# Cómo organizar los archivos Sass de tu proyecto

Diferentes propuestas para organizar las carpetas de tus archivos SASS

Gerardo Fernández May 1, 2019 · 5 min read ★


**Sass** se ha convertido en uno de mis lenguajes / preprocesadores favoritos a la hora de escribir las hojas de estilo de mis proyectos. De ahí que haya decidido preparar este artículo donde explicaros la estructura que habitualmente escojo para organizar los archivos SASS de un proyecto, ya que adoptar una buena organización nos permitirá mantener el proyecto mucho más fácilmente y beneficiarnos de la gran flexibilidad que

Por supuesto que esto es sólo una forma de trabajar con Sass (concretamente la que a mí me funciona) y no estáis obligados a llevarla al pie de la letra. Sentíos libres de introducir las modificaciones y mejoras que más se adapten a vuestra forma de trabajar.

Dicho ésto, comencemos.

## Empleando Partials

Una de las múltiples ventajas que nos proporciona Sass es **poder dividir nuestras hojas de estilo en pequeños archivos** (denominados *partials)* que posteriormente podemos importar en nuestras hojas de estilo principales mediante la expresión `@import .`

Por ejemplo, podemos contar con un archivo principal denominado `main.sass`

```
// Archivo: main.sass

@import 'layout/header';
```

y un archivo *partial* denominado `_header.sass` dentro de la carpeta `layout` que contenga el estilo de nuestra cabecera:

```
// File: _header.sass

.the-header
  // estilos del header
```

Es importante destacar que el nombre de un archivo *partial* debe comenzar por `_` tal y como podéis ver en el archivo `_header.sass` .

## Una estructura sencilla

En el caso de que estéis trabajando en un proyecto pequeño seguramente no necesitéis una estructura compleja para organizar vuestros archivos. Por tanto, podéis adoptar

```
_base.sass
_layout.sass
_components.sass
/components
  _buttons.sass
  _jumbo.sass
main.sass
```

Como veis, tenemos un archivo principal denominado `main.sass` que importará tres *partials* principales:

- `_base.sass` , donde aparecerán los *resets*, las variables, los *mixins* que necesitemos y las clases de utilidades
- `_layout.sass` , en donde se encontrarán las clases relacionadas con el layout de nuestra aplicación como el *grid* que estemos empleando
- `_components.sass` , que se encargará de importar los componentes que necesitemos de la carpeta `components` . Estos componentes se corresponden con elementos como botones, barra de navegación, *jumbos*…

Con esto contaremos con una estructura básica para nuestro proyecto. Ahora bien, si éste comienza a crecer es probable que nos veamos obligados a extraer a nuevas *partials* determinados elementos para evitar archivos excesivamente largos tal y como hemos hecho con nuestra carpeta `components` . Es justo en este punto donde veo muy importante escoger una buena estructura. A continuación os explicaré mi favorita.

## La regla del 7+1

También conocida como **patrón 7–1** es una estructura ampliamente estandarizada como forma de organizar proyectos complejos. Consiste en tener todas los *partials* organizados en 7 carpetas y un único archivo en la raíz para gestionar todos los *imports* necesarios.

Veamos un ejemplo para a continuación explicaros lo que se encuentra dentro de cada carpeta

```
|- utilities/
|   |- _variables.sass    // Variables
|   |- _functions.sass    // Funciones
|   |- _mixins.sass       // Mixins
|
|- base/
|   |- _reset.sass        // Reset/normalize
|   |- _typo.sass   // Tipografías
|
|- components
|   |- _buttons.acss      // Bot
|   |- _jumbo.sass     // Jumbo
|
|- layout/
|   |- _navigation.sass   // Navegación
|   |- _grid.sass         // Grid
|   |- _header.sass       // Header
|   |- _footer.sass       // Footer
|   |- _forms.sass        // Formularios
|
|- views/
|   |- _home.sass         // Estilos de la página de inicio
|   |- _contact.sass      // Estilos de la página de contacto
|
|- themes/
|   |- _theme.sass        // Default theme
|   |- _admin.sass        // Admin theme
|
|- vendors/
|   |- _bootstrap.sass    // Bootstrap
|   |- _jquery-ui.sass    // jQuery UI
|
|- main.sass              // Archivo principal
```

- **`utilities`** . En esta carpeta se encuentran las funciones, variables y *mixins* que emplearemos en el resto de ficheros. Estos archivos no generan ninguna regla CSS al ser compilados sino que añaden funcionalidad extra para ser empleada por los otros *partials*.

- **`base`** . Contiene el código base de nuestro proyecto como es las reglas de tipografía (y fuentes adicionales) así como los resets necesarios para sobrescribir las reglas por defecto de los navegadores.

- **`components`** . Contiene los estilos de los distintos componentes que emplearemos dentro de nuestra aplicación como pueden ser botones, jumbos, sliders, carruseles… Separar cada componente en su propio archivo nos permitirá reutilizarlos en otros proyectos.

así como los estilos de la cabecera, el footer…

- **`views`** . Contiene los estilos específicos para determinadas vistas (páginas) de nuestro proyecto como puede ser la página de inicio o la de contacto.
- **`themes`** . Seguramente esta sea la carpeta que más os ha llamado la atención ya que su uso no es muy común. Su razón de ser es la de almacenar los estilos referidos a diferentes *themes* que pueda adoptar nuestro proyecto en función, por ejemplo, del tipo de usuario o la sección que esté visualizándose.
- **`vendors`** . Esta carpeta tiene dos opciones en función de si estáis empleando **webpack** o no.

En el caso de que estéis compilando mediante **webpack**, es probable que vuestros *vendors* residan en la carpeta `/node_modules` , de modo que en nuestra carpeta `/assets/vendors` lo que haremos será alojar *partials* encargados de importar los archivos de `/node_modules` (si no necesitamos realizar ninguna modificación sobre los archivos) y de sobrescribir aquellas partes que necesitemos (no es buena idea modificar directamente las librerías de terceros ya que perderemos los cambios cuando las actualicemos).

Por el contrario, si no disponéis de la carpeta `node_modules` podéis instalar las librerías de terceros directamente en la carpeta `assets/vendors` y añadir dentro de ella otra carpeta donde se encuentren los archivos *partials* que las sobrescriban.

Finalmente, el archivo **`main.sass`** , situado en la raíz del proyecto, importará todos los archivos situados dentro de cada carpeta sin contener código sass adicional. Por ejemplo:

```
@import 'utilities/variables';
@import 'utilities/functions';
@import 'utilities/mixins';

@import 'utilities/bootstrap';
@import 'vendors/jquery-ui';

@import 'base/reset';
@import 'base/typo';
```

```
@import 'layout/footer';
@import 'layout/forms';

@import 'components/buttons';
@import 'components/jumbo';

@import 'pages/home';
@import 'pages/contact';

@import 'themes/theme';
@import 'themes/admin';
```

## Conclusión

Como veis, esta estructura nos permite mantener nuestros archivos organizados a la vez que nos permite modularizar suficientemente nuestro proyecto de cara a poder reutilizar cualquiera de sus partes en otros sitios.

## ¿Quieres recibir más artículos como este?

Suscríbete a nuestra newsletter:

**Latte and code**

Estás a punto de suscribirte a la newsletter de Latte and Code que recibirás cada domingo. — Los dos últimos artículos...

eepurl.us20.list-manage.com

Sass Frontend Development Front End Development Frontend CSS