



Unión Europea

Fondo Social Europeo
El FSE invierte en tu futuro



1

DIW Curso 24-25 | IES CAMP DE MORVEDRE

El preprocesador CSS: SASS

DAW - DIW

PROF: ALICIA FERNÁNDEZ CATALÁN

CURSO 24-25



Contenidos

- ▶ ¿Qué es Sass?
- ▶ Descarga e instalación
- ▶ Configuración de Sass
- ▶ Estructura de archivos
- ▶ Uso de Sass

¿Qué es SASS?

- ▶ El lenguaje de hojas de estilo CSS permite trabajar con recursos adicionales que agilizan la programación de los archivos de estilo, se trata de los preprocesadores CSS.
- ▶ Los preprocesadores permiten que el desarrollo, modificación y mantenimiento de hojas CSS resulte más sencillo gracias al uso de variables o bloque de código(mixin)
- ▶ **SASS (Syntactically Awesome Stylesheets)** es un preprocesador de CSS que extiende las capacidades de CSS, añadiendo características como variables, anidamiento de reglas, mixins, herencia, entre otras.
- ▶ El framework **Bootstrap se basa en este preprocesador**

¿Qué es SASS? - Sintaxis

► **SASS tiene dos sintaxis:**

- **SASS** (sin llaves ni punto y coma): es la sintaxis original de SASS, más concisa, sin necesidad de llaves {} ni punto y coma ;.
- **SCSS** (Sassy CSS): es una versión más moderna de SASS que es completamente compatible con CSS. Permite usar llaves y punto y coma, lo que lo hace más familiar para los desarrolladores que ya conocen CSS.
- En este manual, **trabajaremos principalmente con la sintaxis SCSS**, ya que es la más utilizada hoy en día.

¿Por qué usar SASS?

- ▶ SASS facilita la escritura de CSS al introducir funciones como la reutilización de código, modularidad y la organización de estilos.
- ▶ Algunos beneficios clave de SASS son:
 - ▶ **Variables:** permiten almacenar valores como colores, tamaños, fuentes, etc.
 - ▶ **Anidamiento:** permite anidar reglas dentro de otras reglas para reflejar la estructura del HTML.
 - ▶ **Mixins:** son fragmentos de código reutilizables, similares a funciones.
 - ▶ **Extends/Inheritance:** permiten compartir un conjunto de reglas CSS entre diferentes selectores.
 - ▶ **Funciones y operaciones:** se pueden realizar operaciones aritméticas y usar funciones para manipular valores CSS.

Descarga e instalación con diferentes plataformas

- ▶ **La instalación se puede hacer por la línea de comandos con Ruby o con Node.js**
 - ▶ Ruby es más anticuado
- ▶ Lee el documento de Aules con las principales diferencias entre una instalación y otro.
 - ▶ Instalación de Sass con Ruby o con Node.js

Additional resources



Instalación de Sass con Ruby o con Node.js

Sass se puede instalar mediante dos gestores:

- Ruby
- Node.js

En el documento te presento la diferencia entre los dos. Léelo atentamente

Descarga e instalación con Ruby

- ▶ Sass es un gem(un paquete del gestor de paquetes de Ruby-RubyGems) de Ruby, por tanto, **hemos de instalar Ruby en nuestro equipo.**
- ▶ **La descarga de Ruby es diferente en función del SO que tengamos**
 - ▶ **Windows:** Descargamos el paquete del sitio web <https://rubyinstaller.org/>
 - ▶ **Linux:** Esta preinstalado, pero si no usamos apt install ruby o el comando de instalación según la distribución. apt es para distribuciones derivadas de Debian
 - ▶ **Mac:** Ejecutando rvm install ruby-versión-actual

Instalación en Windows con Ruby

- ▶ Cuando descargamos, le damos a ejecutar al .exe.
- ▶ Puede que nos aparezca una advertencia de seguridad, la aceptamos y continuamos la instalación.



Downloads

RubyInstallers

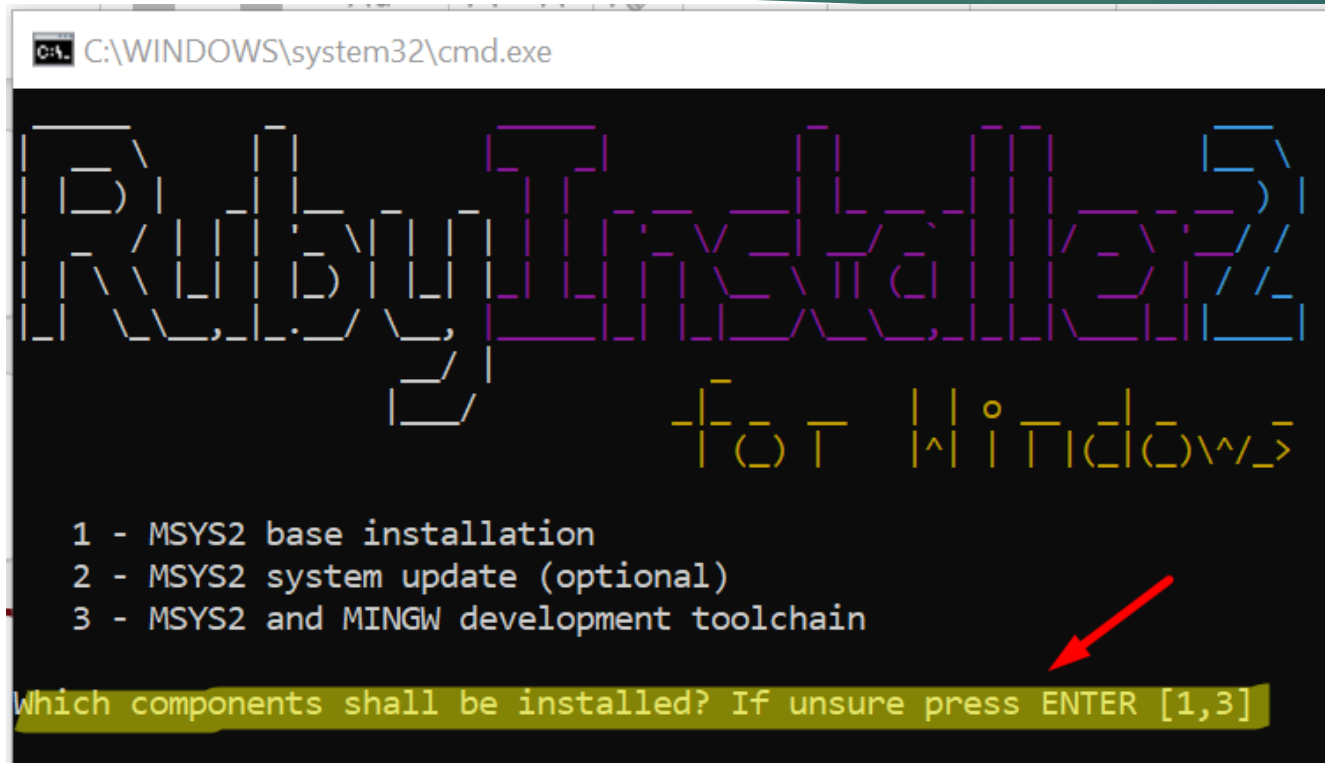
Archives»

Looking for an **ARM64 version**? There's a [fundraiser to buy the main developer of RubyInstaller a notebook](#) with ARM processor to make it happen.

WITH DEVKIT

- => **Ruby+Devkit 3.3.6-2 (x64)**
- Ruby+Devkit 3.3.6-2 (x86)
- Ruby+Devkit 3.2.6-1 (x64)
- Ruby+Devkit 3.2.6-1 (x86)
- Ruby+Devkit 3.1.6-1 (x64)
- Ruby+Devkit 3.1.6-1 (x86)
- Ruby+Devkit 3.0.7-1 (x64)
- Ruby+Devkit 3.0.7-1 (x86)

Instalación en Windows con Ruby



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

Ruby Installed
For Windows

1 - MSYS2 base installation
2 - MSYS2 system update (optional)
3 - MSYS2 and MINGW development toolchain

Which components shall be installed? If unsure press ENTER [1,3]
```

Al completar la instalación, al final nos da un mensaje de que se ha instalado, le damos a la tecla enter y se cierra el cmd

Instalación en Windows con Ruby

C:\> Símbolo del sistema

```
Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\alici>gem install sass
Successfully installed ffi-1.17.0-x64-mingw-ucrt
Successfully installed rb-inotify-0.11.1
Successfully installed rb-fsevent-0.11.2
Successfully installed sass-listen-4.0.0

Ruby Sass has reached end-of-life and should no longer be used.
```

Instalamos el paquete de Sass

A partir de ahora ya podemos empezar a trabajar con Sass

```
C:\Users\alici>sass -v
Ruby Sass 3.7.4
C:\Users\alici>
```

Comprobamos la versión para saber que lo tenemos instalado

Para saber más, consulta:

<https://www.arsys.es/blog/sass-css-preprocesador#tree-1>

Instalación y configuración con Node.js en Windows

► Instalación de SASS con Node.js (npm)

- **Instalar Node.js:** Si aún no lo tienes instalado, descarga e instala Node.js.
 - <https://nodejs.org/en>
 - Descarga un paquete .msi
 - Ejecútalo y sigue el asistente. Dale al check para que también se te instalen herramientas adicionales.
 - Para comprobar que tenemos instalado node ejecuta **node -v** en un terminal
- **Instalar SASS:** Una vez que Node.js esté instalado, puedes instalar SASS con el siguiente comando:

```
npm install -g sass
```

Si necesitas más información:

<https://marantbq.dev/blog/compilar-sass-node-js/>

Resultado instalación node.js y Sass en windows

```
Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\alici>node -v
v22.12.0

C:\Users\alici>npm install -g sass

added 17 packages in 4s

5 packages are looking for funding
  run `npm fund` for details

npm notice
npm notice New patch version of npm available! 10.9.0 -> 10.9.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.9.2
npm notice To update run: npm install -g npm@10.9.2
npm notice

C:\Users\alici>
```

```
C:\Users\alici>sass --version
1.82.0 compiled with dart2js 3.5.4

C:\Users\alici>
```

Para ver la versión de Sass que tenemos instalada

Configuración de Sass

► Compilar archivos SASS/SCSS a CSS

- Una vez instalado SASS, puedes compilar archivos SCSS a CSS utilizando el siguiente comando en la terminal:

```
sass input.scss output.css
```

- Si deseas que la compilación se haga de manera continua (es decir, cada vez que cambies el archivo SCSS), puedes usar el siguiente comando:

```
sass --watch input.scss:output.css
```


Estructura de Archivos

- La estructura típica de un proyecto con SASS podría ser la siguiente:

```
/scss
├── _variables.scss
├── _mixins.scss
├── _reset.scss
├── main.scss
/css
├── main.css
```

- Los archivos con un guión bajo (por ejemplo, `_variables.scss`) son archivos parciales que SASS no compilará directamente, pero se importarán en otros archivos SCSS.
- **main.scss** será el archivo principal donde se importan los demás archivos parciales.

Uso de SASS - Variables

- ▶ **Variables**
- ▶ Las variables en SASS permiten almacenar valores que se usan de manera repetida, como colores, fuentes, tamaños, etc.
- ▶ Para la creación de variables se usa el \$

```
SCSS

// Definición de variables
$primary-color: #3498db;
$secondary-color: #2ecc71;
$font-stack: 'Helvetica', sans-serif;

// Uso de variables
body {
  font-family: $font-stack;
  background-color: $primary-color;
}

h1 {
  color: $secondary-color;
}
```

Uso de SASS - Nesting

- ▶ **Anidamiento (Nesting)**
- ▶ Una de las características más poderosas de SASS es el anidamiento, que te permite escribir reglas CSS de forma jerárquica, reflejando la estructura HTML.

SCSS

```
// Sin anidamiento
header {
  background-color: #333;
}

header h1 {
  color: #fff;
}
```

```
// Con anidamiento
header {
  background-color: #333;

  h1 {
    color: #fff;
  }
}
```

Uso de SASS - Mixins

- ▶ **Mixins**
- ▶ Los mixins son bloques de código que puedes reutilizar en varios lugares, incluso con parámetros, lo que te permite crear estilos flexibles.
- ▶ Se les conoce también como *componentes reutilizables*
- ▶ Para implementarlos se usa la etiqueta @mixin, seguida del nombre del bloque que se quiera utilizar, pudiendo recibir de forma opcional parámetros.
- ▶ Para usarlo, utilizamos la etiqueta @include

SCSS

```
// Definir un mixin
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
  -moz-border-radius: $radius;
  border-radius: $radius;
}
```

Uso del mixin

```
.box{
  @include border-radius(10px);
  width: 100px;
  height: 100px;
  background-color: aqua;
}
```

Uso de SASS - Funciones

► Funciones

- Las funciones en SASS te permiten manipular valores y devolver resultados.

```
SCSS

// Definir una función
@function calculate-rem($pixels) {
  $rem-size: $pixels / 16;
  @return #{$rem-size}rem;
}

// Usar la función
h1 {
  font-size: calculate-rem(32px); // 2rem
}
```


Uso de SASS - Herencia

► Extends / Herencia

- La directiva @extend permite que un selector herede las reglas de otro, evitando la repetición de código.

```
// Definir un estilo base
%button-style {
  padding: 10px 20px;
  font-size: 16px;
  border-radius: 5px;
  border: none;
  cursor: pointer;
}

// Usar extend
.primary-button {
  @extend %button-style;
  background-color: #3498db;
  color: white;
}

.secondary-button {
  @extend %button-style;
  background-color: #2ecc71;
  color: white;
}
```

Uso de SASS - @import

- ▶ **Partials y @import**
- ▶ SASS te permite dividir tu código en archivos más pequeños y luego combinarlos usando @import.
 - ▶ Lo hemos nombrado ya cuando hemos hablado de la [estructura de ficheros de Sass](#)
- ▶ Esto mejora la organización y el mantenimiento del código.

```
// _variables.scss
$primary-color: #3498db;

// _buttons.scss
.primary-button {
  background-color: $primary-color;
  color: white;
}

// main.scss
@import 'variables';
@import 'buttons';

// Estilo final
body {
  font-family: Arial, sans-serif;
}

.primary-button {
  padding: 10px 20px;
}
```

Uso de SASS - Operaciones

- ▶ **Operaciones**
- ▶ **SASS permite realizar operaciones aritméticas, lógicas y de comparación directamente en el código.**
- ▶ Los operadores de comparación(==, !=, >, <, >=, <=) devuelven un valor booleano (true o false).
- ▶ Los operadores lógicos (and, or, not) se utilizan para operaciones booleanas en expresiones condicionales.
- ▶ Los operadores +, -, *, /, % se usan comúnmente para trabajar con números, colores, y tamaños.

```
$base-font-size: 16px;  
$heading-font-size: $base-font-size * 2;  
  
h1 {  
  font-size: $heading-font-size; // 32px  
}
```

Mira el documento de Aules sobre Operadores en Sass

Uso de Sass. Operaciones lógicas

- ▶ En Sass, las operaciones lógicas se utilizan principalmente para realizar comparaciones entre valores.
- ▶ Estas operaciones también permiten tomar decisiones dentro de los estilos o aplicar condiciones basadas en los valores que estamos usando.
- ▶ Sass, como extensión de CSS, proporciona algunas herramientas para realizar operaciones lógicas con los valores.
- ▶ A continuación, veremos las **principales operaciones lógicas disponibles**
 - ▶ Condicionales
 - ▶ Bucles

Uso de Sass. Condicionales

► Estructuras de control. Condicionales

1. **if:** Nos permite crear una condición dentro de una variable CSS.
 - `$variable: if ($condición, $valor-true, $valor-false);`
2. **@if, @else if, @else:** Se utilizan para crear bloques de código condicional basado en variables condicionales

Ejemplo 1

SCSS

```
$a: 10;
$b: 15;
@if $a > $b or $a == 10 {
  // Esto se ejecutará porque la segunda condición es verdadera
  color: cyan;
}
```

Ejemplo 2

SCSS

```
$color: blue;
$font-size: 16px;

@if $color == blue and $font-size == 16px {
  color: green;
  font-size: 18px;
} @else {
  color: red;
}
```


Uso de Sass. Condicionales

► Condicionales. Ejemplo 3:

- Primera condición (@if): Si la variable \$color es igual a blue y \$font-size es igual a 18px, se aplican los estilos definidos dentro de esta condición: fondo azul, texto blanco y un tamaño de fuente de 18px.
- Segunda condición (@else if): Si la primera condición no se cumple, pero la variable \$color es red y \$layout es 'flex', entonces se aplican los estilos definidos en esta condición: fondo rojo, texto negro y un tamaño de fuente de 20px.
- Condición por defecto (@else): Si ninguna de las condiciones anteriores se cumple, se aplican los estilos definidos en el bloque @else: fondo gris, texto blanco y un tamaño de fuente de 16px.
- Resultado:
- Este código permite que el estilo aplicado dependa de las variables que hayas configurado (\$color, \$font-size, \$layout). Si las condiciones no se cumplen, se aplican los estilos por defecto.
- Puedes cambiar los valores de las variables para ver cómo cambia el resultado de los estilos.

SCSS

```
$color: blue;
$font-size: 18px;
$layout: 'grid';

@if $color == blue and $font-size == 18px {
  // Si el color es azul y el tamaño de fuente es 18px
  background-color: blue;
  color: white;
  font-size: $font-size;
} @else if $color == red and $layout == 'flex' {
  // Si el color es rojo y el layout es flex
  background-color: red;
  color: black;
  font-size: 20px;
} @else {
  // Si no se cumple ninguna de las condiciones anteriores
  background-color: gray;
  color: white;
  font-size: 16px;
}
```

Uso de Sass. Bucles - @for

► Estructuras de control. Bucles

- **@for:** itera sobre un rango de valores para la \$variable. El bucle @for en Sass se usa para generar estilos repetitivos o aplicar reglas a una secuencia de valores, como números o intervalos.

► Explicación del código:

- Variable \$base-size: Es una variable que guarda el tamaño base de la fuente.
- Bucle @for:
- El bucle @for \$i from 1 through 5 crea un ciclo que va desde 1 hasta 5, inclusive.
- Dentro del bucle, el selector .font-size-#{ \$i } crea una clase con un número dinámico que va a ser reemplazado por el valor de \$i.
- La propiedad font-size se calcula multiplicando el tamaño base \$base-size por el valor actual de \$i. Así, los tamaños de fuente aumentarán en múltiplos de 16px.

SCSS

```
$base-size: 16px;
```

```
@for $i from 1 through 5 {  
  .font-size-#{ $i } {  
    font-size: $base-size * $i;  
  }  
}
```

Uso de Sass. Bucles - @for

► Estructuras de control. Bucles

- Resultado en CSS del ejemplo anterior

► Variación con to:

- Si prefieres que el ciclo no incluya el último valor, puedes usar **to** en lugar de **through**

SCSS

```
@for $i from 1 to 5 {  
  .font-size-#{ $i } {  
    font-size: $base-size * $i;  
  }  
}
```

En este caso, el bucle se ejecutaría de 1 a 4, excluyendo el 5.

Este tipo de bucles en Sass es muy útil para generar estilos dinámicos, como tamaños de fuente, márgenes, o cualquier propiedad que necesite un patrón repetitivo.

CSS

```
.font-size-1 {  
  font-size: 16px;  
}  
  
.font-size-2 {  
  font-size: 32px;  
}  
  
.font-size-3 {  
  font-size: 48px;  
}  
  
.font-size-4 {  
  font-size: 64px;  
}  
  
.font-size-5 {  
  font-size: 80px;  
}
```

Uso de Sass. Bucles - @each

- ▶ **Estructuras de control. Bucles**
- ▶ **@each** en Sass se utiliza para iterar sobre una lista o un mapa y aplicar estilos o reglas CSS a cada uno de sus elementos.
- ▶ Es muy útil cuando necesitas aplicar el mismo conjunto de reglas a varios elementos sin escribir manualmente cada uno de ellos.

Uso de Sass. Bucles - @each

► Estructuras de control. Bucles

- Aquí tienes un ejemplo de cómo utilizar @each con una lista y con un mapa.

```
SCSS

$colors: red, blue, green, yellow;

@each $color in $colors {
  .bg-#{$color} {
    background-color: $color;
  }
}
```

Explicación:

- **Lista de colores:** La variable \$colors contiene una lista de colores.
- **Bucle @each:** El bucle @each itera sobre cada uno de los valores en la lista \$colors.
 - En cada iteración, el valor actual se asigna a la variable \$color.
 - La clase .bg-#{\$color} se crea para cada valor de color, y el color de fondo se establece en el valor correspondiente.
- **Interpolación:** La interpolación #{\$color} dentro del nombre de la clase permite que el nombre de la clase cambie dinámicamente dependiendo del color actual.

Uso de Sass. Bucles - @each

- ▶ **Estructuras de control. Bucles**
- ▶ Resultado CSS generado del ejemplo anterior

```
css

.bg-red {
  background-color: red;
}

.bg-blue {
  background-color: blue;
}

.bg-green {
  background-color: green;
}

.bg-yellow {
  background-color: yellow;
}
```

Uso de Sass. Bucles - @each

► Estructuras de control. Bucles

► Ejemplo con un mapa:

- Un mapa en Sass es una estructura de datos que contiene claves y valores.
- Es útil cuando quieres asociar un conjunto de propiedades con claves específicas.

SCSS

```
$themes: (  
  light: #f0f0f0,  
  dark: #333333,  
  primary: #007bff,  
  secondary: #6c757d  
);  
  
@each $name, $color in $themes {  
  .theme-#{$name} {  
    background-color: $color;  
    color: white;  
  }  
}
```

Uso de Sass. Bucles - @each

► Estructuras de control. Bucles

► Explicación:

- **Mapa de temas:** La variable \$themes es un mapa donde cada clave (por ejemplo, light, dark, etc.) tiene un valor de color correspondiente.

- **Bucle @each:** El bucle @each recorre las claves y los valores del mapa.

- \$name recibe la clave del mapa (por ejemplo, light, dark).
- \$color recibe el valor correspondiente (por ejemplo, #f0f0f0, #333333).

- Se crea una clase para cada clave (.theme-light, .theme-dark, etc.), y el color de fondo se asigna según el valor del mapa.

CSS

```
.theme-light {  
  background-color: #f0f0f0;  
  color: white;  
}  
  
.theme-dark {  
  background-color: #333333;  
  color: white;  
}  
  
.theme-primary {  
  background-color: #007bff;  
  color: white;  
}  
  
.theme-secondary {  
  background-color: #6c757d;  
  color: white;  
}
```

Uso de Sass. Bucles - @while

- ▶ La directiva **@while** en Sass permite ejecutar un bloque de código mientras se cumpla una condición.
 - ▶ Es similar a los bucles while en otros lenguajes de programación, donde el código se repite mientras que la condición especificada sea true

Uso de Sass. Bucle @while

33

► La directiva @while en Sass

SCSS

```
$i: 1;

@while $i <= 5 {
  .item-#{ $i } {
    width: $i * 20px;
    height: $i * 20px;
  }
  $i: $i + 1;
}
```

Explicación:

- Inicialización de \$i: La variable \$i comienza en 1.
- Condición @while: El bucle se ejecutará mientras \$i sea menor o igual a 5.
- Estilos generados: Por cada iteración del bucle, se genera una clase .item-#{ \$i } con un ancho y altura de 20px multiplicado por el valor de \$i.
- Incremento de \$i: Después de cada iteración, \$i se incrementa en 1.

CSS

```
.item-1 {
  width: 20px;
  height: 20px;
}

.item-2 {
  width: 40px;
  height: 40px;
}

.item-3 {
  width: 60px;
  height: 60px;
}

.item-4 {
  width: 80px;
  height: 80px;
}

.item-5 {
  width: 100px;
  height: 100px;
}
```

Código CSS
generado

Uso de Sass. @warn

- ▶ La directiva **@warn** en Sass se utiliza para mostrar mensajes de advertencia en la consola cuando estás compilando tus archivos Sass.
- ▶ Estos mensajes no afectan al CSS generado, pero sirven para proporcionar información o alertas durante el desarrollo.

```
SCSS

$primary-color: blue;
$secondary-color: red;

@warn "El color primario es: #{$primary-color}";
@warn "Recuerda usar colores accesibles.";
```

```
WARNING: El color primario es: blue
         on line 5 of src/styles.scss
         Use --trace for backtrace.
WARNING: Recuerda usar colores accesibles.
         on line 6 of src/styles.scss
         Use --trace for backtrace.
```

Resultado del @warn por consola del editor que estemos usando

Uso de SASS. Media Queries

- ▶ **Media Queries**
- ▶ SASS facilita el uso de media queries mediante anidamiento, lo que hace que el código sea más limpio y fácil de leer.

```
.container {  
  width: 100%;  
  
  @media (max-width: 768px) {  
    width: 80%;  
  }  
  
  @media (max-width: 480px) {  
    width: 100%;  
  }  
}
```

Ejemplo Completo

- ▶ Aquí tienes un ejemplo completo de un archivo SCSS que incluye varias de las características mencionadas:
- ▶ **Ejemplo de Aules: Ejemplo-SASS.scss**
- ▶ En este ejemplo:
 - ▶ Definimos variables para colores y fuentes.
 - ▶ Usamos un mixin para aplicar sombras.
 - ▶ Creamos un botón con un efecto hover utilizando la función `darken()`.
 - ▶ Utilizamos un contenedor y tarjetas con propiedades anidadas.

Conclusión

- ▶ SASS es una herramienta poderosa que mejora la productividad en el desarrollo de hojas de estilo CSS.
- ▶ Su sintaxis sencilla y sus características avanzadas (como variables, mixins, anidamiento y herencia) hacen que escribir y mantener estilos sea mucho más eficiente y organizado.
- ▶ A medida que avances en tu carrera como desarrollador web, aprender y dominar SASS será una habilidad valiosa que te permitirá trabajar de manera más profesional.