



Documentación SASS

Diseño de aplicaciones Web (DAW)

Julián Camilo Castro Restrepo

12-11-2024

ÍNDICE

1. Instalar Sass	2
2. Conceptos básicos de Sass.....	3
3. Mapas	11

1. Instalar Sass

Cuando instales Sass en la línea de comandos, podrás ejecutar el sass ejecutable para compilar .sass y convertir .scss archivos en .css archivos. Por ejemplo:

```
sass source/stylesheets/index.scss build/stylesheets/index.css
```

Primero, instala Sass usando una de las opciones que se indican a continuación y luego ejecútalo sass --version para asegurarte de que se haya instalado correctamente. Si es así, se incluirá 1.81.0. También puedes ejecutarlo sass --help para obtener más información sobre la interfaz de línea de comandos.

Una vez que esté todo configurado, ve y juega. Si eres nuevo en Sass, hemos preparado algunos recursos para ayudarte a aprender bastante rápido.

Obtenga más información sobre Sass

Instalar en cualquier lugar (independiente)

Puedes instalar Sass en Windows, Mac o Linux descargando el paquete para tu sistema operativo desde GitHub y agregándolo a tu PATH. Eso es todo: no hay dependencias externas ni nada más que debas instalar.

Instalar en cualquier lugar (npm)

Si usa Node.js, también puede instalar Sass usando npm ejecutando

```
npm install -g sass
```

2. Conceptos básicos de Sass

Antes de poder usar Sass, debes configurarlo en tu proyecto. Si solo quieres navegar aquí, hazlo, pero te recomendamos que primero instales Sass.

Preprocesamiento

EL CSS por sí solo puede ser divertido, pero las hojas de estilo son cada vez más grandes, más complejas y más difíciles de mantener. Aquí es donde un preprocesador puede ayudar. Sass tiene características que aún no existen en CSS, como anidación, mixins, herencia y otras ventajas ingeniosas que lo ayudan a escribir CSS robusto y fácil de mantener .

Una vez que comiences a trabajar con Sass, tomará tu archivo Sass preprocesado y lo guardará como un archivo CSS normal que podrás usar en tu sitio web.

La forma más directa de hacer que esto suceda es en tu terminal. Una vez que Sass esté instalado, puedes compilar tu Sass a CSS usando el `sass` comando. Necesitarás decirle a Sass desde qué archivo compilar y dónde generar CSS . Por ejemplo, ejecutar `sass input.scss output.css` desde tu terminal tomaría un solo archivo Sass, `input.scss` y compilaría ese archivo a `output.css`.

También puedes observar archivos o directorios individuales con el `--watch` indicador. El indicador de observación le indica a Sass que observe los archivos de origen para ver si hay cambios y vuelva a compilar EL CSS cada vez que guardes tu Sass. Si quisieras observar (en lugar de compilar manualmente) tu `input.scss` archivo, simplemente agregarías el indicador de observación a tu comando, de la siguiente manera:

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

Puede ver y enviar datos a directorios utilizando rutas de carpeta como entrada y salida, y separándolas con dos puntos. En este ejemplo:

```
sass --watch app/sass:public/stylesheets
```

Sass vigilaría todos los archivos de la `app/sass` carpeta para detectar cambios y compilaría CSS en la `public/stylesheets` carpeta.

Dato curioso:

Sass tiene dos sintaxis. La sintaxis SCSS (`.scss`) es la más utilizada. Es un superconjunto de CSS, lo que significa que todo CSS VÁLIDO TAMBIÉN ES SCSS válido. La sintaxis con sangría (`.sass`) es más inusual: utiliza sangría en lugar de llaves para anidar declaraciones y saltos de línea en lugar de punto y coma para separarlas. Todos nuestros ejemplos están disponibles en ambas sintaxis.

Variables

Piense en las variables como una forma de almacenar información que desea reutilizar en toda su hoja de estilo. Puede almacenar elementos como colores, pilas de fuentes o cualquier valor CSS que crea que desea reutilizar. Sass utiliza el `$`símbolo para convertir algo en una variable. A continuación, se muestra un ejemplo:

Patio de juegos

SINTAXIS SCSS

```
$font-stack: Helvetica, sans-serif;  
$primary-color: #333;
```

```
body {  
  font: 100% $font-stack;  
  color: $primary-color;  
}
```

SALIDA CSS

```
body {  
  font: 100% Helvetica, sans-serif;  
  color: #333;  
}
```

Cuando se procesa Sass, toma las variables que definimos para `$font-stack` y `$primary-color` y genera UN CSS normal con los valores de nuestras variables colocados en el CSS. Esto puede ser extremadamente eficaz cuando se trabaja con colores de marca y se mantienen consistentes en todo el sitio.

Anidación

Al escribir HTML probablemente hayas notado que tiene una jerarquía anidada y visual clara. CSS , por otro lado, no la tiene.

Sass te permitirá anidar tus selectores CSS de una manera que siga la misma jerarquía visual de tu HTML . TEN EN CUENTA QUE LAS REGLAS DEMASIADO ANIDADAS DARÁN COMO RESULTADO CSS sobrecalificado que podría resultar difícil de mantener y, por lo general, se considera una mala práctica.

Con esto en mente, aquí hay un ejemplo de algunos estilos típicos para la navegación de un sitio:

Patio de juegos

SINTAXIS SCSS

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

SALIDA CSS

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

Notarás que los `ul` selectores `li`, y `a` están anidados dentro del `nav` selector. Esta es una excelente manera de organizar tu CSS y hacerlo más legible.

Parciales

Puedes crear archivos Sass parciales que contengan pequeños fragmentos de CSS que puedes incluir en otros archivos Sass. Esta es una excelente manera de modularizar tu CSS y ayudar a que las cosas sean más fáciles de mantener. Un archivo Sass parcial es un archivo Sass cuyo nombre está precedido por un guión bajo. Puedes ponerle un nombre como `_partial.scss`. El guión bajo le permite a Sass saber que el archivo es solo un archivo parcial y que no debe generarse en un archivo CSS . Los archivos Sass parciales se usan con la `@use` regla.

Módulos

Compatibilidad:

Dardo Sass

desde 1.23.0

LibSass

X

Rubí Sass

X



No tienes que escribir todo tu Sass en un solo archivo. Puedes dividirlo como quieras con la `@use` regla. Esta regla carga otro archivo Sass como *módulo* , lo que significa que puedes hacer referencia a sus variables, mixins y funciones en tu archivo Sass con un espacio de nombres basado en el nombre del archivo. ¡Usar un archivo también incluirá el CSS que genera en tu salida compilada !

SINTAXIS SCSS

```
//_base.scss
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}

// styles.scss
@use 'base';

.inverse {
  background-color: base.$primary-color;
  color: white;
}
```

SALIDA CSS

```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}

.inverse {
  background-color: #333;
  color: white;
}
```

Ten en cuenta que lo estamos usando `@use 'base';` en el `styles.scss` archivo. Cuando usas un archivo, no necesitas incluir la extensión del archivo. Sass es inteligente y lo descubrirá por ti.

Mezclas

Algunas cosas en CSS son un poco tediosas de escribir, especialmente con CSS3 y los muchos prefijos de proveedores que existen. Un mixin te permite crear grupos de declaraciones CSS que quieras reutilizar en todo tu sitio. Ayuda a mantener tu Sass muy DRY . Incluso puedes pasar valores para que tu mixin sea más flexible. Aquí tienes un ejemplo para `theme`.

Patio de juegos

SINTAXIS SCSS

```
@mixin theme($theme: DarkGray) {
  background: $theme;
  box-shadow: 0 0 1px rgba($theme, .25);
  color: #fff;
}
```

```

.info {
  @include theme;
}
.alert {
  @include theme($theme: DarkRed);
}
.success {
  @include theme($theme: DarkGreen);
}

```

SALIDA CSS

```

.info {
  background: DarkGray;
  box-shadow: 0 0 1px rgba(169, 169, 169, 0.25);
  color: #fff;
}

.alert {
  background: DarkRed;
  box-shadow: 0 0 1px rgba(139, 0, 0, 0.25);
  color: #fff;
}

.success {
  background: DarkGreen;
  box-shadow: 0 0 1px rgba(0, 100, 0, 0.25);
  color: #fff;
}

```

Para crear un mixin, utiliza la `@mixin` directiva y dale un nombre. Hemos llamado a nuestro mixin `theme`. También estamos usando la variable `$theme` dentro de los paréntesis para que podamos pasar un `theme` de lo que queramos. Después de crear tu mixin, puedes usarlo como una declaración CSS comenzando con `@include` seguido del nombre del mixin.

Extender/Herencia

El uso de `@extend` permite compartir un conjunto de propiedades CSS de un selector a otro. En nuestro ejemplo, vamos a crear una serie simple de mensajes para errores, advertencias y éxitos utilizando otra característica que va de la mano con las clases extendidas, las clases de marcador de posición. Una clase de marcador de posición es un tipo especial de clase que solo se imprime cuando se extiende y puede ayudar a mantener su CSS compilado ordenado y limpio.

Patio de juegos

SINTAXIS SCSS

```
/* This CSS will print because %message-shared is extended. */
%message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

// This CSS won't print because %equal-heights is never extended.
%equal-heights {
  display: flex;
  flex-wrap: wrap;
}

.message {
  @extend %message-shared;
}

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

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

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

Lo que hace el código anterior es indicar a .message, .success, .error, y .warning que se comporten como %message-shared. Esto significa que en cualquier lugar donde %message-shared aparezcan, .message, .success, .error, y .warning también lo harán. La magia ocurre en el CSS generado , donde cada una de estas clases obtendrá las mismas propiedades CSS %message-shared que . Esto le ayuda a evitar tener que escribir varios nombres de clase en elementos HTML .

Puede ampliar la mayoría de los selectores CSS simples además de las clases de marcador de posición en Sass, pero usar marcadores de posición es la forma más fácil de asegurarse de no estar extendiendo una clase que está anidada en otra parte de sus estilos, lo que puede generar selectores no deseados en su CSS .

Tenga en cuenta que el CSS %equal-heights no se genera porque nunca %equal-heights se extiende.

Operadores

Realizar operaciones matemáticas en CSS es muy útil. Sass tiene varios operadores matemáticos estándar como +, -, *, math.div() y %. En nuestro ejemplo, vamos a realizar operaciones matemáticas simples para calcular el ancho de un article y aside.

Patio de juegos

SINTAXIS SCSS

```
@use "sass:math";  
  
.container {  
  display: flex;  
}  
  
article[role="main"] {  
  width: math.div(600px, 960px) * 100%;  
}  
  
aside[role="complementary"] {  
  width: math.div(300px, 960px) * 100%;  
  margin-left: auto;  
}
```

SALIDA CSS

```
.container {  
  display: flex;  
}  
  
article[role=main] {  
  width: 62.5%;  
}  
  
aside[role=complementary] {  
  width: 31.25%;  
  margin-left: auto;  
}
```

Hemos creado una cuadrícula fluida muy simple, basada en 960 píxeles. Las operaciones en Sass nos permiten hacer algo como tomar valores de píxeles y convertirlos en porcentajes sin demasiados problemas.

3. Mapas

Los mapas en Sass contienen pares de claves y valores, y facilitan la búsqueda de un valor por su clave correspondiente. Se escriben (<expression>: <expression>, <expression>: <expression>). La expresión antes de :es la clave, y la expresión después es el valor asociado con esa clave. Las claves deben ser únicas, pero los valores pueden estar duplicados. A diferencia de las listas, los mapas *deben* escribirse entre paréntesis. Un mapa sin pares se escribe () .

💡 Dato curioso:

Los lectores astutos pueden notar que un mapa vacío, (), se escribe igual que una lista vacía. Esto se debe a que cuenta como un mapa y una lista. De hecho, ¡*todos* los mapas cuentan como listas! Cada mapa cuenta como una lista que contiene una lista de dos elementos para cada par clave/valor. Por ejemplo, (1: 2, 3: 4) cuenta como (1 2, 3 4).

Los mapas permiten utilizar cualquier valor de Sass como clave. El ==operador se utiliza para determinar si dos claves son iguales.

⚠ ¡Atención!

La mayoría de las veces, es una buena idea utilizar cadenas entre comillas en lugar de cadenas sin comillas para las claves de mapa. Esto se debe a que algunos valores, como los nombres de colores, pueden *parecer* cadenas sin comillas, pero en realidad son de otro tipo. Para evitar problemas de confusión en el futuro, ¡solo use comillas!

Usando mapas

Dado que los mapas no son valores CSS válidos, no hacen mucho por sí solos. Por eso, Sass ofrece una serie de funciones para crear mapas y acceder a los valores que contienen.

Busque un valor

Los mapas se basan en la asociación de claves y valores, por lo que, naturalmente, existe una forma de obtener el valor asociado a una clave: la map.get(\$map, \$key)función!. Esta función devuelve el valor del mapa asociado a la clave dada. Devuelve nullsi el mapa no contiene la clave.

Patio de juegos

SINTAXIS SCSS

```
@use "sass:map";
$font-weights: ("regular": 400, "medium": 500, "bold": 700);

@debug map.get($font-weights, "medium"); // 500
```

```
@debug map.get($font-weights, "extra-bold"); // null  
Haz algo por cada par
```

En realidad, esto no utiliza una función, pero sigue siendo una de las formas más comunes de utilizar mapas. La [@eachregla](#) evalúa un bloque de estilos para cada par clave/valor en un mapa. La clave y el valor se asignan a variables para que se pueda acceder a ellos fácilmente en el bloque.

Patio de juegos

SINTAXIS SCSS

```
$icons: ("eye": "\f112", "start": "\f12e", "stop": "\f12f");  
  
@each $name, $glyph in $icons {  
  .icon-#$name:before {  
    display: inline-block;  
    font-family: "Icon Font";  
    content: $glyph;  
  }  
}
```

SALIDA CSS

```
.icon-eye:before {  
  display: inline-block;  
  font-family: "Icon Font";  
  content: "\f112";  
}  
  
.icon-start:before {  
  display: inline-block;  
  font-family: "Icon Font";  
  content: "\f12e";  
}  
  
.icon-stop:before {  
  display: inline-block;  
  font-family: "Icon Font";  
  content: "\f12f";  
}  
Agregar a un mapa
```

También resulta útil para agregar nuevos pares a un mapa o para reemplazar el valor de una clave existente. La [map.set\(\\$map, \\$key, \\$value\)función](#) hace lo siguiente: devuelve una copia de \$map con el valor en \$key establecido en \$value.

Patio de juegos

SINTAXIS SCSS

```
@use "sass:map";  
  
$font-weights: ("regular": 400, "medium": 500, "bold": 700);  
  
@debug map.set($font-weights, "extra-bold", 900);  
// ("regular": 400, "medium": 500, "bold": 700, "extra-bold": 900)  
@debug map.set($font-weights, "bold", 900);  
// ("regular": 400, "medium": 500, "bold": 900)
```

En lugar de establecer valores uno por uno, también puedes fusionar dos mapas existentes usando `map.merge($map1, $map2)`.

Patio de juegos

SINTAXIS SCSS

```
@use "sass:map";  
  
$light-weights: ("lightest": 100, "light": 300);  
$heavy-weights: ("medium": 500, "bold": 700);  
  
@debug map.merge($light-weights, $heavy-weights);  
// ("lightest": 100, "light": 300, "medium": 500, "bold": 700)
```

Si ambos mapas tienen las mismas claves, los valores del segundo mapa se utilizan en el mapa que se devuelve.

Patio de juegos

SINTAXIS SCSS

```
@use "sass:map";  
  
$weights: ("light": 300, "medium": 500);  
  
@debug map.merge($weights, ("medium": 700));  
// ("light": 300, "medium": 700)
```

Tenga en cuenta que los mapas Sass son inmutables y no modifican la lista original.`map.set()``map.merge()`

Inmutabilidad

Los mapas en Sass son *inmutables*, lo que significa que el contenido de un valor de mapa nunca cambia. Todas las funciones de mapas de Sass devuelven nuevos mapas en lugar de modificar los originales. La inmutabilidad ayuda a evitar muchos errores ocultos que pueden aparecer cuando el mismo mapa se comparte en diferentes partes de la hoja de estilo.

Sin embargo, aún puedes actualizar tu estado a lo largo del tiempo asignando nuevos mapas a la misma variable. Esto se suele usar en funciones y mixins para realizar un seguimiento de la configuración en un mapa.

Patio de juegos

SINTAXIS SCSS

```
@use "sass:map";  
  
$prefixes-by-browser: ("firefox": moz, "safari": webkit, "ie": ms);  
  
@mixin add-browser-prefix($browser, $prefix) {  
  $prefixes-by-browser: map.merge($prefixes-by-browser, ($browser: $prefix)) !global;  
}  
  
@include add-browser-prefix("opera", o);  
@debug $prefixes-by-browser;  
// ("firefox": moz, "safari": webkit, "ie": ms, "opera": o)
```