MEDAC
Instituto Oficial de Formación Profesional

DISEÑO DE INTERFACES WEB

TÉCNICO EN DESARROLLO DE APLICACIONES WEB

Preprocesadores. SASS y LESS

05

ÍNDICE

OBJETIVOS

Definir estilos de forma directa

Definir y asociar estilos globales en hojas externas

Definir hojas de estilo alternativas

Redefinir estilos

Crear clases de estilos

Validar hojas de estilo

Utilizar y actualizar guías de estilo

/ 1. Introducción y contextualización práctica

El lenguaje de hojas de estilo CSS permite trabajar con otros recursos adicionales que agilizan la programación de este tipo de archivos de estilo. Por un lado, encontramos los framework de diseño analizados en el tema 4, pero además podemos hablar de los preprocesadores CSS que permiten trabajar de forma fluida y estructurada con las hojas de estilo.

Hasta ahora si se deseaba modificar el valor de una propiedad había que revisar línea a línea de código CSS buscándola y cambiado su valor en cada aparición. Los preprocesadores van a hacer esta tarea mucho más sencilla y ágil a través del uso de variables o bloques de código. Los segundos funcionan como funciones y reciben el nombre de mixin. Algunos de los preprocesadores más habituales son: Sass y Less.

En este tema se analizará el funcionamiento de ambos, desde el proceso de instalación, que aunque sencillos, requiere de una serie de pasos muy importantes a tener en cuenta, hasta la descripción y puesta en práctica de algunos de sus elementos más característicos.

Aunque Sass y Less presentan un funcionamiento muy similar, también se establecen ciertas diferencias. En la tabla se muestran algunas de las características más destacadas.

Escucha el siguiente audio donde se plantean algunas cuestiones prácticas que te ayudarán a reflexionar sobre el uso de este tipo de componentes de diseño.

LESS	SASS
Más similar a CSS	Menos similar a CSS
Funciona bajo JavaScript	Funciona bajo Ruby
Se requiere compilación lessc en cada modificación, lado servidor	Actualización “dinámica” del fichero .css. Mejor compilación

Tabla 1. Comparativa Less y Sass

Audio intro. “¿Se podría optimizar y agilizar la programación de hojas de estilo CSS?”
https://bit.ly/2CsX0sT

/ 2. Sass

Uno de los preprocesadores más comunes en la actualidad es **Sass: Syntactically Awesome Sytle Sheets**, que significa "Hojas de Estilo Sintácticamente Increíbles" o como aparece en su sito web, "CSS con superpoderes". El framework Bootstrap 4, se basa en este preprocesador.

Hansen (1971), en su libro User Engineering Principles for Interactive Systems, hace la primera enumeración de principios para el diseño de sistemas interactivos:

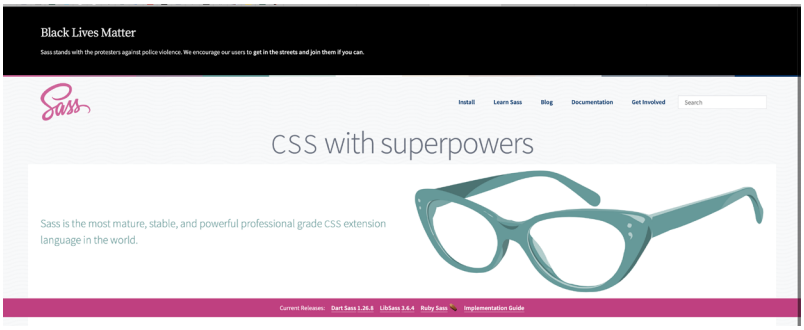


Fig. 1. Sitio web oficial Sass https://sass-lang.com

Sass es una herramienta **multiplataforma** escrita en Ruby para el desarrollo de hojas de estilo estructuradas. Presenta dos tipos de sintaxis (**Sass y SCSS**), cuya diferencia radica en ciertos aspectos de implementación en las reglas de formato que se utilizan para programar hojas de estilo en CSS. Por ejemplo, la sintaxis Sass elimina los ";" por saltos de línea o las llaves. Lo habitual es utilizar la sintaxis SCSS en Sass.

Descarga e instalación de sass

La instalación de Sass se puede realizar a través de la línea de comandos; se trata de la opción más aconsejable ya que es un proceso sencillo que permite tener instalado Sass en pocos minutos.

En primer lugar, **Sass es un gem de Ruby**, por lo que es completamente necesario instalar Ruby en el equipo donde se está realizando la programación. En función del sistema operativo la descarga e instalación de Ruby tendrá sus particularidades:

- **Windows:** basta con descargar el paquete de instalación desde este enlace, ejecutarlo e instalarlo en el equipo. https://rubyinstaller.org
- **Mac:** en el caso de Mac, Ruby aparece ya pre-instalado en el sistema, aunque si presenta una versión demasiado anticuada es conveniente actualizarla a versiones posteriores (rvm install ruby-2.7.1)
- **Linux:** habitualmente aparece preinstalado en Linux; si no, basta con introducir lo siguiente por línea de comandos en el terminal.

```
$ sudo apt-get install ruby
```

A continuación se lleva a cabo la instalación de Sass, para ello basta con introducir por línea de comando (desde cmd en el caso de Windows o desde el terminal en Mac y Linux) la siguiente instrucción:

```
$ gem install sass
```

2.1. HTML y SASS

El desarrollo en Sass se basa en la implementación del código de estilo en un documento de tipo SCSS, el cual requiere de cierta conversión a formato CSS para poder ser reconocido por un fichero en HTML. Una de las ventajas de este preprocesador es que actualiza de forma automática el contenido del fichero CSS.

Fig. 2. Diagrama de ficheros en Sass/Scss

El fichero en el que se lleva a cabo la implementación de estilo presenta la extensión .scss. Mientras, la etiqueta <link> contenida en el documento .html, requiere de una ruta de enlace a un fichero .css de la siguiente forma:

```
<link rel=”stylesheet” href=“estilos_.css”>
```

Código 1. Instrucción de enlace HTML-CSS

Por lo tanto, para realizar la vinculación con el fichero HTML es necesario generar un archivo CSS. Para ello se utiliza la instrucción “sass --watch...”, la cual, tras ser ejecutada en el terminal por línea de comandos, permanece “escuchando” si se produce algún cambio en el fichero .scss, y si es así, genera un nuevo fichero .css, al que se apunta desde HTML.

```
sass --watch ficheroSCSS.scss:ficheroCSS.css
```

Código 2. Instrucción evento escuchador de cambios en Sass

Al lanzar esta instrucción se queda esperando a que se produzca un cambio en el fichero .scss (en este caso estilos. scss) y si se modifica algo en el código y se guarda, se detecta el cambio y se almacena la nueva versión en el arhivo .css (estilos_.css).

```
>>> Sass is watching for changes. Press Ctrl-C to stop.
      write estilos_.css
      write estilos_.css.map
>>> Change detected to: estilos.scss
      write estilos_.css
      write estilos_.css.map
>>> Change detected to: estilos.scss
      write estilos_.css
      write estilos_.css.map
>>> Change detected to: estilos.scss
      write estilos_.css
      write estilos_.css.map
```

Fig. 3. Salida por terminal con detección de cambios en fichero scss

Audio 1.“Ventajas y funcionamiento dinámico de Sass”
https://bit.ly/3iZDMfa

2.2. Estructura de ficheros en SASS y variables. Estructura de carpetas y ficheros

Para trabajar desde Sass se aconseja establecer un sistema de carpetas y ficheros de estilo SCSS, creando una estructura clara que permitirá optimizar la implementación del sitio web. Tener un archivo de hoja de estilo lleno de líneas de código sin ningún tipo de orden ralentizará enormemente el diseño.

Por lo tanto, desde HTML se hace el enlace a la hoja de estilo CSS utilizando el elemento link, en concreto, al fichero destino donde se está haciendo la traducción scss-css.

```
sass --watch ficheroSCSS.scss:ficheroCSS.css

<link rel=”stylesheet” href=“ficheroCSS.css”>
```

Código 3. Resumen-comparativa fichero destino código scss-css

Desde ficheroSCSS.scss se importan el resto de ficheros de estilo, de forma que resultará más eficiente la programación, ya que HTML solo enlazará con un fichero y éste se encarga de importar los demás. El contenido de ficheroSCSS.scss podría quedar de la siguiente forma:

```
@import “config”;

@import “utilites”;

@import “mixins”;

.....
```

Código 4. Ejemplo contenido fichero base.scss para importar el resto de hojas scss

Si los cambios se producen en cualquiera de los ficheros importados desde ficheroSCSS.scss, se detectará como un cambio y se actualizará el fichero .css, que se genera automáticamente cada vez que se produzca una modificación y ésta se guarde.

Vídeo 1. “Instalación de Sass. Primeros pasos”
https://bit.ly/3eofMik

Variables en Sass: Uno de los aspectos clave e innovadores es la inclusión de variables. Sass permite crear variables en las que almacenar un valor; de esta forma, si se desea modificar el color de fondo de un determinado elemento no habría que cambiarlo en todos los elementos que apuntan a esta propiedad, sino solamente en la variable que luego es llamada por el resto de reglas de estilo. Para la creación de una variable se utiliza el símbolo $ y a continuación se indica el nombre identificativo que se le va a asignar a la variable:

```
$variable: (valor);
Selector (etiqueta|id|clase) {
        propiedad: $variable;

}
```

Código 5. Sintaxis básica variabas en Sass

/ 3. MIXIN en SASS y operadores

MIXIN

Con el nombre de mixin, se denominan a los bloques de código reutilizables, bajo los cuales se indican algunas propiedades de estilo que se repiten en diversas ocasiones a lo largo de todo el código. De esta forma, se ahorrará tiempo y costes de desarrollo.

Para implementarlos se utiliza la etiqueta @mixin seguida del nombre del bloque que se quiera indicar.

```
@mixin nombreBloque {
        propiedad_1: valor;
...
}
```

Código 6. Sintaxis básica creación de mixin en Sass

Así quedan definidos los bloques mixin. Ahora debemos incluirlos en el resto del código del fichero de estilo; y para que sea efectivo, el formato que definen se ha de agregar bajo los selectores de la siguiente forma utilizando la etiqueta @include.

```
Selector (etiqueta|id|clase){
        @include nombreBloque;
        resto de propiedades...
}
```

Código 7. Sintaxis básica llamada a mixin en Sass

Esta funcionalidad permite crear un bloque de código reutilizable, como si de una función se tratara. En el siguiente ejemplo, se ha creado un mixin con el nombre size. En este caso recibe como argumentos width y height (si se indica algo más después de los ":" indica que ese o esos argumentos son opcionales, no se tienen que declarar de forma obligatoria, sino que tomarán el valor del primer argumento).

```
@mixin sizes($width, $height: $width) {
 height: $height;
 width: $width;
}
.body {
 @include sizes(100px);
}
```

Código 8. Ejemplo completo mixin en Sass

Operadores en Sass

Sass permite la inclusión de operadores en el código CSS (+, -, *, /, %). Este tipo de elementos permite la realización de operaciones como, por ejemplo, ajustar las dimensiones de un elemento en función de un conjunto de condicionantes.

```
$dim1: 100px;

h1 {

        width: 200px/ $dim1 * 100%

}
```

Código 9. Ejemplo uso de operadores en Sass

3.1. Identificación en SASS y GRID SYSTEM

Identificación

En Sass se realiza una identificación de clases y componentes, creando así una estructura jerárquica. Este tipo de asociaciones ya está presente en CSS pero resulta algo compleja en su implementación. Sass permite realizar esta identificación indicando entre las llaves los diferentes componentes sobre los que se aplicará el estilo (en su traducción a CSS quedaría de la forma habitual).

```
Sel1 (etiqueta|id|clase) {
   propiedad1: valor1;
   Sel2 (etiqueta|id|clase){
                  propiedad2: valor2;
   }

}
```

Código 10. Sintaxis identificación en Sass

El resultado en CSS quedaría como:

```
Sel1 {

 propiedad1: valor1;

}

Sel1 Sel2{

 propiedad2: valor2;

}
```

Código 11. Sintaxis identación Sass traducida a CSS

Grid system

Finalmente, uno de los componentes más importantes es el sistema de rejilla, o sistema de columnas que incorpora Sass. Recordemos que en Bootstrap 4 la creación de columnas se realizaba en base a un sistema que denominamos de 12 columnas.

A través de los ficheros de configuración es posible crear un sistema de columnas desde cero para crear la estructura deseada. Se aconseja realizarlo sobre un fichero llamado **config.scss,** que después será importado desde el fichero base en Sass, habitualmente denominado **base.scss**, utilizando la regla **@import config.scss.**

Ahora bien, para la creación de nuestro sistema de columnas personalizado bajo el nombre escogido se utilizan los siguientes atributos:

- **col-qty.** Número total de columnas
- **gutter.** Tamaño del margen lateral de cada una de las columnas
- **width.** Valor del ancho de cada columna

El código básico final quedaría de la siguiente forma:

```
$nombreSistemaColumnas {
 col-qty: valor;
 gutter: valor;
 width: valor;
}
```

Código 12. Sintaxis básica creación de sistema de columnas en Sass

/ 4. Caso práctico 1: "Creación de un fichero de estilo con Sass"

Planteamiento: Se quiere realizar el diseño con el que ya se ha trabajado a lo largo del módulo (imagen 4), pero ahora vamos a incorporar algunos cambios utilizando Sass:

- Crea tantas variables como necesites para que en caso de querer cambiar el color de la letra o modificar alguna otra dimensión, solo necesitemos acudir a las variables comunes, buscar la deseada y modificarla.
- Crea al menos dos bloques mixin que incluyan las propiedades de estilo comunes entre la cabecera y la subcabecera, y entre el resto del texto (en el siguiente apartado encontrarás ayuda).
- Además, el tamaño del banner principal, el que aparece color azul, será siempre 5 veces mayor que el de color verde.

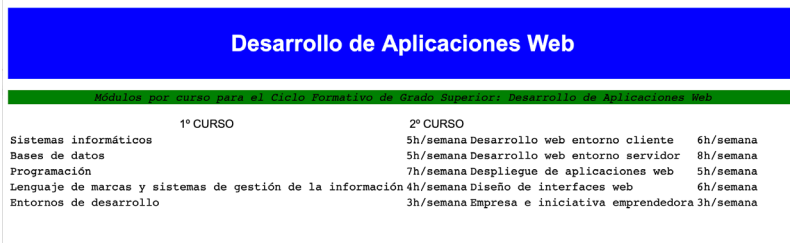


Fig. 4. Ejemplo resultado Caso Práctico 1

Nudo: En primer lugar, en el fichero HTML se mantiene el elemento link apuntando al fichero .css de forma habitual. A continuación, se lanza desde línea de comando la instrucción sass --watch estilos.scss:estilos.css, así con cada cambio en el fichero estilos.scss, se producirá la traducción equivalente en el archivo estilos.css.

Desenlace:

```
$fondo: white;
$fondoCabecera: blue;
$letra: 12;
$alturabanner_secundario: 20px;
$alineacionCabeceras: center;
$letraTitulos: arial;
$letraTexto: courier;
$colorTexto: black;

body {background-color: $fondo;}
@mixin formatoCabecerasCompartidos{
        text-align: $alineacionCabeceras;
   font-family: $letraTitulos;
}
@mixin formatoTextoCompartidos{
        color: $colorTexto;
        font-family: $letraTexto;
}
.titulo {@include formatoCabecerasCompartidos;}
.cabecera {
  background: $fondoCabecera;
  color:white;
  font: italic  bold smaller;
  height: $alturabanner_secundario * 5;
  line-height: $alturabanner_secundario * 5;
  @include formatoCabecerasCompartidos
 }
p {
        background-color: green;
        font-size:  $letra * 10;
        line-height: $alturabanner_secundario;
   font-style: oblique;
   @include formatoCabecerasCompartidos;
}
table {@include formatoTextoCompartidos;}
```

Código 13. Código .scss Caso Práctico 1

/ 5. Less

El preprocesador Less (Leaner Style Sheets), al igual que el Sass, pretende realizar una implementación de hojas de estilo CSS más eficiente. Se incluyen ciertos complementos que agilizan la programación del diseño web a través de una sintaxis propia. Después, estos ficheros son convertidos a lenguaje CSS, que HTML es capaz de entender.

Podemos decir que una de las principales diferencias entre CSS y Less es que el primero es estático y el segundo dinámico, lo que permite el uso de variables y mixin, entre otros.

Instalación de Less

La instalación de Less puede realizarse desde el lado del cliente o del servidor, siendo más recomendada la segunda, puesto que la primera puede ocasionar una disminución en el rendimiento del sitio web si JavaScript se encuentra desactivado. Por lo tanto, se describe a continuación como hacerlo desde el lado del servidor.

- En primer lugar, se realiza la descarga de Node.js desde su sitio web (https://nodejs.org/en/download/).
- Desde el terminal o cmd, en función del sistema operativo, se introduce la instrucción siguiente (recuerda hacerlo como administrador):

```
npm install -g less
```

- Ya estaría instalado. Ahora simplemente se lanza la compilación del fichero escrito en .less para ser convertido a .css.

```
lessc estilo.less > estilo.css
```

Finalmente, desde un fichero en HTML se incluiría la etiqueta link para vincular este documento con la hoja de estilo generada a partir del archivo .less.

Fig. 5. Diagrama de ficheros en Less

5.1. Variables, operadores y anidamiento en LESS

Variables y operadores

Al igual que en otros preprocesadores, Less permite la creación de variables que pueden almacenar cualquier tipo de valor. Para la creación de variables se utiliza el símbolo @:

```
@variable: (valor);

selector (etiqueta|id|clase) {

        propiedad: @variable;

}
```

Código 14. Sintaxis básica variables en Less

Este fragmento se almacenaría en un archivo .less, y tras compilarlo se obtendría un archivo .css que habría quedado de la forma que sigue (Código 15).

Como se puede observar este proceso reestructura el código en .less a la apariencia habitual de CSS, para que HTML reconozca las reglas de estilo:

```
selector (etiqueta|id|clase) {

        propiedad: valor;

}
```

Código 15. Sintaxis de variable traducida a CSS

Less permite también utilizar operadores en el código (+, -, *, /, %), que funcionan de la misma forma que la descrita en el apartado de Sass. Basta con incluir el operador entre los operandos deseados en forma de valor como llamada a una variable.

Anidamiento en Less

El **anidamiento de selectores en Less** funciona de forma similar a la identificación en Sass. Con este preprocesador es posible realizar una programación jerárquica de una forma más intuitiva que en CSS. Simplemente se tratar de ir indicando entre las llaves "{ }" los elementos a los que se va aplicar el estilo. Por ejemplo, si se va a dar un formato determinado a los enlaces que están dentro de un elemento 'párrafo', basta con ir indicando las capas que se tienen que cumplir de forma jerárquica. En el siguiente fragmento solo los enlaces que se ubican dentro de un elemento de tipo párrafo serán de color verde, el resto van a ser de color rojo.

```
p{
        font-size:12px;
 a{
                color: green;
        }
}
a{
        color: red;
}
```

Código 16. Sintaxis básica anidamiento en Less

/ 6. Mixin en less

Los mixin nos permitían realizar agrupamiento de código que luego se puede utilizar desde otro lugar del fichero .less, como si de una función se tratara. De esta forma se utilizan todos los estilos definidos dentro del mixin.

Para implementarlos se utiliza el símbolo "." que precede al nombre asignado al bloque o mixin.

```
.nombreBloque{

        propiedad_1: valor;

...

        propiedad_N: valorN;

}
```

Código 17. Sintaxis básica Mixin sin parámetro en Less

También permite recibir por parámetro (entre paréntesis a continuación del nombre) valores que se quieran considerar por defecto.

```
.nombreBloque(parámetro por defecto) {
        propiedad_1: valor;
...
        propiedad_N: valorN;
}
```

Código 18. Sintaxis básica Mixin con paso de parámetro en Less

En el siguiente ejemplo se ha creado un bloque que permite redondear las esquinas de una caja y recibe por parámetro el valor del radio que se va a utilizar para suavizar dichas esquinas.

```
.redondeador(@radio: 10px) {
        border-radius: @radio;
   -webkit-border-radius: @radio;
   -moz-border-radius: @radio;
}
```

Código 19. Ejemplo completo mixin en Less

Al llamar a este bloque desde el resto de selectores en el fichero .less (siempre precedido del símbolo ".") se utiliza el valor del nuevo parámetro si se indica, en caso contrario se utilizará el valor por defecto indicado en la creación, en este caso 10px.

```
#caja1 {
        .redondeador;
}
#caja2 {
        .redondeador(20px);
}
```

Código 20. Sintaxis ejemplo traducida a CSS

Vídeo 2. "Descarga e instalación de Less. Primeros pasos"
https://bit.ly/3iZrXWw

/ 7. Caso práctico 2: "Creación de un fichero de estilo con Less"

Planteamiento: En este segundo caso práctico se va a implementar de nuevo el diseño expuesto en el práctico 1, pero utilizando el preprocesador Less, de esta forma se podrá realizar una comparativa en cuanto a su uso. Una de las ventajas del uso de Less es que al lanzar el compilador devuelve en el terminal los errores de sintaxis línea a línea.

El planteamiento del ejercicio ya lo conocemos:

- Crea tantas variables como necesites para que en el caso de querer cambiar el color de la letra o modificar alguna dimensión, solo necesitemos acudir a las variables comunes, buscar la deseada y modificarla.

- Crea al menos dos bloques mixin que incluyan las propiedades de estilo comunes entre la cabecera y la sub-cabecera, y entre el resto del texto.

- Además, el tamaño del banner principal, el que aparece en color azul, será siempre 5 veces mayor que el de color verde.

Nudo: En primer lugar, en el fichero HTML se mantiene el elemento link apuntando al fichero .css de forma habitual.

Los cambios de estilo se realizan en el fichero con extensión .less. Cuando se completa el cambio en éste, se lleva a cabo la compilación a un fichero destino .css con el que queda enlazado el documento en HTML.

```
lessc estilo.less > estilo.css
```

Desenlace:

```
@fondo: white;
@fondoCabecera: blue;
@letra: 12;
@alturabanner_secundario: 20px;
@alineacionCabeceras: center;
@letraTitulos: arial;
@letraTexto: courier;
@colorTexto: black;

body {background-color: @fondo;}
.formatoCabecerasCompartidos{
   text-align: @alineacionCabeceras;
   font-family: @letraTitulos;
}
.formatoTextoCompartidos{
        color: @colorTexto;
        font-family: @letraTexto;
}
.titulo {.formatoCabecerasCompartidos;}
.cabecera {
  background: @fondoCabecera;
  color:white;
  font: italic  bold smaller;
  height: @alturabanner_secundario * 5;
  line-height: @alturabanner_secundario * 5;
  .formatoCabecerasCompartidos
 }
p {
        background-color: green;
        font-size:  @letra * 10;
        line-height: @alturabanner_secundario;
   font-style: oblique;
   .formatoCabecerasCompartidos;
}
table {
        .formatoTextoCompartidos;
}
```

Código 21. Código .less Caso Práctico 2

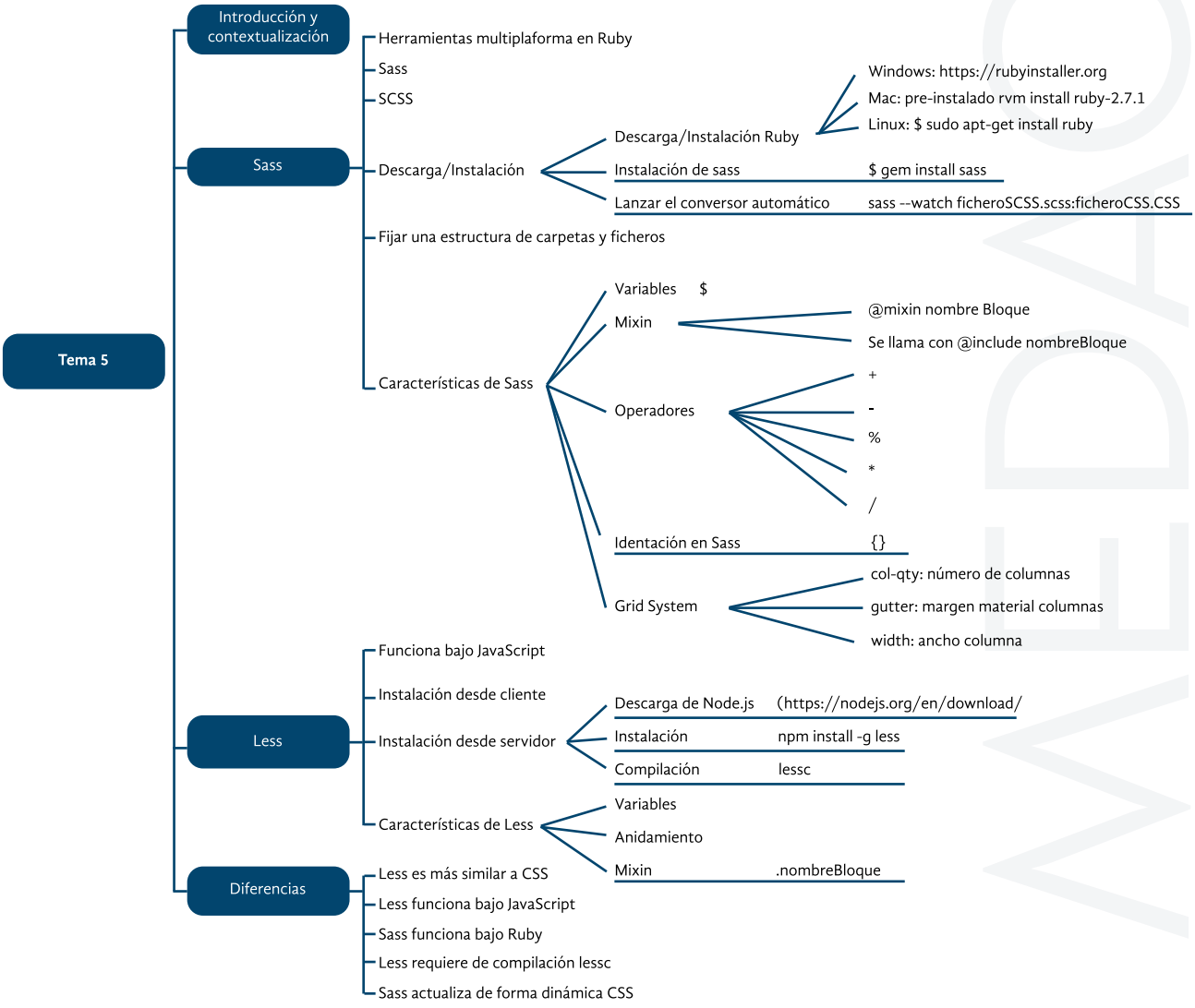
/ 8. Resumen y resolución del caso práctico de la unidad

Como se ha visto a lo largo de este tema, el uso de los preprocesadores puede agilizar en gran medida la implementación de hojas de estilo. Este tipo de programación resulta más eficiente puesto que incorpora ciertas pautas de diseño que hacen más intuitivo el desarrollo de hojas de estilo.

Tanto Sass como Less incorporan funcionalidades bastantes similares, tales como la posibilidad de utilizar operadores, la creación de variables o la de los bloques de código a través de los elementos mixin. Si bien es cierto que cambian algunos aspectos en la sintaxis de su implementación, el resultado es prácticamente el mismo.

Hemos comprobado también, que algunas de las grandes diferencias que pueden decantar al diseñador entre una u otra opción, es que mientras que Sass trabaja bajo Ruby, Less lo hace en JavaScript. Por lo tanto, la instalación de Sass resulta algo más compleja.

Además, recordemos que para la creación de ficheros .css en Sass se hace de forma automática, mientras que en el caso de Less, cuando se trabaja del lado del servidor, se requiere de la compilación a través de la línea de comandos con la instrucción lessc.



Resolución del caso práctico de la unidad

Los preprocesadores agilizan el diseño de hojas de estilo, gracias a entre otras características, la posibilidad de crear variables o bloques de código. CSS es muy simple de utilizar, lo que provoca en ocasiones que sea bastante limitado en el desarrollo. Aquí es donde aparecen Sass y Less, que nos permiten responder afirmativamente a las preguntas planteadas al inicio del tema.

Es decir, las variables nos permiten almacenar cualquier tipo de valor, de esta forma ya no será necesario revisar línea a línea de código para buscar la propiedad a modificar (lo que da respuesta a la primera cuestión planteada al inicio del tema), sino que en un fichero de variables, o en la parte superior de un fichero .scss o .less podemos agruparlas todas, y asignarles un nombre identificativo, para que desde el resto del fichero llamemos al nombre de la variable y no directamente al valor.

Por otro lado, a la cuestión planteada sobre la reutilización de código ya desarrollado: la respuesta es sí; a través de los mixin es posible agrupar tantas líneas de código como se deseen. De esta forma si queremos asignarles un mismo estilo a varios elementos, ya no será necesario replicar las mismas líneas innumerables veces, sino que basta como incluir el mixin contenedor.

Existen otros preprocesadores como por ejemplo Stylus, con un comportamiento similar a los descritos en este tema.

/ 9. Webgrafía

García-Miguel, D. (2019). Diseño de Interfaces Web (1.a ed.). Madrid, España: Síntesis.

Less. Recuperado de http://lesscss.org

Sass. Recuperado de https://sass-lang.com