UD4. Sass

Pedro Puerma

1. Introducción

Sass es un metalenguaje de Hojas de Estilo en Cascada (CSS). Es un lenguaje de script que es traducido a CSS, SassScript es el lenguaje de script en sí mismo.

2. Instalación

Tenemos diferentes opciones:

- Si tenemos **Nodejs** Instalado:

```
npm install -g sass
```

- Si tenemos **Windows Chocolatery**:

```
choco install sass
```

- Si tenemos **Mac Homebrew**:

```
brew install sass/sass/sass
```

2. Instalación

Otra alternativa sería descargar el código de **github**:

https://github.com/sass/dart-sass/releases

3. Ventajas

Las **ventajas** de utilizar Sass son todas aquellas derivadas de ser un "**Preprocesador CSS**", es decir, básicamente usar Sass nos **facilitará el desarrollo de CSS complejos proporcionado estructuras y herramientas** para trabajar de forma más óptima y organizada..

3. Ventajas

- **Reduce el tiempo** para crear y mantener mi CSS.
- Permite una **organización modular** de mis estilos (proyectos grandes).
- Proporciona **estructuras de lenguaje de programación** (variables, listas, funciones, estructuras de control).
- Permite **generar distintos tipos de salidas** (comprimida, normal, minimizada)
- Permite regenerar **salida** (CSS) de **manera automática** (modo watch)
- **Herramientas, librerías, comunidad** etc.…

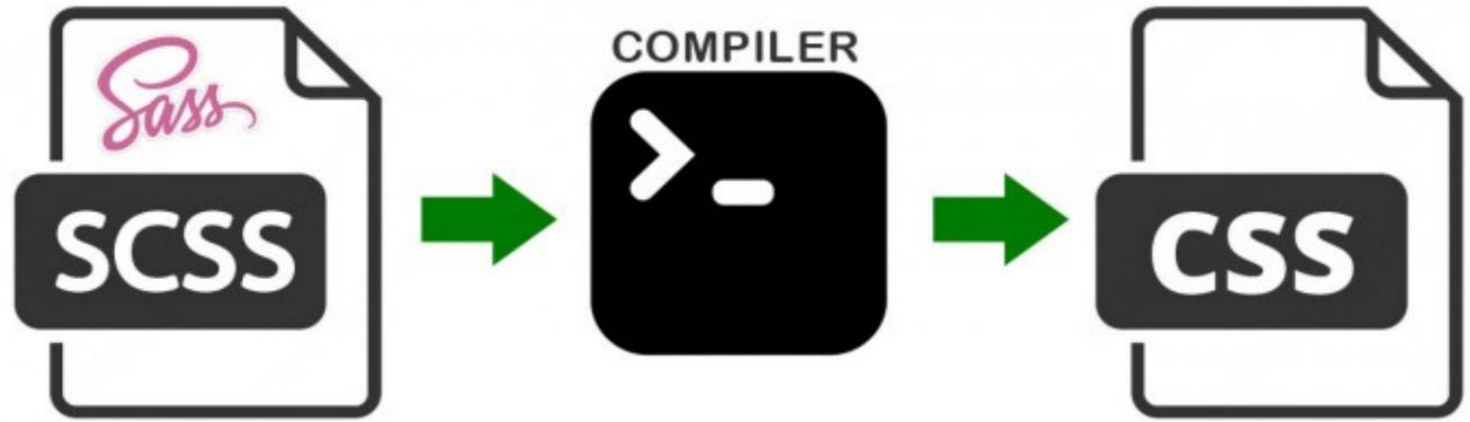
4. Desventajas

Las **desventajas** de usas Sass son las mismas que aparecen al utilizar cualquier preprocesador:

- ¿Necesidad de **aprender una nueva herramienta**? ¿Desventaja?.
- Se **consume un tiempo** en la **compilación** para obtener nuestros estilos (la transformación de SCSS a CSS).
- **Sintaxis** más **compleja** que CSS.

5. Compilación

Existen **diferentes tipos** de Compilación:



5. Compilación

- Simple
- Múltiple (varios ficheros)
- Expandida (por defecto)
- Comprimida
- Vigilando los cambios
- ……Y muchas más posibilidades incluyendo distintos flags

5. Compilación

SIMPLE

Sass *entrada.scss* *salida.css*

MULTIPLE

Sass file1.scss:output_file1.css … fileN.scss:output_fileN.css …

COMPRIMIDA (Quita la mayor cantidad de caracteres posible)

Sass --style=compressed file.scss output_file.css

VIGILANDO LOS CAMBIOS Y ACTUALIZANDO FICHEROS

Sass --watch file.scss output_file.css

6. Gestión de hojas de estilo

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** .

6. Gestión de hojas de estilo

Por ejemplo:

- Podemos contar con un archivo principal denominado main.sass

```
// Archivo: main.sass

@import 'layout/header';
```

6. Gestión de hojas de estilo

Por ejemplo:

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

```
// Archivo: _header.sass

.mi-header{
    color : blue;
    // estilos del header
}
```

6. Gestión de hojas de estilo

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

7. Variables

Permiten **almacenar valores reutilizables** en las hojas de estilo. Se definen con el símbolo **$** seguido del nombre de la variable y luego se le asigna un valor. Estas variables pueden contener valores como colores, números, cadenas de texto, etc.

```
$nombre: valor;
```

7. Variables

Ejemplo:

Estilo SCSS:

```scss
// Definición de variables
$color-primario: #3498db;
$color-secundario: #e74c3c;
$tamano-fuente: 16px;
$espaciado: 20px;

// Uso de variables
.encabezado {
  background-color: $color-primario;
  font-size: $tamano-fuente;
  padding: $espaciado;
}
.boton {
  background-color: $color-secundario;
  font-size: $tamano-fuente;
  margin: $espaciado;
}
```

Compila a:

```css
.encabezado {
  background-color: #3498db;
  font-size: 16px;
  padding: 20px;
}

.boton {
  background-color: #e74c3c;
  font-size: 16px;
  margin: 20px;
}
```

7. Variables

Debemos tener precaución con el ámbito de las variables:

```
// Variable Global
$logo-width : 50%;

.header {
//Variable Local
 $header-width : 50%
}
```

8. Comentarios

Sass admite **dos tipos** de comentarios:

- Comentarios de una sola línea `// ESTO ES UN COMENTARIO`
- Comentarios multilínea `/* ESTO TAMBIÉN ES UN COMENTARIO */`

8. Comentarios

Las recomendaciones en el uso de comentarios:

- Comenta todo lo que no sea evidente.
- Es importante!! “No comentar demasiado”.
- Describe las funciones.
- Explica las agrupaciones de las reglas y sus objetivos

9. Listas

Sass nos proporciona dos tipos de **datos más complejos** como son las listas y los mapas:

- Listas → **secuencia ordenada** de valores, como números, colores o cadenas de texto. Los valores **se pueden separar por comas o no**. El uso de **comillas** para los valores es solo necesario cuando se **usan caracteres especiales**.
- Mapas → colecciones de **pares clave-valor**. Se definen **utilizando paréntesis ()** y separando cada par clave-valor con dos puntos.

9. Listas

Definición:

```
$colores: red, green, blue, yellow; // Valores
separados con comas
$colores: red green blue yellow; // Valores
separados sin comas
$colores: 'red' 'green' 'blue' 'yellow'; //
Valores con comillas
```

Acceso:

```
$primer-color: nth($colores, 1); // red
$segundo-color: nth($colores, 2); // green
```

9. Listas

También puedes usar funciones como las siguientes:

Función	Descripción	Ejemplo
length($list)	Devuelve la cantidad de elementos en la lista $list.	length(1 2 3) retorna `3`
index($list, $value)	Retorna el primer índice en el que se encuentra $value en $list. Si no encuentra nada, retorna false.	index(apples oranges bananas, oranges) retorna `2`
join($list1, $list2[, $separator])	Combina $list1 y $list2 usando `$separator` como separador en la lista resultante. Por defecto toma el separador de $list1.	join(1 2 3, 4 5 6, comma) retorna `1, 2, 3, 4, 5, 6`
append($list, $value[, $separator])	Agrega $value al final de $list, utilizando `$separator` como separador. Por defecto toma el separador de $list.	append(1 2 3, 4) retorna `1 2 3 4`

10. Mapas

Recordad, son colecciones de **pares clave-valor**. Se definen utilizando paréntesis () y separando cada par clave-valor con dos puntos.

```
$botones: (
  primary: #3498db,
  secondary: #e74c3c,
  success: #2ecc71
);
```

Puedes **acceder** a los valores de un mapa utilizando su clave con **map-get()**. Por ejemplo:

```
$color-primary: map-get($botones, primary); // #3498db
$color-secondary: map-get($botones, secondary); // #e74c3c
```

10. Mapas

También puedes usar funciones como **map-merge()** para **combinar** mapas, **map-keys()** para **obtener todas las claves** de un mapa y **map-values()** para obtener **todos los valores**.

11. Anidamiento

Permite crear reglas más concisas y mejor organizadas. Permite anidar selectores dentro de otros, lo que **refleja la estructura de HTML** y hace que tu código sea más legible y organizado.

Supongamos que tenemos el siguiente código HTML:

```
<div class="contenedor">
  <h1>Título</h1>
  <p>Párrafo de ejemplo</p>
</div>
```

11. Anidamiento

En Sass, **puedes anidar selectores para representar esta estructura HTML** de la siguiente manera:

Estilo SCSS:

```scss
.contenedor {
    background-color: #f0f0f0;
    padding: 20px;
    h1 {
      font-size: 24px;
      color: #333;
    }
    p {
      font-size: 16px;
      color: #666;
    }
  }
```

Compila a:

```css
.contenedor {
  background-color: #f0f0f0;
  padding: 20px;
}
.contenedor h1 {
  font-size: 24px;
  color: #333;
}
.contenedor p {
  font-size: 16px;
  color: #666;
}
```

11. Anidamiento

El símbolo «**&**» se usa para hacer **referencia al selector padre dentro** de una regla anidada. Esto es útil cuando quieres aplicar **estilos específicos a elementos que son descendientes directos del selector padre**.

11. Anidamiento

Supongamos que tenemos el siguiente código en Sass:

```
.button {
  background-color: blue;
  &:hover {
    background-color: red;
  }
}
```

El «&» en **&:hover** hace referencia al selector .button, por lo que cuando el cursor se coloca sobre un elemento con la clase .button, se aplicará el estilo **background-color: red**; solo a ese elemento en particular.

12. Ejercicio

Supongamos que tenemos el siguiente HTML:

```
<!DOCTYPE html>
<head>
    <link rel="stylesheet"
href="sass/estilos1.css">
</head>
<body>
    <main class="encabezado">
        <h1>Primeros pasos</h1>
        <p>Yuuki Puerma</p>
    </main>
    <article class="contenido">
        <h2>Prueba con SCSS</h2>
        <button class="boton">No
pulsar</button>
    </article>
</body>
</html>
```

12. Ejercicio

Replicar: