Material complementario

CODERHOUSE

Clase 12- SASS I

PREPROCESADORES CSS

Un preprocesador de CSS es una **herramienta de desarrollo** que nos permite **escribir *pseudocódigo CSS.*** Este **luego** será **compilado** para **convertirse en CSS** tal y como lo conocemos. Está formado por **variables, condicionales, bucles o funciones,** elementos habituales de cualquier **lenguaje de programación**. Por este motivo, podríamos decir que tenemos un **lenguaje de programación cuya misión es la de generar el código CSS**.

El circuito de trabajo con los preprocesadores de CSS está formado por 2 pasos. En primer lugar se crean los estilos utilizando la sintaxis del preprocesador. En el segundo paso se compila el archivo para generar el CSS definitivo.

El objetivo de los preprocesadores es el de disponer de un código más limpio y sencillo de mantener y editar.

SASS

Sass significa “Syntactically Awesome Stylesheets”, es una herramienta escrita en lenguaje Ruby, que nos permite crear hojas de estilos estructuradas, limpias y fáciles de mantener.

Con SASS vamos a poder escribir hojas de estilo que nos ayudarán a generar ficheros CSS más optimizados, incorporando mayor contenido semántico y permitiendo utilizar funcionalidades que normalmente encontraríamos en lenguajes de programación tradicionales, como el uso de variables, creación de funciones, etc.

¿Por qué es útil?

Normalmente crear una hoja de estilos es relativamente sencillo. Lo malo es cuando el proyecto va creciendo en tamaño: su CSS puede acabar siendo muy extenso.

Sass nos permite una sintaxis más simple, más elegante, implementando además bastantes características extras para hacer más manejable nuestra hoja de estilos.

Inicializar NPM

NPM es el Gestor de Paquetes de Node para JavaScript, que facilita instalar y desinstalar paquetes de terceros. Para inicializar un proyecto en Sass con NPM, abre tu terminal y CD (cambia de directorio) a tu directorio de proyecto.

Una vez en el directorio correcto, ejecuta el comando *npm init*. Se te pedirá responder varias preguntas sobre el proyecto, después de las cuales NPM generará un archivo

package.json en tu directorio.

$ npm init

En este punto, es muy importante que realices un backup (copia de resguardo) de tus archivos CSS, ya que se sobreescribirán cuando empieces a usar SASS.

Instalar sass

sass se usará para compilar tus archivos *scss* en archivos *css*. Para instalarlo de manera global, ejecutaremos el siguiente comando en la terminal:

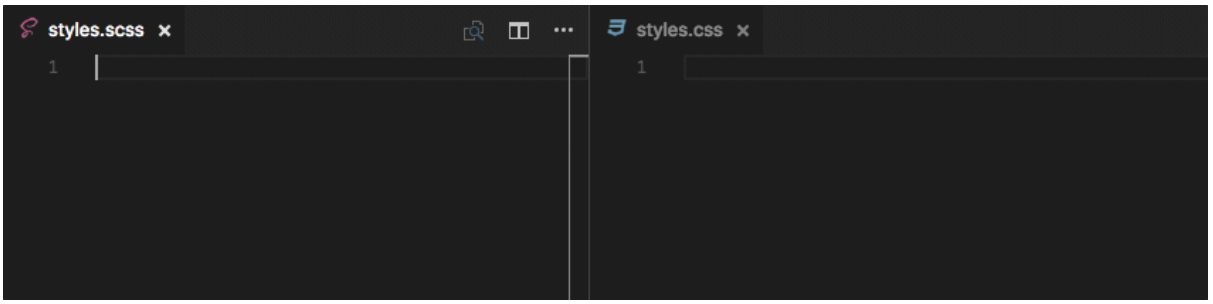
$ npm install -g sass

Comando para compilar

Para empezar a compilar, necesitamos ejecutar el siguiente comando en la terminal:

$ sass --watch scss:css

¡Y *voila*! Estamos vigilando y compilando SASS.



Más información acerca de NODEJS y NPM:
https://nodejs.org/en/about/
https://devcode.la/blog/que-es-npm/
https://www.w3schools.com/nodejs/nodejs_npm.asp
https://www.sitepoint.com/beginners-guide-node-package-manager/

Sintaxis SASS

En Sass contamos con dos diferentes tipos de sintaxis: scss y sass.

La **primera** y más popular es conocida como **SCSS** (Sassy CSS), es muy similar a la sintaxis nativa de CSS, tanto así que nos permite importar hojas de estilos CSS (copiar y pegar) directamente en un archivo SCSS y obtener un resultado válido.

Para utilizarla solo debemos crear un archivo con terminación .scss de la siguiente manera: *archivo.scss*

La **segunda** opción es conocida como **Indented Syntax** (sintaxis de indentación). Utiliza la indentación en lugar de corchetes para expresar el anidamiento de selectores y saltos de línea en lugar de (;) punto y coma para separar las diferentes propiedades que se declaren. Usarla también es muy sencillo: creamos un archivo con terminación .sass de la siguiente manera: *archivo.sass*

Nesting, Import y Vars

Nesting o Anidación

HTML sigue una estricta estructura de anidación, mientras que CSS por lo general, es un caos total. Con **la anidación de Sass puedes organizar tu hoja de estilo de una manera que se asemeja a la de HTML**, lo que reduce la posibilidad de conflictos en el CSS.

Echa un vistazo a este ejemplo rápido, en el que definimos una lista que contiene una serie de enlaces:

SCSS

```
ul {
 list-style: none;
      li {
           padding: 15px;
           display: inline-block;
            a {
                text-decoration: none;
                      font-size: 16px;
                color: #444;
            }
             }
```

```
}
```

CSS

```
ul {
 list-style: none;
}
ul li {
 padding: 15px;
 display: inline-block;
}
ul li a {
 text-decoration: none;
 font-size: 16px;
 color: #444;
}
```

Vars (variables)

Las variables son una manera de guardar información que necesites reutilizar en tus hojas de estilos: colores, dimensiones, fuentes o cualquier otro valor. Sass utiliza el símbolo dólar ($) al principio de la palabra clave para crear una variable.

Estas variables se comportan como atributos CSS, y su valor puede ser cualquiera que pudiera adquirir un atributo CSS.

A continuación puedes ver un sencillo ejemplo. Te mostramos tanto el código SCSS, como el CSS.

SCSS

```
$title-font: normal 24px/1.5 'Open Sans', sans-serif;
$cool-red: #F44336;
$box-shadow-bottom-only: 0 2px 1px 0 rgba(0, 0, 0, 0.2);

h1.title {
```

```
 font: $title-font;
 color: $cool-red;
}

div.container {
 color: $cool-red;
 background: #fff;
 width: 100%;
 box-shadow: $box-shadow-bottom-only;
}
```

CSS

```
h1.title {
 font: normal 24px/1.5 "Open Sans", sans-serif;
 color: #F44336;
}

div.container {
 color: #F44336;
 background: #fff;
 width: 100%;
 box-shadow: 0 2px 1px 0 rgba(0, 0, 0, 0.2);
}
```

La idea detrás de todo esto es que luego podamos **volver a utilizar los mismos valores de una manera rápida**, o si es necesario un **cambio**, podemos declarar el **nuevo valor en un solo lugar** (la definición de la variable), en lugar de aplicar el cambio manualmente a todas partes afectadas por la propiedad.

Una variable se podrá definir fuera o dentro de algún selector. Si se define fuera, dicha variable será **global** y podrá utilizarse en cualquier bloque. Si se define dentro de un selector, la variable será **local** y únicamente se podrá utilizar en el selector que la contiene y en sus selectores anidados.

Una **buena práctica** común consiste en **definir todas las variables globales al principio del fichero**, para que puedan localizarse rápidamente.

Uso de !default en las variables

Si hacemos:

```
$color: #FF0000;
$color: #000000;
```

El color que se usará es el #000000.

Pero si hacemos:

```
$color: #333333;
$color: #000000 !default;
```

El color que se usará será #333333. Esta directiva indicará que la asignación que estamos realizando a la variable solo se haga en caso de que dicha variable no se haya definido anteriormente.

Operaciones

Con Sass puedes realizar **operaciones matemáticas básicas** en la misma hoja de estilo: es tan sencillo como poner el símbolo aritmético adecuado.

SCSS

```
$ancho: 720px;
$blue: #4285F4;
$green: #33D374;
```

```
.box_uno {
 background-color: $blue;
 width: $ancho/2;
}
.box_dos {
 background-color: $green;
 width: ($ancho/2)-50;
}
.box_tres {
 background-color: $blue;
 width: ($ancho/2)+50;
}
.box_cuatro {
 background-color: $green;
 width: ($ancho/7)*2;
}
```

CSS

```
.box_uno {
  background-color: #4285f4;
  width: 360px;
}

.box_dos {
  background-color: #33d374;
```

```
  width: 310px;
}

.box_tres {
  background-color: #4285f4;
  width: 410px;
}

.box_cuatro {
  background-color: #33d374;
  width: 205.7142857143px;
}
```

Aunque *vanilla CSS* actualmente ofrece esta característica también en forma de *calc()*, la alternativa de Sass es más rápida, tiene operaciones % y puede ser aplicada a una gama más amplia de datos, como por ejemplo colores y strings.

Import

Una de las características más útiles de Sass es poder **separar tus hojas de estilo en archivos separados**. A continuación, puedes usar @import para incluir la fuente de tus archivos individuales en una hoja de estilo maestra.

¿Pero cómo debes estructurar tus proyectos Sass? ¿Hay una forma estándar de separar tus archivos CSS?

Estructura de directorio básica

```
stylesheets/
|
|-- modules/            # Modulos Comunes
|   |-- _all.scss       # Todos los modules
|   |-- _utility.scss   # Utilitarios
|   |-- _colors.scss    # Etc...
|   ...
```

```
|
|-- partials/
|   |-- _base.sass        # imports for all mixins + global project variables
|   |-- _buttons.scss     # buttons
|   |-- _figures.scss     # figures
|   |-- _grids.scss       # grids
|   |-- _typography.scss  # typography
|   |-- _reset.scss       # reset
|   ...
|
|-- vendor/               # CSS o Sass de otros plugins
|   |-- _colorpicker.scss
|   |-- _jquery.ui.core.scss
|   ...
|
`-- main.scss             # Archivo Sass principal
```

Esto nos permite mantener nuestro archivo Sass principal extremadamente limpio, importando código desde otros archivos Scss o CSS:

```
// Modules and Variables
@import "partials/base";

// Partials
@import "partials/reset";
@import "partials/typography";
@import "partials/buttons";
@import "partials/figures";
@import "partials/grids";
// ...

// Third-party
@import "vendor/colorpicker";
@import "vendor/jquery.ui.core";
```

Enlaces Relacionados:

- ✓ Sass Guidelines
- ✓ Sass
- ✓ Qué es SASS y por qué los CSS pueden volver a divertir
- ✓ Sass, el manual oficial
- ✓ Learn Sass | Codecademy