Preprocesador Sass

- C.F.G.S. DAW
- 6 horas semanales
- Mes aprox. de impartición: Ene - Feb
- iPasen - cjaedia071@g.educaand.es

Carmelo José Jaén Díaz

Índice

OBJETIVO

- Analizar y seleccionar los colores y las tipografías adecuados para la visualización en la pantalla.
- Utilizar marcos y tablas para presentar la información de manera ordenada.
- Identificar nuevos elementos y etiquetas en HTML5.
- Reconocer las posibilidades de modificar etiquetas HTML.
- Valorar la utilidad de las hojas de estilo para conseguir un diseño uniforme en todo el sitio web.

GLOSARIO

Formularios. Documentos interactivos utilizados para recoger información en un sitio web. Esta información es enviada al servidor, donde es procesada. Cada formulario contiene uno o varios tipos de controles que permiten recolectar la información de varias formas diferentes.

Fuentes seguras. Fuentes tipográficas que los usuarios tenían instaladas por defecto en su dispositivo. En la actualidad, gracias a que la mayoría de los navegadores soportan la directiva @font-face, es posible utilizar casi cualquier tipografía a través de Google Fonts.

Guías de estilo. Documentos con directrices que permiten la normalización de estilos. En estas guías se recogen los criterios y normas que debe seguir un proyecto; de esta forma se ofrece una apariencia más uniforme y atractiva para el usuario.

HTML. Lenguaje de marcado de hipertexto utilizado en las páginas web. Este tipo de lenguaje presenta una forma estructurada y agradable, con hipervínculos que conducen a otros documentos y con inserciones multimedia (sonido, imágenes, vídeos...).

GLOSARIO

HTML5. Última versión del lenguaje para la programación de páginas web HTML. Los sitios implementados con este lenguaje solo pueden visualizarse correctamente en los navegadores más actuales.

Legibilidad. Cualidad deseable en una familia tipográfica. Se trata de la facilidad de la lectura de una letra. Esta cualidad puede venir determinada por varios parámetros como el interletrado, el interpalabrado o el interlineado.

Marcos. Son las ventanas independientes incorporadas dentro de la página general. Gracias a ellos, cada página quedará dividida en varias subpáginas, permitiendo realizar un diseño más organizado y limpio a la vista. Con HTML5 ha quedado obsoleto.

Tipografía. Se trata del tipo de letra que se escoge para un determinado diseño. Según la RAE, significa ”modo o estilo en que está impreso un texto” o ”clase de tipos de imprenta”.

INTRODUCCIÓN

En este **tutorial de *Sass* (Syntactically Awesome Style Sheets)**, **preprocesador** que extiende las capacidades de **CSS (Cascading Style Sheets)**, descubrirás cómo optimizar el desarrollo de aplicaciones web, creando estilos flexibles y reutilizables.

¿QUÉ ES UN PREPROCESADOR CSS?

Un **preprocesador CSS** es una herramienta software que mejora la funcionalidad del lenguaje CSS. Los preprocesadores CSS añaden características y abstracciones que no están disponibles en CSS puro, lo que facilita la creación y el mantenimiento de hojas de estilo más complejas.

Algunos ejemplos destacados de preprocesadores CSS son ***Sass*** **(Syntactically Awesome Style Sheets)**, ***Less*** **(Leaner Style Sheets)** y ***Stylus***. Estos preprocesadores se utilizan en el desarrollo web para crear hojas de estilo más mantenibles, eficientes y organizadas. Sin embargo, los navegadores web no pueden interpretar directamente código *Sass*, *Less* o *Stylus*, por lo que estos **archivos deben compilarse en CSS estándar** antes de implementarse en un sitio web.

La compilación se realiza generalmente mediante **herramientas de línea de comandos, extensiones de editores de código o con herramientas de automatización de tareas (como Gulp o Grunt).**

VENTAJAS DE *Sass*

- **Mejora la eficiencia en la escritura de código CSS**: Sass permite escribir CSS de una manera más eficiente y organizada. Con características como variables, anidación y funciones, puedes reducir la redundancia de código, lo que hace que tus estilos sean más limpios y fáciles de mantener.

- **Reutilización de código:** Las funciones en Sass permiten reutilizar segmentos de código CSS en todo el proyecto. Esto facilita la creación y el mantenimiento de estilos coherentes en toda la aplicación o sitio web.

- **Variables:** Con Sass, puedes utilizar variables para almacenar valores que se utilizan repetidamente, como colores, tamaños de fuente o márgenes. Esto hace que sea más sencillo realizar cambios globales en el diseño, ya que solo tienes que actualizar la variable en lugar de buscar y cambiar cada instancia manualmente.

VENTAJAS DE *Sass*

- **Anidación de selectores:** Sass permite anidar selectores CSS, lo que refleja mejor la estructura HTML de tu sitio web. Esto hace que tu código sea más legible y reduce la posibilidad de conflictos de nombres.

- **Modularidad:** Sass te permite dividir tu código en módulos o archivos separados, lo que facilita la gestión y organización de estilos para proyectos grandes y complejos.

- **Facilita el mantenimiento:** La capacidad de dividir tu código en archivos más pequeños y la reutilización de código hacen que sea más fácil realizar cambios y actualizaciones en tu sitio web sin tener que revisar y editar manualmente todo el CSS.

VENTAJAS DE *Sass*

- **Compatibilidad con CSS existente:** Puedes incorporar Sass en proyectos CSS existentes gradualmente. No necesitas reescribir todo tu código, sino que puedes comenzar a usar Sass en nuevas secciones o estilos, lo que facilita la transición.

- **Comunidad y herramientas:** Sass tiene una comunidad activa y una amplia gama de herramientas y librerías complementarias que pueden mejorar tu flujo de trabajo. Además, muchas empresas y proyectos de código abierto utilizan Sass.

- **Preprocesadores en demanda:** Aunque Sass es uno de los preprocesadores de CSS más populares, aprenderlo también te proporciona una base sólida para comprender otros preprocesadores como Less y Stylus.

INSTALACIÓN DE *Sass* USANDO *Node.js*

Puedes usar Sass instalando **Node.js** y el **paquete *Sass*** de la siguiente manera:

- **Instala *Node.js*:**

  Puedes descargar e instalar *Node.js* desde el sitio web oficial: Descargar Node.js.

- **Instala el paquete *Sass*:**
  - **Abre una terminal o línea de comandos:** Para comenzar, abre una terminal o línea de comandos en tu sistema.
  - **Instala Sass:** Ejecuta el siguiente comando para instalar *Sass* en tu sistema a través de `npm` (*Node Package Manager* o manejador de paquetes de node):

```
npm install -g sass
```

INSTALACIÓN DE *Sass* USANDO *Node.js*

El argumento `-g` significa *`global`*, lo que instala *Sass* como una herramienta global que puedes usar desde cualquier ubicación en tu sistema.

- **Verifica la instalación:** Después de la instalación, verifica que Sass se haya instalado correctamente ejecutando el siguiente comando:

```
sass --version
```

INSTALACIÓN DE *Sass* USANDO *Node.js*

Dart Sass es la implementación principal de *Sass*, lo que significa que recibe nuevas características antes que cualquier otra implementación.

Ahora que ya tienes la instalación hecha, puedes comenzar a compilar archivos Sass (`.scss` o `.sass`) en archivos CSS estándar. En la sección 5 de este tutorial veremos cómo compilar Sass por línea de comandos.

Puedes ver **otras formas de instalación** de *Sass* en el siguiente enlace: sass-lang.com/dart-sass/

ELEMENTOS BÁSICOS DE *Sass*

Como hemos visto, *Sass* ofrece características y funcionalidades adicionales para hacer que la escritura de hojas de estilo sea más eficiente y legible. A continuación, se presentan algunos elementos básicos de *Sass*:

ELEMENTOS BÁSICOS DE *Sass*

Variables

Las variables de *Sass* 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. Ver más sobre valores de variables.

```
$nombre: valor;
```

EJEMPLO

En el siguiente ejemplo hemos definido cuatro variables: `$color-primario`, `$color-secundario`, `$tamano-fuente`, y `$espaciado`, y luego hemos utilizado estas variables en las reglas de estilo CSS para los elementos `.encabezado` y `.boton`.

ELEMENTOS BÁSICOS DE *Sass*

Variables

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

```scss
// 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;
}
```

ELEMENTOS BÁSICOS DE *Sass*

Comentarios

Sass admite dos tipos de comentarios:

- **Comentarios de una línea:** Estos comentarios son similares a los comentarios de una línea en CSS y comienzan con `//`.

> **// Este tipo de comentarios no se incluyen en el fichero CSS al compilar.**

```
// Este es un comentario de una línea en Sass.
// No se incluye en el fichero CSS compilado
$color-primario: #3498db; // Define un color principal.
```

ELEMENTOS BÁSICOS DE *Sass*
Comentarios

Sass admite dos tipos de comentarios:

- **Comentarios de varias líneas:** Los comentarios de varias líneas son similares a los comentarios de bloque en CSS y se encierran entre `/*` y `*/`.

```
/* Este tipo de comentarios se incluyen en el fichero CSS al compilar, salvo en modo COMPRESSED */
```

```
/*! Este tipo de comentarios se incluyen en el fichero CSS al compilar, TAMBIÉN en modo COMPRESSED */
```

ELEMENTOS BÁSICOS DE *Sass*

Comentarios

```
/* Este es un comentario de
varias líneas en Sass.
Puede abarcar varias líneas de texto.
Se incluye en el fichero CSS al compilar, salvo en modo COMPRESSED*/
.elemento-desactivado {
   display: none;
}
```

Nota: *El modo COMPRESSED elimina la mayor cantidad de caracteres posibles. Se verá en próximas secciones del tutorial.*

ELEMENTOS BÁSICOS DE *Sass*

Anidamiento

El anidamiento es una característica de Sass que te permite **anidar selectores dentro de otros**, lo que refleja la estructura de HTML y hace que tu código sea más legible y organizado. Cuando anidas selectores, los estilos aplicados al selector padre también se aplican a los elementos anidados.

Supongamos que tenemos el siguiente código HTML:

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

ELEMENTOS BÁSICOS DE *Sass*

Anidamiento

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

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

ELEMENTOS BÁSICOS DE *Sass*

Anidamiento

En este ejemplo:

- `.contenedor` es el selector principal.
- `h1` y `p` están anidados dentro de `.contenedor`.
- Los estilos aplicados a `.contenedor` también se aplican a todos los elementos anidados dentro de él, a menos que se especifiquen estilos diferentes.

Esto se compila en el siguiente CSS:

```
.contenedor {
  background-color: #f0f0f0;
  padding: 20px;
}
```

```
.contenedor h1 {
  font-size: 24px;
  color: #333;
}
```

```
.contenedor p {
  font-size: 16px;
  color: #666;
}
```

ELEMENTOS BÁSICOS DE *Sass*

Uso del símbolo &

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.

Supongamos que tienes 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.

ELEMENTOS BÁSICOS DE *Sass*

Listas en Sass

Las listas en Sass son una **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. Ejemplos:

```
$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
```

Puedes acceder a los elementos individuales de una lista utilizando índices, comenzando desde 1. Por ejemplo:

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

ELEMENTOS BÁSICOS DE *Sass*

Listas en Sass

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`

ELEMENTOS BÁSICOS DE *Sass*

Listas en Sass

También puedes usar funciones como las siguientes:

Función	Descripción	Ejemplo
`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`

Estas funciones son útiles para realizar diversas operaciones en listas *Sass*, como obtener su longitud, buscar elementos específicos, combinar listas y agregar elementos al final de una lista existente.

ELEMENTOS BÁSICOS DE *Sass*

Mapas en Sass

Los mapas en Sass son **colecciones de pares clave-valor**. Se definen utilizando paréntesis `()` y separando cada par clave-valor con dos puntos `:`. Ejemplo:

Supongamos que tenemos el siguiente código HTML:

```
$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
```

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.

ELEMENTOS BÁSICOS DE *Sass*
Estructuras de control

Las estructuras de control son **estructuras típicas de lenguajes de programación** que nos van a permitir desarrollar CSS de una manera óptima, más organizada y reusable.

- `@if`: Permite aplicar estilos condicionalmente en función de una expresión booleana. Por ejemplo:

```
$color: blue;
.element {
      @if $color == blue {
        background-color: $color;
      } @else {
        background-color: red;
      }
}
```

RESULTADO

```
.element {
  background-color: blue;
}
```

ELEMENTOS BÁSICOS DE *Sass*
Estructuras de control

Las estructuras de control son **estructuras típicas de lenguajes de programación** que nos van a permitir desarrollar CSS de una manera óptima, más organizada y reusable.

- `@for`: Permite crear bucles `for` para generar reglas CSS repetitivas. Por ejemplo:

```
@for $i from 1 through 3 {
      .element-#{$i} {
        font-size: 10px * $i;
      }
}
```

RESULTADO

```
.element-1 {
  font-size: 10px;}
.element-2 {
  font-size: 20px;}
.element-3 {
  font-size: 30px;}
```

ELEMENTOS BÁSICOS DE *Sass*
Estructuras de control

Las estructuras de control son **estructuras típicas de lenguajes de programación** que nos van a permitir desarrollar CSS de una manera óptima, más organizada y reusable.

- `@each`: Utilizado para iterar sobre listas o mapas y aplicar estilos a cada elemento. Por ejemplo:

```
$colors: red, green, blue;
@each $color in $colors {
      .element-#{$color} {
        background-color: $color;
      }
}
```

RESULTADO

```
.element-red {
  background-color: red;}
.element-green {
  background-color: green;}
.element-blue {
  background-color: blue;}
```

ELEMENTOS BÁSICOS DE *Sass*
Estructuras de control

Las estructuras de control son **estructuras típicas de lenguajes de programación** que nos van a permitir desarrollar CSS de una manera óptima, más organizada y reusable.

- `@while`: Permite crear bucles `while` basados en una condición. Por ejemplo:

```
$i: 1;
@while $i < 4 {
      .element-#{$i} {
        width: 100px * $i;
      }
      $i: $i + 1;
}
```

RESULTADO

```
.element-1 {
  width: 100px;}
.element-2 {
  width: 200px;}
.element-3 {
  width: 300px;}
```

ELEMENTOS BÁSICOS DE *Sass*

Funciones

Sass proporciona una serie de funciones incorporadas que puedes utilizar para realizar cálculos y manipulaciones de datos en tus estilos. Veamos las más utilizadas:

- `lighten($color, $amount)`: Aclara un color al aumentar su luminosidad en función de la cantidad especificada.

```
$color: #3498db;
$lighter-color: lighten($color,
20%); // Aclara el color un 20%
h1{ color: $lighter-color;}
```

SALIDA

```
h1 {
  color: #8bc4ea;
}
```

ELEMENTOS BÁSICOS DE *Sass*

Funciones

Sass proporciona una serie de funciones incorporadas que puedes utilizar para realizar cálculos y manipulaciones de datos en tus estilos. Veamos las más utilizadas:

- `rgba($color, $alpha)`: Cambia la opacidad de un color al agregar un valor alfa.

```
$color: #3498db;
$semi-transparent: rgba($color, 0.5);
// Cambia la opacidad a 50%
```

ELEMENTOS BÁSICOS DE *Sass*

Funciones

Sass proporciona una serie de funciones incorporadas que puedes utilizar para realizar cálculos y manipulaciones de datos en tus estilos. Veamos las más utilizadas:

- `round($number)`: Redondea un número al número entero más cercano.

```
$value: 3.7;
$rounded-value: round($value); // Redondea a 4
```

ELEMENTOS BÁSICOS DE *Sass*

Funciones

Sass proporciona una serie de funciones incorporadas que puedes utilizar para realizar cálculos y manipulaciones de datos en tus estilos. Veamos las más utilizadas:

- `map-get($map, $key)`: Obtiene el valor asociado a una clave en un mapa.

```
$colors: (primary: #3498db, secondary: #e74c3c);
$primary-color: map-get($colors, primary);
// Obtiene el color primario
```

ELEMENTOS BÁSICOS DE *Sass*

Funciones

Sass proporciona una serie de funciones incorporadas que puedes utilizar para realizar cálculos y manipulaciones de datos en tus estilos. Veamos las más utilizadas:

- `nth($list, $n)`: Obtiene el elemento en la posición `$n` de una lista.

```
$numbers: 1px 2px 3px 4px 5px;
$third-element: nth($numbers, 3);
// Obtiene el tercer elemento (3px)
```

ELEMENTOS BÁSICOS DE *Sass*

Funciones

Sass proporciona una serie de funciones incorporadas que puedes utilizar para realizar cálculos y manipulaciones de datos en tus estilos. Veamos las más utilizadas:

- `str-length($string)`: Obtiene la longitud de una cadena de texto.

```
$text: "¡Hola, mundo!";
$text-length: str-length($text);
// Obtiene la longitud de la cadena (12)
```

ELEMENTOS BÁSICOS DE *Sass*

Funciones

Sass proporciona una serie de funciones incorporadas que puedes utilizar para realizar cálculos y manipulaciones de datos en tus estilos. Veamos las más utilizadas:

- `unquote($string)`: Elimina comillas de una cadena de texto.

```
$quoted-text: '"Este es un texto entre comillas"';
$unquoted-text: unquote($quoted-text);
// Elimina las comillas ("Este es un texto entre comillas")
```

ELEMENTOS BÁSICOS DE *Sass*

Mixins

Los `mixins` permiten **definir bloques de código reutilizables** que se pueden incluir en diferentes partes de tus estilos. Los `mixins` son especialmente útiles cuando se quiere aplicar el mismo conjunto de estilos a múltiples elementos sin tener que repetir el código.

Para **definir un `mixin`** en Sass, utiliza la palabra clave **`@mixin`**, seguida del nombre del `mixin` y, opcionalmente, los parámetros que aceptará el `mixin`. Ejemplo:

```
@mixin box-shadow($x, $y, $blur, $color) {
  box-shadow: $x $y $blur $color;
}
```

En este ejemplo, hemos definido un `mixin` llamado *`box-shadow`* que acepta cuatro parámetros: `$x`, `$y`, `$blur`, y `$color`.

ELEMENTOS BÁSICOS DE *Sass*

Mixins

Para utilizar un `mixin` en tu código Sass, utiliza la palabra clave **`@include`** seguida del nombre del `mixin` y los valores para los parámetros, si es necesario. A continuación, se muestra cómo se usa el `mixin` *`box-shadow`* en una regla CSS:

```
.element {
  @include box-shadow(2px, 2px, 4px, #888);
}
```

Cuando compiles tu código Sass, esta regla se compilará en CSS con el conjunto de estilos especificado en el `mixin`:

```
.element {
  box-shadow: 2px 2px 4px #888;}
```

Mixins de *Sass*

Mixins con parámetros por defecto

Puedes asignar valores por defecto a los parámetros de un `mixin` en caso de que no se proporcionen valores cuando se llama al `mixin`. Por ejemplo:

```
@mixin box-shadow($x: 0, $y: 0, $blur: 4px, $color: #000) {
  box-shadow: $x $y $blur $color;
}
```

Mixins de *Sass*
Mixins con parámetros por defecto

Si no se proporcionan valores para los parámetros al llamar al `mixin`, los valores por defecto se utilizarán en su lugar. Por ejemplo:

```
.element {
  @include box-shadow;
}
```

El código anterior se compilará a:

```
.element {
  box-shadow: 0 0 4px #000;
}
```

Mixins de *Sass*
Mixins sin parámetros

Los `mixins` también pueden definirse sin parámetros, lo que los hace útiles para agrupar un conjunto de propiedades CSS relacionadas. Por ejemplo:

```
@mixin flexbox-center {
  display: flex;
  justify-content: center;
  align-items: center;
}
```

Luego, puedes incluir este mixin en cualquier regla que necesite centrar elementos utilizando `@include`:

```
.header {
  @include flexbox-center;}
```

ELEMENTOS BÁSICOS DE *Sass*

Directiva *@import*

La directiva `@import` se utiliza para **importar otros archivos Sass en un archivo principal.**

1. **Creación de archivos Sass:** En tu proyecto Sass, puedes dividir tu código en varios archivos según la lógica y la organización que necesites. Por ejemplo, puedes tener un archivo para variables, otro para `mixins`, uno para estilos de encabezado, otro para estilos de botones, etc.

2. **Uso de `@import`:** En el archivo principal de Sass (generalmente llamado *styles.scss*), utilizas la directiva `@import` para importar otros archivos Sass. Puedes importar archivos individuales o carpetas completas.

ELEMENTOS BÁSICOS DE *Sass*

Directiva *@import*

La regla `@import` espera como argumento el nombre del archivo a importar. Por defecto busca un archivo Sass y lo importa directamente. Ejemplos:

```
// Importa un archivo Sass individual (la extensión ".scss" es opcional)
@import '_colors';
// Importa un archivo Sass individual
@import 'scss/_colors.scss';
// Importa varios archivos
@import 'scss/_colors.scss', 'scss/_layout.scsss;
// Importa un archivo CSS individual
@import 'footer.css';
// Importa todos los archivos de una carpeta
@import 'scss/*';
```

ELEMENTOS BÁSICOS DE *Sass*
Directiva *@import*

3 **Compilación:** Cuando compilas tu archivo principal de Sass (por ejemplo, *styles.scss*), el compilador Sass combinará todos los archivos importados en uno solo y generará un archivo CSS resultante.

```
sass styles.scss styles.css
```

ELEMENTOS BÁSICOS DE *Sass*

Directiva *@extend*

La directiva `@extend` en Sass es una forma de **compartir un conjunto de propiedades y reglas CSS entre dos o más selectores**. En lugar de duplicar el código CSS idéntico en varios lugares, puedes definir un conjunto de propiedades en un selector y luego extender ese conjunto a otros selectores en los que quieras aplicar las mismas propiedades.

La sintaxis básica de `@extend` en Sass es la siguiente:

```
selector-a {
  propiedad: valor;
}
```

```
selector-b {
  @extend selector-a;
  /* Más reglas de estilo para
selector-b */
}
```

ELEMENTOS BÁSICOS DE *Sass*

Directiva *@extend*

En el ejemplo anterior, las propiedades definidas en `selector-a` se aplicarán también a `selector-b`. Esto ayuda a mantener el código limpio y evita la duplicación innecesaria de estilos.

A menudo, `@extend` se utiliza para aplicar estilos comunes a diferentes elementos, como botones o elementos con estilos similares. En el siguiente ejemplo, `primary-button` y `secondary-button` extienden el conjunto de propiedades de `button`, lo que facilita la aplicación de estilos consistentes a diferentes tipos de botones.

ELEMENTOS BÁSICOS DE *Sass*

Directiva *@extend*

```
.button {
  padding: 10px 20px;
  background-color: #3498db;
  color: #ffffff;
  text-decoration: none;
}
.primary-button {
  @extend .button;
  background-color: #e74c3c;
}
.secondary-button {
  @extend .button;
  background-color: #27ae60;
}
```

ELEMENTOS BÁSICOS DE *Sass*

Operadores aritméticos

Sass admite varios operadores aritméticos que permiten realizar cálculos matemáticos en tiempo de compilación para definir propiedades CSS de manera más dinámica. Los operadores aritméticos en Sass son los siguientes:

Operador	Descripción	Ejemplo
+	Suma	`$ancho: 100px + 50px;`
-	Resta	`$padding: 20px - 10px;`
*	Multiplica	`$columnas: 3 * 4;`
/	Divide	`$ancho-total: 300px / 2;`
%	Resto	`$resto: 9 % 4;`

ELEMENTOS BÁSICOS DE *Sass*

Operadores aritméticos

- Suma (+): Utilizado para sumar valores.

  Ejemplo:

```
$ancho: 100px + 50px; // Resultado: $ancho será igual a 150px
```

- Resta (-): Utilizado para restar valores.

  Ejemplo:

```
$padding: 20px - 10px; // Resultado: $padding será igual a 10px
```

- Multiplicación (*): Utilizado para multiplicar valores.

  Ejemplo:

```
$columnas: 3 * 4; // Resultado: $columnas será igual a 12
```

ELEMENTOS BÁSICOS DE *Sass*

Operadores aritméticos

- División (/): Utilizado para dividir valores.

  Ejemplo:

```
$ancho-total: 300px / 2; // Resultado: $ancho será igual a 150px
```

- Módulo (%): Devuelve el resto de una división.

  Ejemplo:

```
$resto: 9 % 4; // Resultado: $padding será igual a 10px
```

COMPILACIÓN DE *Sass*

La **compilación de un archivo Sass** se puede hacer de varias formas: mediante **herramientas de línea de comandos, herramientas en la nube**, con **extensiones en los editores CSS** o **herramientas de automatización de tareas como** Gulp **o** Grunt. En esta sección vamos a ver cómo compilar un archivo Sass utilizando la línea de comandos y el paquete Sass.

Requisitos previos: Asegúrate de tener **el paquete Sass instalado** en tu sistema. Puedes instalarlo siguiendo los pasos explicados anteriormente (sección 3 sobre la instalación de Sass).

COMPILACIÓN DE *Sass*
Mediante línea de comandos

Abre una ventana de **línea de comandos** (CMD en Windows o Terminal en macOS y Linux). A continuación, navega al **directorio** donde se encuentra tu archivo *Sass*. Por ejemplo utilizando el comando `cd`:

```
cd ruta/al/directorio
```

COMPILACIÓN DE Sass POR LÍNEA DE COMANDOS

Compilación simple

Una vez que te encuentres en el directorio correcto, puedes **compilar tu archivo Sass** en un archivo CSS utilizando el **comando** `sass`. Por ejemplo, si tienes un archivo Sass llamado *`estilos.scss`* y quieres compilarlo a *`estilos.css`*, ejecuta el siguiente comando:

```
sass estilos.scss estilos.css
```

Esto cogerá el archivo *`estilos.scss`*, lo compilará y generará un nuevo archivo *`estilos.css`* en el mismo directorio.

COMPILACIÓN DE Sass POR LÍNEA DE COMANDOS

Observar cambios en tiempo real

Para **observar los cambios en tiempo real** mientras trabajas en tu archivo Sass, puedes utilizar el comando *`sass --watch`* para que el compilador esté atento a cambios y compile automáticamente cuando detecte modificaciones:

```
sass --watch estilos.scss estilos.css
```

Esto compilará automáticamente *`estilos.scss`* en *`estilos.css`* cada vez que realices cambios en el archivo Sass. Una vez que hayas compilado tu archivo Sass con éxito, puedes utilizar el archivo CSS resultante en tu sitio web.

COMPILACIÓN DE Sass POR LÍNEA DE COMANDOS

Compilación de varios archivos a la vez

Puedes **compilar varios archivos Sass a la vez** usando el siguiente comando:

```
sass archivo1.scss:archivo1.css … archivoN.scss:archivoN.css
```

También puedes usar el siguiente comando para **compilar todos los ficheros** `.scss` **de un directorio**:

```
sass ./
```

COMPILACIÓN DE Sass POR LÍNEA DE COMANDOS

Compilación con compresión

Mediante la compilación con compresión **se elimina la mayor cantidad de caracteres posibles.** Se recomienda poner junto al nombre del archivo de salida la palabra `min`, por ejemplo *`archivo.min.css`* Utiliza el siguiente comando:

```
sass --style = compressed archivo1.scss archivo1.min.css
```

COMPILACIÓN DE Sass POR LÍNEA DE COMANDOS

Resumen de comandos para compilar Sass

Propósito	Comando
Compilación Simple	`sass estilos.scss estilos.css`
Observar Cambios	`sass --watch estilos.scss estilos.css`
Compilación de Varios Archivos	`sass archivo1.scss:archivo1.css … archivoN.scss:archivoN.css`
Todos los ficheros del directorio	`sass ./`
Compilación con Compresión	`sass --style = compressed archivo1.scss archivo1.min.css`

COMPILACIÓN DE Sass POR LÍNEA DE COMANDOS

Ejemplo de fichero Sass y compilación

Veamos un ejemplo con todas las características vistas hasta ahora: variables, anidamiento, comentarios, listas y mapas.

El código del ejemplo se encuentra en el siguiente enlace: **ejercicio.scss**

```scss
// Variables
$color-primario: #3498db;
$color-secundario: #e74c3c;
$color-terciario: #2ecc71;
$tamano-fuente: 16px;
$espaciado: 20px;

// Mapa de estilos de botones
$estilos-botones: (
  default: (
    fondo: $color-primario,
    color-texto: white,
  ),
  secundario: (
    fondo: $color-secundario,
    color-texto: white,
  ),
  exitoso: (
    fondo: $color-terciario,
    color-texto: white,
  ),
);

// Estilos
.contenedor {
  padding: $espaciado;
  background-color: #f0f0f0;
  h1 {
    font-size: $tamano-fuente;
    color: $color-primario;
  }
  ul {
    list-style: none;

    li {
      margin-bottom: 10px;
    }
  }
}
.boton {
  display: inline-block;
  padding: 10px 20px;
  border: 1px solid transparent;
  border-radius: 5px;
  cursor: pointer;
  &.default {
    background-color: map-get($estilos-botones, default, fondo);
    color: map-get($estilos-botones, default, color-texto);
  }
  &.secundario {
    background-color: map-get($estilos-botones, secundario, fondo);
    color: map-get($estilos-botones, secundario, color-texto);
  }
  &.exitoso {
    background-color: map-get($estilos-botones, exitoso, fondo);
    color: map-get($estilos-botones, exitoso, color-texto);
  }
  &:hover {
    border-color: darken(map-get($estilos-botones, default, fondo), 10%);
  }
}
```

COMPILACIÓN DE Sass POR LÍNEA DE COMANDOS

Ejemplo de fichero Sass y compilación

En este ejemplo:

- Hemos definido variables para colores, tamaños de fuente y espaciado.
- Hemos creado un mapa llamado `$estilos-botones` que contiene diferentes estilos para botones.
- Utilizamos anidamiento para estructurar los estilos de un `.contenedor` que contiene un encabezado (`h1`) y una lista (`ul` y `li`).
- También hemos anidado estilos para los botones y utilizado el mapa `$estilos-botones` para aplicar diferentes estilos a los botones con clases como `.default`, `.secundario` y `.exitoso`.
- Además, hemos utilizado funciones como `map-get()` para acceder a los valores en el mapa y `darken()` para ajustar el color de borde en el estado `hover`.

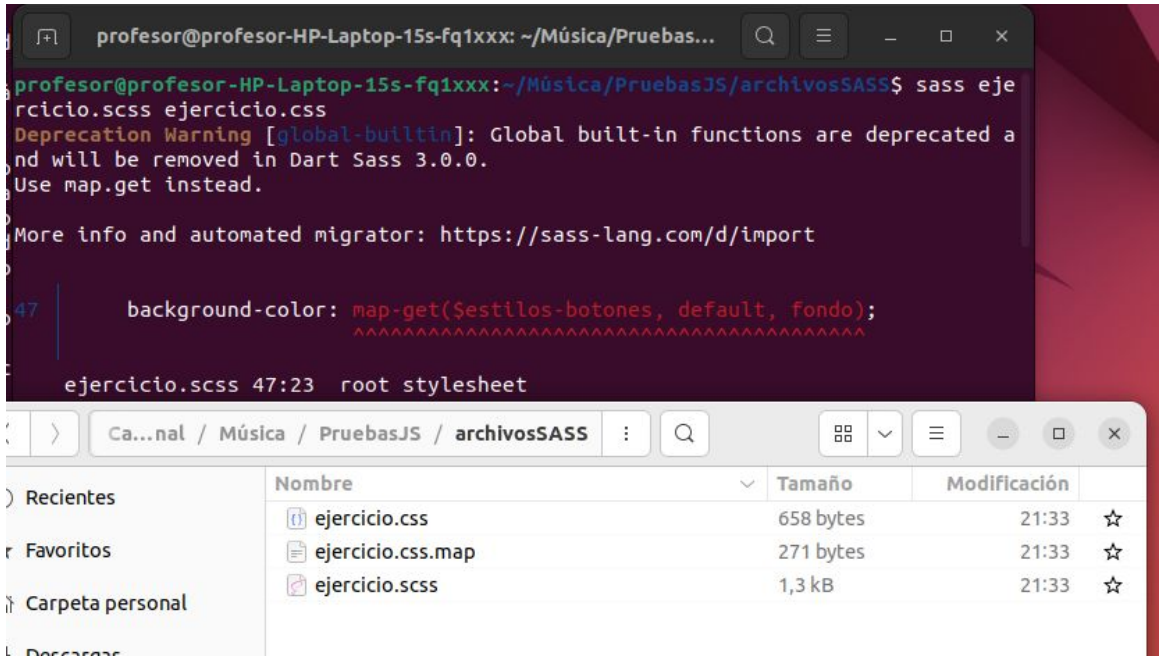
COMPILACIÓN DE Sass POR LÍNEA DE COMANDOS

Ejemplo de fichero Sass y compilación

Para compilar el archivo ejecutaremos el siguiente comando:

```
sass estilos.scss estilos.css
```

Para observar los cambios en tiempo real mientras trabajas en tu archivo Sass, puedes utilizar el comando `sass --watch`:



COMPILACIÓN DE Sass POR LÍNEA DE COMANDOS

Ejemplo de fichero Sass y compilación

El archivo generado a partir de ejercicio.scss será el siguiente:

El código se encuentra en el siguiente enlace: **ejercicio.css**

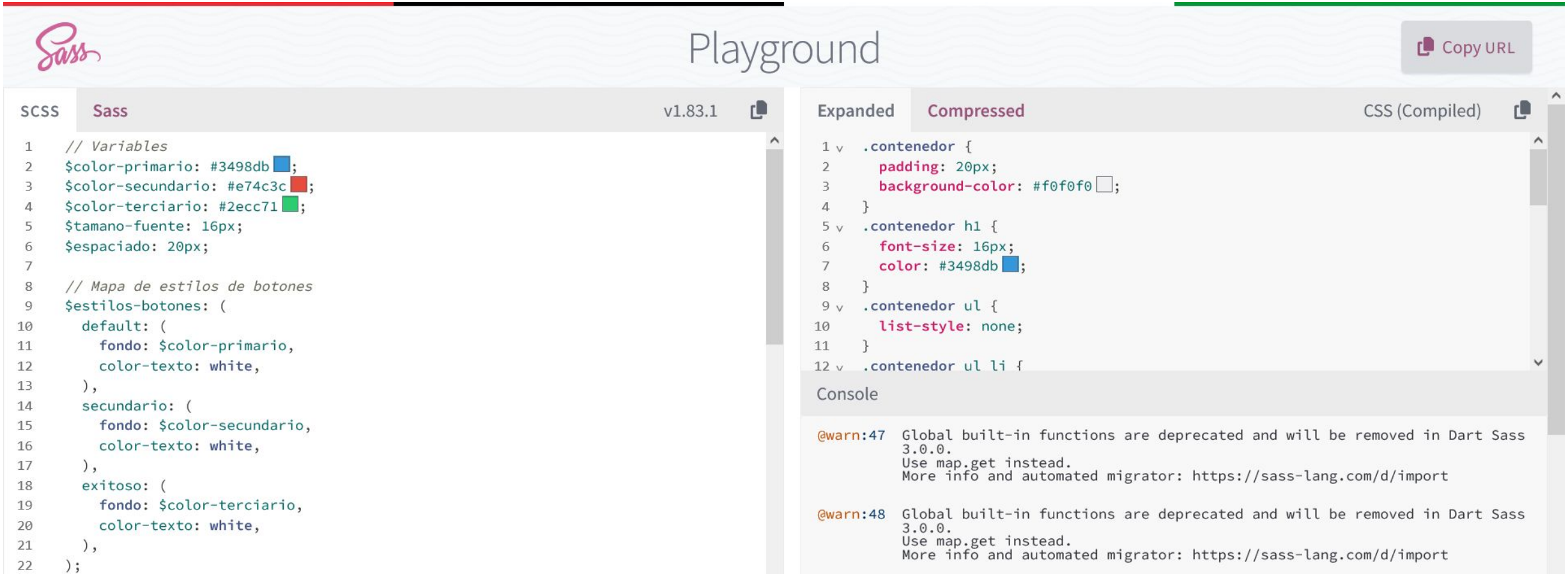
```
.contenedor {
  padding: 20px;
  background-color: #f0f0f0;
}
.contenedor h1 {
  font-size: 16px;
  color: #3498db;
}
.contenedor ul {
  list-style: none;
}
.contenedor ul li {
  margin-bottom: 10px;
}
.boton {
  display: inline-block;
  padding: 10px 20px;
  border: 1px solid transparent;
  border-radius: 5px;
  cursor: pointer;
}
.boton.default {
  background-color: #3498db;
  color: white;
}
.boton.secundario {
  background-color: #e74c3c;
  color: white;
}
.boton.exitoso {
  background-color: #2ecc71;
  color: white;
}
.boton:hover {
  border-color: #217dbb;
}
/*# sourceMappingURL=ejercicio.css.map */
```

COMPILACIÓN DE *Sass*
Mediante en un editor en línea

Los editores de Sass en línea proporcionan un entorno virtual donde los desarrolladores pueden escribir código Sass y ver los resultados de forma inmediata, lo que facilita la creación y prueba de estilos de manera interactiva. Existen numerosos editores de SCSS en línea, algunos ejemplos son los siguientes:

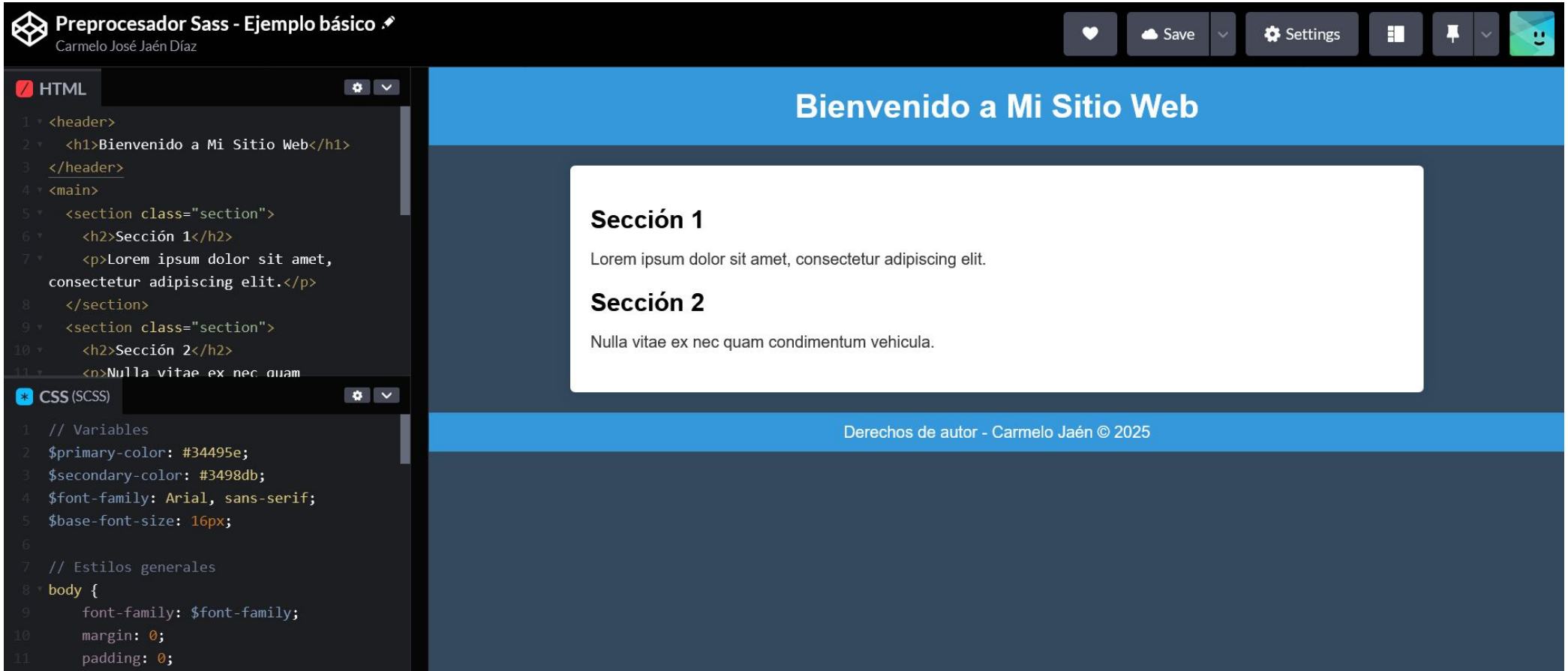
COMPILACIÓN DE *Sass*
Mediante en un editor en línea

Sass: Playground (https://sass-lang.com/playground/): Es un editor en línea específico para Sass que te permite escribir tu código en el lado izquierdo y ver el resultado compilado en CSS en el lado derecho.



COMPILACIÓN DE *Sass*
Mediante en un editor en línea

CodePen (codepen.io): CodePen es un editor en línea que admite múltiples lenguajes, incluido Sass. Puedes crear *bolígrafos* (pens) en los que escribir tu código HTML y Sass para ver el resultado visual en tiempo real.



RECOMENDACIONES *Sass*

- **Aprende las bases:** Antes de aventurarte en características más avanzadas, asegúrate de entender bien los conceptos básicos de Sass, como variables, anidación y reglas de estilo. Esto te ayudará a escribir código más limpio y comprensible.

- **Organiza tu estructura de archivos:** Utiliza una estructura de archivos organizada para tus estilos. Puedes dividir tus archivos Sass en módulos separados (por ejemplo, `_variables.scss`, `_botones.scss`, `_encabezados.scss,` etc.) y luego importarlos en un archivo principal. Esto facilitará la gestión y el mantenimiento de tus estilos.

- **Usa variables de manera inteligente:** Aprovecha las variables para almacenar valores reutilizables, como colores, tamaños de fuente y márgenes. Mantén un conjunto coherente de variables para mantener la consistencia en todo tu sitio web.

RECOMENDACIONES *Sass*

- **Evita la anidación excesiva:** Aunque la anidación de selectores puede ser útil para reflejar la estructura HTML, evita la anidación excesiva, ya que puede generar selectores CSS largos y específicos que son difíciles de mantener. Mantén la anidación a un nivel razonable.

- **Documenta tu código:** Agrega comentarios descriptivos en tu código Sass para explicar el propósito de las secciones de código o las decisiones de diseño. Esto facilitará la colaboración con otros desarrolladores y ayudará en futuras actualizaciones.

- **Usa funciones con moderación:** reserva su uso para casos donde la reutilización de código es beneficiosa.

RECOMENDACIONES *Sass*

- **Realiza pruebas y control de versiones:** Al igual que con cualquier código, realiza pruebas en diferentes navegadores para garantizar la compatibilidad. Además, utiliza un sistema de control de versiones como Git para rastrear cambios en tus archivos Sass y colaborar de manera efectiva en proyectos en equipo.

- **Mantén actualizadas tus herramientas:** Asegúrate de tener la versión más reciente de Sass y las herramientas de compilación que utilices. Las actualizaciones pueden incluir correcciones de errores y mejoras de rendimiento.

- **Explora extensiones y librerías:** La comunidad de Sass ofrece muchas extensiones y librerías que pueden ahorrarte tiempo y mejorar tu flujo de trabajo. Investiga y utiliza las que se adapten a tus necesidades.

- **Aprende y adapta:** El mundo de Sass está en constante evolución. Sigue aprendiendo y mantente actualizado con las mejores prácticas y nuevas características.

RECOMENDACIONES *Sass*

La **compilación de un archivo Sass** se puede hacer de varias formas: mediante **herramientas de línea de comandos, herramientas en la nube**, con **extensiones en los editores CSS** o **herramientas de automatización de tareas como** Gulp **o** Grunt. En esta sección vamos a ver cómo compilar un archivo Sass utilizando la línea de comandos y el paquete Sass.

Más recomendaciones en: sass-guidelin.es y sass-lang.com/documentation/