *\$list* con el elemento de índice *\$n* reemplazado por *\$value*, , contando desde el final si *\$n* es negativo y devolviendo un error si no existe el índice.
- `zip($lists...)`: Combina cada lista en *\$lists* en una única lista de sub-listas de longitud la más corta de las sub-listas y separada por comas.

sass:list

Elementos básicos

► Listas y Mapas

- `keywords($args)`: Devuelve un mapa con las keywords pasadas a un *mixin* o función con argumentos opcionales (estando estos en forma de lista de argumentos)
- `map-get($map, $key)`: Devuelve el valor en `$map` asociado a `$key`, o *null* si no lo encuentra
- `map-has-key($map, $key)`: Devuelve el booleano *true* si `$map` tiene un valor asociado a `$key`, *false* en caso contrario.
- `map-keys($map)`: Devuelve una lista separada por comas con todas las claves en `$map`.
- `map-merge($map1, $map2)`: Devuelve un nuevo mapa con todas las claves y valores de `$map1` y `$map2`. También se puede usar para añadir sobre-escribir valores en `$map1`. Si los dos mapas tienen la misma clave, prevalece la de `$map2`.
- `map-remove($map, $keys...)`: Devuelve una copia de `$map` sin los valores asociados de `$keys` (si alguna de las claves de `$keys` no existe, será ignorada).
- `map-values($map)`: Devuelve una lista separada por comas con todos los valores en `$map`.

sass:map

Elementos básicos

► 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 parte del código CSS final.

Se pueden insertar en:

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

```
#{expresión}
```

Elementos básicos

► Interpolación

Ejemplo:

```
// Interpolación en selectores
$button-type: "error";
$btn-color : ■ #f00;

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

Elementos básicos

► Anidamiento

El anidamiento en Sass permite escribir estilos de manera más estructurada y de una forma jerárquica, reflejando la estructura del HTML en la hoja de estilos.

Generamos un código más limpio y fácil de mantener al evitar la repetición de selectores.

```
a {  
  color: #f39c12;  
}  
a:hover {  
  color: #12f325;  
}
```

```
a {  
  color: #f39c12;  
  &:hover {  
    color: #12f325;  
  }  
}
```


Elementos básicos

➡ Anidamiento

```
nav {  
  background-color: ■#222;  
  color: □white;  
  
  ul {  
    list-style: none;  
    padding: 0;  
  
    li {  
      display: inline-block;  
      margin-right: 15px;  
  
      a {  
        text-decoration: none;  
        color: □white;  
  
        &:hover {  
          color: ■#f39c12;  
        }  
      }  
    }  
  }  
}
```

```
nav {  
  background-color: ■#222;  
  color: □white;  
}  
  
nav ul {  
  list-style: none;  
  padding: 0;  
}  
  
nav ul li {  
  display: inline-block;  
  margin-right: 15px;  
}  
  
nav ul li a {  
  text-decoration: none;  
  color: □white;  
}  
  
nav ul li a:hover {  
  color: ■#f39c12;  
}
```

Elementos básicos

► Compilación

La manera más sencilla sería usar el siguiente comando:.

```
sass file.scss salida.css
```



Live Sass Compiler

Glenn Marks |  2,142,463 |  (59)

Compile Sass or Scss to CSS at realtime.

Uninstall



Auto Update



Elementos básicos

► Compilación

Múltiple:

```
sass file1.scss:salida1.css file2.scss:salida2.css .....
```

Comprimida: quita la mayor cantidad de caracteres posibles

```
sass --style=compressed file.scss salida.min.css
```

Vigilando los cambios y actualizando ficheros:

```
sass --watch file.scss salida.css
```

Elementos básicos

► Compilación

Podemos decidir, dependiendo de la importancia, si queremos que nuestros **comentarios** en ficheros Sass se incluyan o no en los CSS generados.

// Este tipo de comentarios **no se incluyen** en CSS

/* Este comentario **se incluye, salvo en modo COMPRESSED** */

/*! Este comentario **también en MODOCOMPRESSED** */

Elementos básicos

► Estructuras de control

Sass incorpora estructuras típicas de lenguajes de programación que nos van a permitir desarrollar CSS de una manera más óptima, más organizada y reusable.

@if / @else

@while

@for

@each

Elementos básicos

➡ Estructuras de control

➡ Sintaxis if

```
// Ejemplo de @if en SASS
$tema-oscuro: true;

.botonos {
  @if $tema-oscuro {
    background-color: ■ #333;
    color: □ #fff;
  } @else {
    background-color: □ #fff;
    color: ■ #333;
  }
}
```

Elementos básicos

➡ Estructuras de control

➡ Sintaxis while

```
$font-size: 16;

@while $font-size <= 24 {

    .font-size-#{ $font-size } {
        font-size: #{ $font-size }px;
        line-height: #{ $font-size + 2 }px;
    }

    // Increment the counter
    $font-size: $font-size + 2;
}
```

```
.font-size-16 {
    font-size: 16px;
    line-height: 18px;
}

.font-size-18 {
    font-size: 18px;
    line-height: 20px;
}

.font-size-20 {
    font-size: 20px;
    line-height: 22px;
}

.font-size-22 {
    font-size: 22px;
    line-height: 24px;
}

.font-size-24 {
    font-size: 24px;
    line-height: 26px;
}
```

Elementos básicos

➡ Estructuras de control

➡ Sintaxis for

Si usamos **to** en lugar de **through** excluiríamos la última iteración del bucle.

```
$font-size: (24px, 22px, 20px, 18px);

@for $i from 1 through 4 {
  h#{$i} {
    font-size: nth($font-size, $i);
  }
}
```

```
h1 {
  font-size: 24px;
}

h2 {
  font-size: 22px;
}

h3 {
  font-size: 20px;
}

h4 {
  font-size: 18px;
}
```

Elementos básicos

➔ Estructuras de control

➔ Sintaxis each

Facilita la iteración sobre listas y mapas. Cada bucle comenzará en el primer elemento de la lista o mapa y continuará hasta llegar al último elemento. (similar al foreach en Java o PHP)

```
$font-sizes: (16, 18, 20, 22);

@each $size in $font-sizes {

    .font-size-#{ $size } {
        font-size: #{ $size }px;
        line-height: #{ $size + 2 }px;
    }
}
```

```
.font-size-16 {
    font-size: 16px;
    line-height: 18px;
}

.font-size-18 {
    font-size: 18px;
    line-height: 20px;
}

.font-size-20 {
    font-size: 20px;
    line-height: 22px;
}

.font-size-22 {
    font-size: 22px;
    line-height: 24px;
}
```

Elementos básicos

► Funciones

Además de todas estas estructuras de control tengo la posibilidad de usar funciones:

Nativas (propias de Sass)

- Numéricas
- Cadenas (Strings)
- Colores
- Listas
- Mapas
- Selectores
- Introspection

```
sass:color  
sass:list  
sass:map  
sass:math  
sass:meta  
sass:selector  
sass:string
```


Elementos básicos

► Funciones

Además de todas estas estructuras de control tengo la posibilidad de usar funciones:

Definidas por el usuario

```
@function name() {  
    //....  
    @return ...;  
}  
  
@function name($argumentos) {  
    //...  
    @return ...;  
}
```

Elementos básicos

► Implementa

Error Warning Accepted Normal

Tabla

Nombre	Apellidos	Dirección	Email
Pepe	Pérez	Aquí	pepe@deaqui.es
Manuel	López	Allí	manuel@dealli.es
Ana	Martínez	Lejos	ana@delejos.es
Lola	Fernández	Cerca	lola@decerca.es

Sistema de columnas

--	--	--	--