

Sass

EN 5 MINUTOS

Guía rápida para
usar SASS



CodingTube

Índice

| | |
|--|----|
| Requisitos para poder usar SASS | 4 |
| Node.js | 4 |
| Instalar SASS | 5 |
| Empezando a usar SASS | 6 |
| Diferencia entre SASS Y SCSS | 7 |
| Generando la hoja de estilos CSS | 7 |
| Superpoderes para CSS | 9 |
| Variables | 9 |
| Nesting | 10 |
| El selector & | 11 |
| Anidamiento en las propiedades | 12 |
| Partials | 12 |
| Modules | 13 |
| Mixins | 13 |
| Uso de parámetros | 14 |
| Extend | 15 |
| Operators | 17 |
| Condicionales y bucles | 18 |

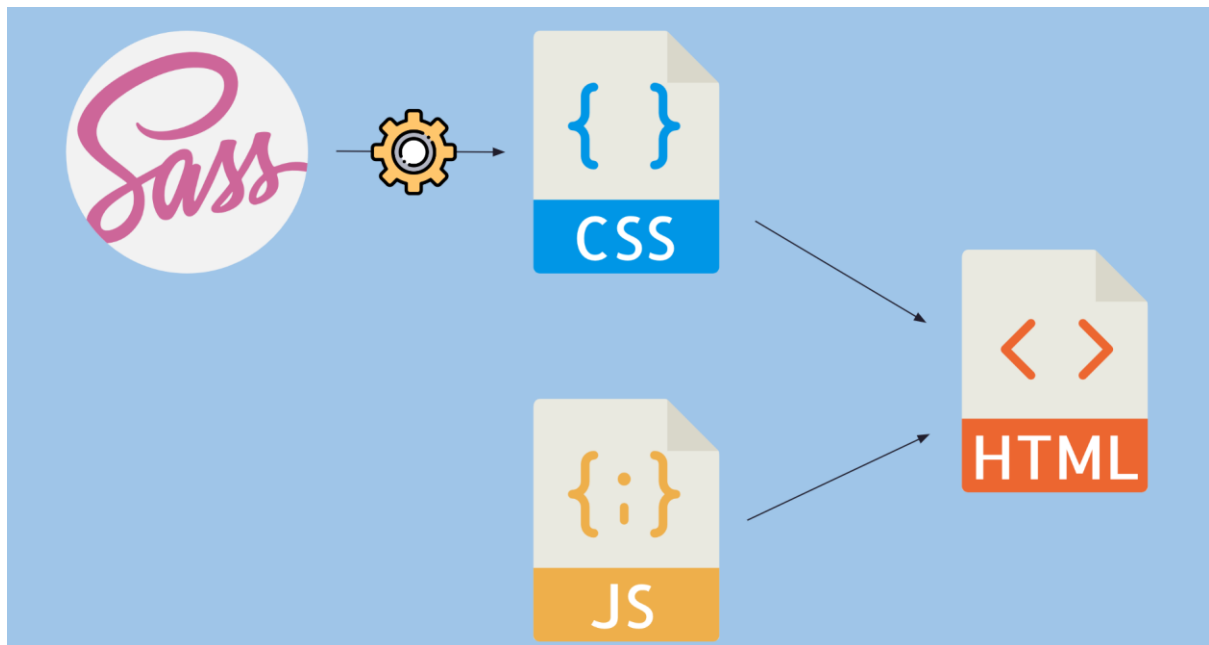
GUÍA RÁPIDA PARA USAR SASS

SASS se define a sí mismo como Superpoderes para CSS, pero ¿cuáles son estos superpoderes? En esta guía los vamos a conocer uno por uno.

⚠ Esta guía está dirigida para aquellas personas que ya conocen el funcionamiento básico de CSS y su uso dentro del desarrollo web. Si crees que necesitas afianzar tus conocimientos dentro de CSS lo puedes hacer haciendo clic [aquí](#)

SASS es un preprocesador

El hecho de que SASS sea un preprocesador significa que genera un archivo CSS a través de un proceso que se llama compilación. Este archivo CSS generado lo podemos enlazar a nuestro documento HTML.



Requisitos para poder usar SASS

1. Node.js

Para poder usar SASS en nuestro computador debemos tener instalado node.js primero. El proceso de instalación es bastante sencillo y lo podemos hacer desde la siguiente página. <https://nodejs.org/es/>



Node.js® es un entorno de ejecución para JavaScript construido con V8, motor de JavaScript de Chrome.

Descargar para Windows (x64)

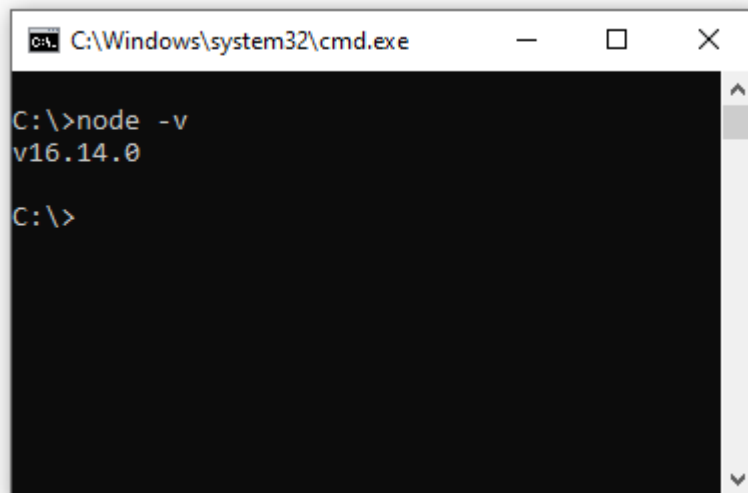
| 16.16.0 LTS | 18.7.0 Actual |
|-----------------------------|-------------------------|
| Recomendado para la mayoría | Últimas características |

Otras Descargas | Cambios | Documentación de la API

O eche un vistazo al [Programa de soporte a largo plazo \(LTS\)](#)

La versión recomendable para descargar es la versión LTS (Long Term Support). Una vez que los tengas descargado simplemente abres el instalador y dejas que el asistente te guíe a través del proceso de instalación.

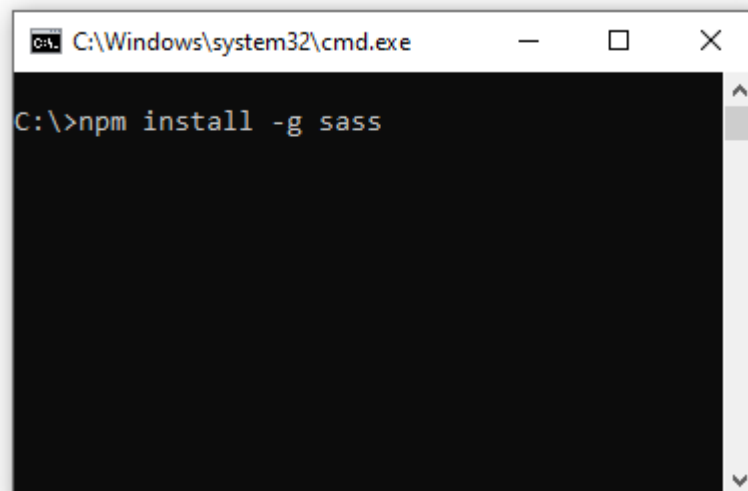
Cuando ya lo tengas instalado podrás comprobar su correcta instalación abriendo una terminal y escribiendo en ella el comando **node -v**. Si todo está bien deberías tener como resultado la versión que hayas instalado, algo parecido a la siguiente captura:

A screenshot of a Windows command prompt window. The title bar reads 'C:\Windows\system32\cmd.exe'. The command prompt shows the command 'node -v' being entered and executed, resulting in the output 'v16.14.0'. The prompt is currently at 'C:\>'.

```
C:\Windows\system32\cmd.exe
C:\>node -v
v16.14.0
C:\>
```

2. Instalar SASS

Ahora que ya tienes instalado Node, ya podrás instalar SASS en tu computador; en la misma terminal que tienes abierta lo puedes hacer escribiendo la siguiente línea de código:

A screenshot of a Windows command prompt window. The title bar reads 'C:\Windows\system32\cmd.exe'. The command prompt shows the command 'npm install -g sass' being entered. The prompt is currently at 'C:\>'.

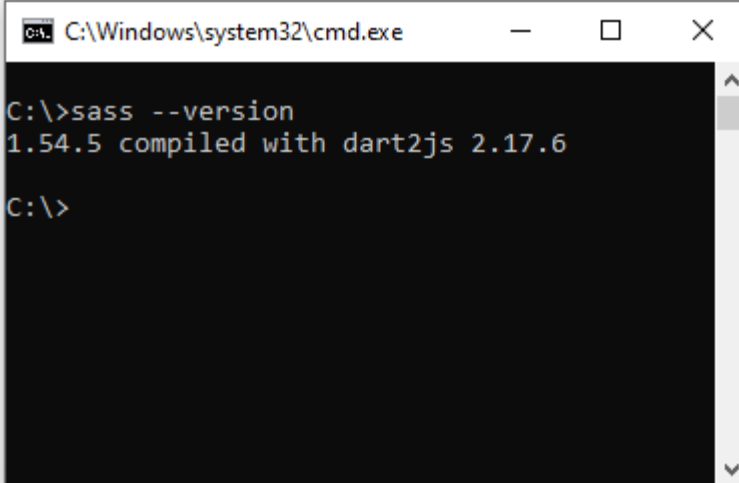
```
C:\Windows\system32\cmd.exe
C:\>npm install -g sass
```

npm (node package manager) es el gestor de paquetes para node que acabamos de instalar.

-g es una bandera que hará que SASS se instale de forma global en tu computador, no solamente en el directorio en el que estés trabajando.

⚠ Si estás usando Linux o Mac el comando que debes utilizar para la instalación es el siguiente: **brew install sass/sass/sass**

Para corroborar que la instalación se ha realizado correctamente podemos correr el siguiente comando en la terminal:



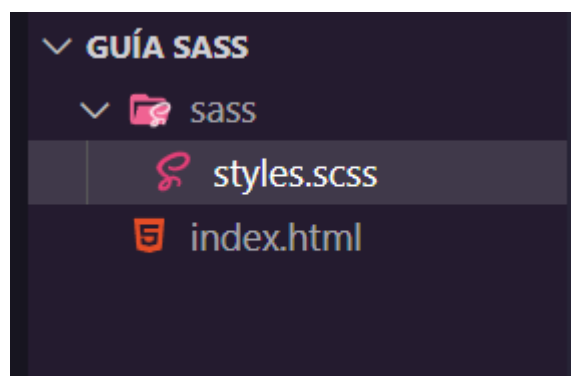
```
C:\Windows\system32\cmd.exe

C:\>sass --version
1.54.5 compiled with dart2js 2.17.6

C:\>
```

Empezando a usar SASS

Una vez que se tiene instalado Node.js y SASS en nuestro sistema ya podemos usarlo. Para esto nos vamos al repositorio que contiene nuestro proyecto y crearemos una carpeta donde pondremos todos nuestros archivos **.scss**. El primer archivo que crearemos será el archivo `styles.scss` (puede llevar el nombre que mejor creas conveniente)



Diferencia entre SASS Y SCSS

Aquí hay que hacer una aclaración. SASS tiene dos tipos de sintaxis:

.sass donde no se usa puntos y coma (;) al final de cada regla ni tampoco usa llaves, en vez de esto utiliza indentación para identificar elementos dentro de otros elementos

.scss por otro lado tiene una sintaxis parecida a la de CSS, es decir usa llaves y puntos y coma, y ésta sintaxis la usaremos en ésta guía.

SASS

```
html
  box-sizing: border-box
  font-size: 100%
```

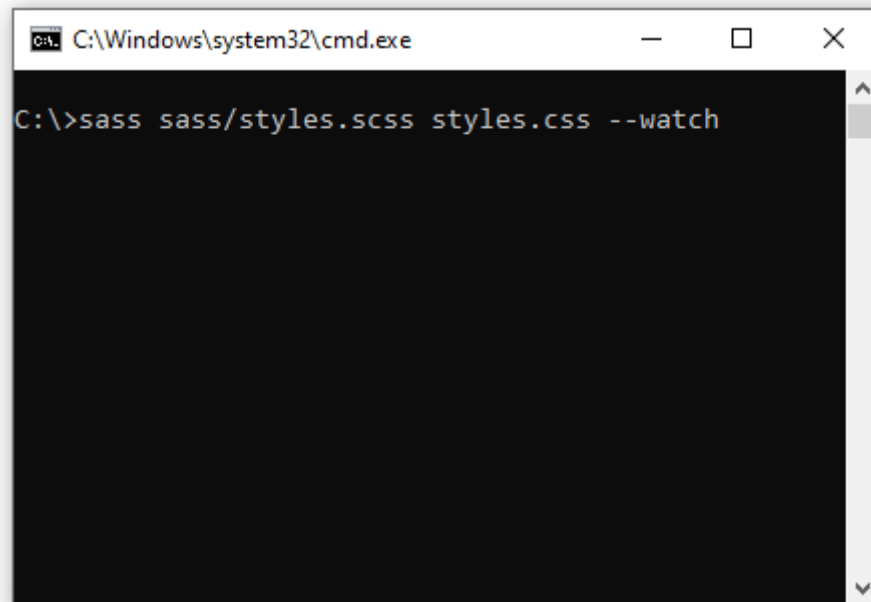
SCSS

```
html{
  box-sizing:
border-box;
  font-size: 100%;
}
```

Y por este motivo le ponemos a nuestro archivo *styles* la extensión **scss**.

Generando la hoja de estilos CSS

Tenemos que recordar que el archivo que vamos a enlazar a nuestro documento HTML no es el archivo *styles.scss*, si no que necesitamos un archivo con extensión **.css**. Éste debemos generarlo a partir de nuestro archivo *styles.scss* de la siguiente manera:



sass es el comando principal que generará la hoja de estilos

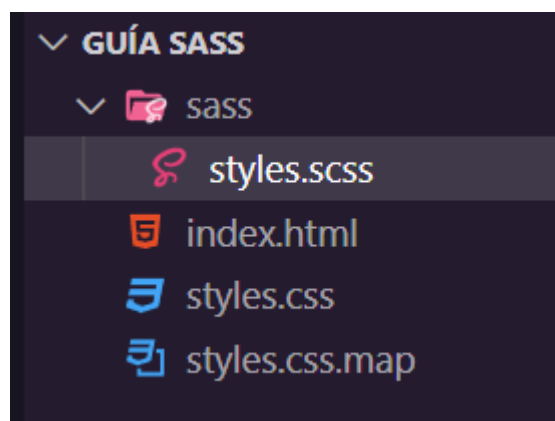
sass/styles.scss es el directorio donde se encuentra nuestro archivo sass

styles.css será nuestra hoja de estilos a generarse

--watch hace que sass constantemente esté mirando por cambios dentro del archivo sass para compilarlo dentro de nuestra hoja de estilos css

El comando de la captura va a generar la hoja de estilos css que necesitamos enlazar a nuestro documento HTML.

Si regresamos a ver nuestra estructura de archivos vamos a ver que la hoja de estilos *styles.css* ya se ha generado junto con un *styles.css.map*



Superpoderes para CSS

A continuación, repasamos los “superpoderes” que SASS otorga a CSS más utilizados:

1. Variables

Las variables actualmente no es algo nuevo para CSS, pero cuando SASS recién salió, sí que lo era.

Las variables nos permiten almacenar información que la podamos reutilizar en nuestro código. Información como tipos de fuente, colores, medidas, etc.

Esto es particularmente importante cuando estás trabajando en un proyecto extenso y es necesario cambiar, por ejemplo, el color principal utilizado en una página, por otro nuevo color. Hacer esto sin variables implica buscar por todo el código el lugar donde hemos aplicado el color inicial y cambiarlo manualmente. Pero si usamos variables, simplemente con hacer el cambio en la declaración de la variable, esta actualización se verá reflejada en todo el código, siendo esto mucho más eficiente y productivo a la hora de hacer cambios de última hora.

Actualmente CSS también goza de variables, pero de momento es lo único que comparten en común con SASS*. La forma de declarar variables en SASS y CSS es ligeramente diferente.

SCSS

```
$body-font: Helvetica, sans-serif;
$main-color: #d2d2d2d2;

body{
  font-family: $body-font;
  background-color: $main-color;
}
```

* Se está trabajando en Nesting para CSS pero de momento está en su fase de beta.

CSS generado

```
body {  
  font-family: Helvetica, sans-serif;  
  background-color: #d2d2d2;  
}
```

Nótese que en el CSS generado no se ven las variables creadas, directamente actualiza el color al valor de la variable. Cuando se necesite actualizar el valor de la variable siempre se deberá hacer desde el archivo SCSS.

2. Nesting

El anidamiento (nesting) en SASS nos permite organizar nuestro código para una mejor lectura y mayor organización de los elementos al igual que la estructura HTML de la página.

Con nesting podemos poner elementos dentro de otros elementos para asignarles estilos. Hay que tener en cuenta que una estructura demasiado anidada es considerada mala práctica ya que generaría un código difícil de mantener. El uso del nesting sería de la siguiente manera:

SCSS

```
nav {  
  ul {  
    margin: 0;  
    padding: 0;  
    list-style: none;  
  }  
  
  li { display: inline-block; }  
  
  a {  
    display: block;  
    padding: 6px 12px;  
    text-decoration: none;  
  }  
}
```

CSS generado

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

Sin nesting en CSS tendríamos que usar reglas de estilos separadas para cada componente.

El selector &

El nesting también se lo puede utilizar usando el selector especial **&** el mismo que representa el elemento actual.

```
.button {
  background-color: red;
  color: white;
  border-radius: 4px;
  width: 100px;
  height: 35px;
  padding: 8px;
  text-align: center;
  &:hover{
    border: 1px solid red;
    background-color: white;
    color: red;
  }
}
```

En este caso podemos asignarle estilos al estado hover del botón sin la necesidad de volverlo a llamar. En vez de esto utilizamos el selector **&** que en este caso representa la clase *.button*.

Anidamiento en las propiedades

Otra forma en la que podemos utilizar el anidamiento es con las propiedades de CSS, como por ejemplo el siguiente código:

SCSS

```
body {  
  background: {  
    color: lightblue;  
    image: url('./images/logo.png');  
    repeat: no-repeat;  
    position: 10px 30px;  
    size: contain;  
  }  
}
```

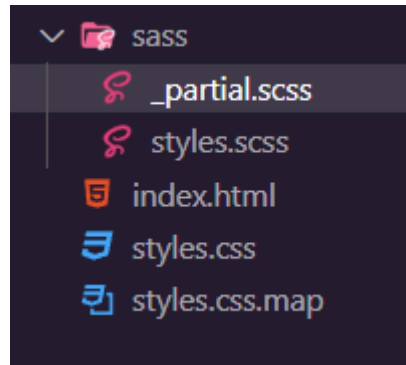
CSS generado

```
body {  
  background-color: lightblue;  
  background-image: url("./images/logo.png");  
  background-repeat: no-repeat;  
  background-position: 10px 30px;  
  background-size: contain;  
}
```

3. Partials

Los Partials nos permiten crear archivos SASS parciales que contengan fragmentos de código y que se pueden incluir en otros archivos SASS.

Ésta es una excelente manera de modularizar tus hojas de estilos y ayudar a que las cosas sean más fáciles de mantener. Un Partial es un archivo SASS nombrado con un guión bajo al principio. Como por ejemplo: “_parcial.scss”.



El guion bajo le permite a SASS saber que el archivo es solo un archivo parcial y que no debe generarse en un archivo CSS. Los parciales de SASS se usan dentro del archivo principal con la regla con la regla **@use** siempre al inicio de todo el código.

```
sass > styles.scss
1  @use 'partial';
2
3
```

Hay que tener en cuenta que cuando se utiliza la regla **@use** no se debe poner el guion bajo dentro de las comillas, ni tampoco la extensión, al momento de llamar al Partial.

4. Modules

El uso de Partials nos permite tener nuestro código modularizado. Esto quiere decir que no tienes que tener todo tu código dentro de una misma hoja SCSS, puedes dividir tu código en diferentes hojas como módulos de un todo.

Para tener nuestro código modularizado, esto usamos los Partials que tratamos en el punto anterior y donde podremos alojar no solamente pedazos de código, sino que también **mixins** los mismos que veremos en los siguientes puntos.

5. Mixins

Los Mixins, son pedazos de código reutilizable, que nos permitirá ahorrar tiempo al construir componentes que tengan estilos repetitivos como por ejemplo un botón.

Para crear un mixin debemos utilizar la regla **@mixin** seguido del nombre que le queremos asignar.

```
@mixin button {  
  background-color: red;  
  color: white;  
  border-radius: 4px;  
  width: 100px;  
  height: 35px;  
  padding: 8px;  
  text-align: center;  
}
```

Para llamar a un mixin, basta con utilizar la regla **@include** seguido del nombre del mixin:

```
.submit{  
  @include button;  
}
```

De esta manera el selector `.submit` tendrá todos los estilos que se encuentran dentro del mixin `button`.

Uso de parámetros

Los mixins también pueden contener parámetros los cuales deben ser ingresados cada vez que se llama al mixin.

SCSS

```
@mixin card($width, $height, $f-size){
  width: $width;
  height: $height;
  border-radius: 4px;
  padding: 10px;
  font-size: $f-size;
  background-color: coral;
}

.card-user{
  @include card(350px, 400px, 16px);
}
```

CSS generado

```
.card-user {
  width: 350px;
  height: 400px;
  border-radius: 4px;
  padding: 10px;
  font-size: 16px;
  background-color: coral;
}
```

Los parámetros tienen el signo **\$** al inicio ya que vienen a ser variables dentro de SASS.

6. Extend

Otra forma de reutilizar código dentro de SASS es a través de la regla **@extend**, el mismo que permite extender el código de otro selector. Esto se logra anteponiendo **@extend** al selector que queremos extender sus estilos de la siguiente forma:

SCSS

```
%message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  @extend %message;
  border-color: green;
}

.error {
  @extend %message;
  border-color: red;
}

.warning {
  @extend %message;
  border-color: yellow;
}
```

CSS generado

```
.warning, .error, .success {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  border-color: green;
}

.error {
  border-color: red;
}

.warning {
  border-color: yellow;
}
```

El selector % representa un elemento que no se crea en el CSS generado pero que nos sirve para poderlo extender.

7. Operators

SASS tiene su propia librería para hacer operaciones matemáticas. Esta librería se llama **sass:math** y se la llama de la siguiente manera:

```
@use 'sass:math';
```

Esta librería permite hacer las siguientes operaciones básicas, cada uno con su respectivo operador:

| Operación Matemática | Operador SASS |
|----------------------|---------------|
| Suma | + |
| Resta | - |
| Multiplicación | * |
| División | div() |
| Porcentaje | % |

Un ejemplo de uso sería el siguiente:

SCSS

```
@use 'sass:math';

article{
  width: math.div(600px, 960px) * 100%;
}

aside{
  width: math.div(300px, 960px) * 100%;
  margin-left: auto;
}
```

CSS generado

```
article {
  width: 62.5%;
}

aside {
  width: 31.25%;
  margin-left: auto;
}
```

8. Condicionales y bucles

Al igual que un lenguaje de programación, SASS también dispone de condiciones como if, else, while o for. El uso de un if/else dentro de un mixin sería de la siguiente manera:

SCSS

```
@mixin tab($active){
  width: 30px;
  height: 15px;
  @if $active == true{
    background-color: ■ crimson;
  }@else{
    background-color: ■ chocolate;
  }
}

.tab--active{
  @include tab(true);
}
```

CSS generado

```
@mixin button {
  background-color: ■ red;
  color: ■ white;
  border-radius: 4px;
  width: 300px;
  height: 40px;
}
```

⚠ El uso de bucles está fuera del alcance de esta guía, sin embargo, si deseas profundizar tus conocimientos en estos temas puedes hacer clic [aquí](#)

Esta guía fue creada por [David Ruiz](#), en base a la [documentación](#) oficial de SASS.
¡Cualquier sugerencia de mejora es bienvenida!

Si te gustó este contenido, mis clases particulares te van
a encantar.

Soy un profesor ☆☆☆☆☆ en **classgap**

[Reservar una clase gratuita de 20 minutos!](#)

También puedes seguirme a través de mis redes sociales:

