INSTITUTO SUPERIOR TECNOLÓGICO DEL SUR

CARRERA

DISEÑO Y PROGRAMACIÓN WEB

LENGUAJE WEB III

"SASS VS LESS"

PROFESOR(A): Amado Cerpa Juan Andrés

ALUMNO : Vilca Apaza Christian

SEMESTRE : V

25/04/2021

Índice

LESS (lenguaje de hojas de estilo)

Less (a veces estilizado como LESS) es un dinámico lenguaje de hojas de estilo que puede ser compilado en hojas de estilo en cascada (CSS) y ejecutarse en el lado del cliente o en el lado del servidor. Diseñado por Alexis Sellier. Está influenciado por Sass y ha influido en la nueva sintaxis "SCSS" de Sass, que adaptó su sintaxis de formato de bloque similar al de CSS.2 LESS es de código abierto. Su primera versión fue escrita en Ruby, sin embargo, en las versiones posteriores, se abandonó el uso de Ruby y se lo sustituyó por JavaScript. La sintaxis indentada de Less es un metalenguaje anidado, lo que es válido en CSS es válido en SCSS con la misma semántica. LESS proporciona los siguientes
mecanismos: variables, anidamiento, operadores, mixins y funciones.
La principal diferencia entre Less y otros precompiladores CSS es que Less permite la compilación en tiempo real vía less.js por el navegador. LESS se puede ejecutar en el lado del cliente y del lado del servidor, o se puede compilar en CSS sin formato.

Variables

LESS permite que se definan las variables. Las variables de LESS se definen con una arroba (@). La asignación de variables se hace con dos puntos (:). Durante la traducción, los valores de las variables se insertan en el documento CSS de salida.

```
@color: #4D926F;
@background: #3d3d3d;

#header {
  color: @color;
  background:@background;
}
h2 {
  color: @color;
}
```

El código anterior en LESS compilaría en el siguiente código CSS:

```
#header {
  color: #4D926F;
  background: #3d3d3d;
}
h2 {
  color: #4D926F;
}
```

Mixin

Los mixins permiten incrustrar todas las propiedades de una clase dentro de otra clase al incluir el nombre de la clase como una de sus propiedades, comportándose así en una especie de constante o variable. También pueden comportarse como funciones y tomar argumentos. CSS no soporta mixins, cualquier código repetido se tiene que repetir en cada lugar. Los mixins permiten repeticiones de código más eficientes y limpias, así como modificaciones de código más fáciles y rápidas.3

```
.rounded-corners (@radius: 5px) {
  -webkit-border-radius: @radius;
  -moz-border-radius: @radius;
  border-radius: @radius;
}

#header {
  .rounded-corners;
}
#footer {
  .rounded-corners(10px);
}
```

El código anterior en LESS compilaría en el siguiente código CSS:

```
#header {
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
}
#footer {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
}
```

LESS tiene un tipo especial de conjunto de reglas llamado mixins paramétricos (parametric mixins) que se pueden mezclar como en las clases, pero aceptan parámetros.

Anidamiento

CSS admite anidamiento lógico, pero los bloques de código no están anidados. LESS permite anidar los selectores dentro de otros selectores. Esto hace la herencia clara y las hojas de estilo más cortas.

```
#header {
  h1 {
    font-size: 26px;
    font-weight: bold;
  }
  p {
    font-size: 16px;
    a {
      text-decoration: none;
      color: red;
      &:hover {
        border-width: 1px;
        color: #fff;
      }
    }
  }
}
```

El código anterior en LESS se compilará como el siguiente código CSS:

```
#header h1 {
  font-size: 26px;
  font-weight: bold;
}
#header p {
  font-size: 16px;
}
#header p a {
  text-decoration: none;
  color: red;
}
#header p a:hover {
  border-width: 1px;
  color: #fff;
}
```

Funciones y operaciones

LESS permite operaciones y funciones. Las operaciones permiten la suma, resta, división y multiplicación de valores de propiedades y colores, que se pueden utilizar para crear relaciones complejas entre propiedades. Las funciones son transformaciones uno a uno con código JavaScript, permitiendo la manipulación de valores.

```
@the-border: 1px;
@base-color: #111;
@red:        #842210;

#header {
  color: @base-color * 3;
  border-left: @the-border;
  border-right: @the-border * 3;
}
#footer {
  color: @base-color + #003300;
  border-color: desaturate(@red, 10%);
}
```

El código anterior en LESS se compilará como el siguiente código CSS:

```
#header {
  color: #333333;
  border-left: 1px;
  border-right: 3px;
}
#footer {
  color: #114411;
  border-color: #7d2717;
}
```

Sass

Cualquier preprocesador es perfectamente válido. Podríamos sin duda elegir otras alternativas como Less o Stylus y estaría estupendo para nosotros y nuestro proyecto, ya que al final todos ofrecen más o menos las mismas utilidades. Pero sin embargo, Sass se ha convertido en el preprocesador más usado y el más demandado.

Al haber recibido un mayor apoyo de la comunidad es más factible que encuentres ofertas de trabajo con Sass o que heredes proyectos que usan Sass, por lo que generalmente te será más útil aprender este preprocesador.
Además, varios frameworks usan Sass, como el caso de Bootstrap. Por ello, si tienes que trabajar con ellos te vendrá mejor aprender Sass.

Cómo es la sintaxis de Sass

Como decimos, la tarea de aprender Sass se divide en dos bloques principales. Primero aprender a trabajar con el código de Sass y conocer su sintaxis especial. La segunda tarea sería aplicar Sass en tu flujo de trabajo, de modo que puedas usar el preprocesador fácilmente y no te quite tiempo el proceso de compilación a CSS. Veremos las dos partes en este primer artículo del Manual de Sass, comenzando por examinar su sintaxis.

Obviamente, no vamos a poder explicar la sintaxis de Sass en unas pocas líneas. Para eso tenemos el manual completo, pero queremos comenzar dando algunos ejemplos para que veas cómo es.

Lo primero que debes conocer de Sass es que existen dos tipos de sintaxis para escribir su código:

Sintaxis Sass: esta sintaxis es un poco diferente de la sintaxis de CSS estándar. No difiere mucho. Por ejemplo, te evita colocar puntos y coma al final de los valores de propiedades. Además, las llaves no se usan y en su lugar se realizan indentados.
Sintaxis SCSS: Es una sintaxis bastante similar a la sintaxis del propio CSS. De hecho, el código CSS es código SCSS válido. Podríamos decir que SCSS es código CSS con algunas cosas extra.

En la práctica, aunque podría ser más rápido escribir con sintaxis Sass, es menos recomendable, porque te aleja más del propio lenguaje CSS. No solo es que te obligue a aprender más cosas, sino que tu código se parecerá menos al de CSS estándar y por lo tanto es normal que como desarrollador te sientas menos "en casa" al usar sintaxis Sass. En cambio, al usar SCSS tienes la ventaja de que tu código CSS de toda la vida será válido para el preprocesador, pudiendo reutilizar más tus conocimientos y posibles códigos de estilo con los que vengas trabajando en los proyectos.
Como entendemos que al usar un preprocesador es preferible mantenerse más cerca del lenguaje CSS, en este manual usaremos siempre sintaxis SCSS.
Ahora veamos algunas cosas sobre la sintaxis de Sass. Por ejemplo, el uso de variables.

```
$color-fondos: #F55;

h1 {
  background-color: $color-fondos;
}
```

En este código estás viendo cómo declaramos una variable y a continuación la usamos en un selector. Obviamente, eso no es un CSS válido, ya que no existen este tipo de declaraciones en el lenguaje.

Nota: Hoy CSS ya incorpora variables, aunque se usa otra sintaxis. El problema es que no están disponibles todavía en todos los navegadores, por lo que no es totalmente seguro usarlas, a no ser que implementes PostCSS con CSS Next.

Una vez compilado el código anterior, obtendrías un código estándar como el siguiente:

```
h1 {
  background-color: #F55; }
```

Cómo usar Sass

Para usar Sass tienes dos alternativas fundamentales.

1. Preprocesar con alguna herramienta de interfaz gráfica, como el caso de Prepros, CodeKit o Scout-App. En principio puede ser más sencillo, ya que no requiere trabajar con la línea de comandos, pero necesitas usar un programa en concreto y no siempre puede estar disponible para ti, o no integrarse en otro flujo de trabajo que puedas tener ya asumido en tu proyecto. Además, las mejores herramientas de interfaz gráfica tienen la desventaja de ser de pago, o necesitar una licencia para desbloquear todo su poder.
2. Usar la línea de comandos para preprocesar. Esta es la opción preferida por la mayoría de desarrolladores. No sólo porque no requiere la compra de una licencia por el software de interfaz gráfica, sino principalmente porque la puedes integrar con todo un ecosistema de herramientas para optimizar multitud de aspectos en un sitio web. Además, está al alcance de cualquier desarrollador, ya que todos tenemos un terminal en nuestro sistema operativo y finalmente, permite personalizar completamente el comportamiento del preprocesador.
3. Usar herramientas de automatización. Como tercera opción es muy común también usar herramientas que permiten automatizar el flujo de trabajo frontend, compilando archivos CSS, Javascript, optimizando imágenes, etc. Nos referimos a paquetes como Gulp, Grunt o Webpack (aunque este último es más un empaquetador). Estos sistemas tienen la particularidad que sirven para cubrir todas las necesidades de trabajo con los lenguajes de la web y no solamente se usan para compilar el código Sass. Sería la opción más potente de las tres comentadas, aunque requiere mayor formación para poder usarlas. En este grupo también podríamos unir a PostCSS, aunque esta herramienta solamente nos sirve para convertir el CSS, aplicando diversos plugins, entre los que podría estar la compilación de Sass, y no entra en áreas como el Javascript.

Las anteriores alternativas pueden marear un poco si no estás acostumbrado a oir acerca de tantas tecnologías. No te preocupes demasiado porque vamos a irnos a una opción sencilla que te permita comenzar a usar Sass, sin demasiado esfuerzo ni configuración. En concreto te vamos a explicar ahora a usar Sass desde tu terminal, con las herramientas "oficiales" del propio preprocesador (la alternativa planteada en el punto 2 anterior). Es una posibilidad al alcance de cualquier lector. Nuestro siguiente paso entonces será comenzar por instalar el preprocesador.

Instalar Sass

La instalación de Sass depende del sistema operativo con el que estás trabajando. Aunque todos requieren comenzar instalando el lenguaje Ruby, ya que el compilador de Sass está escrito en Ruby.
Veamos, para cada sistema, cómo hacernos con Ruby.

- Windows: Existe un instalador de Ruby para Windows. https://rubyinstaller.org/ Puedes usarlo para faciltar las cosas.
- Linux: tendrás que instalar Ruby usando tu gestor de paquetes de la distribución con la que trabajes, apt-get, yum, etc.
- Mac: Ruby está instalado en los sistemas Mac, por lo que no necesitarías hacer ningún paso adicional.

Una vez instalado el lenguaje Ruby tendremos el comando "gem", que instala paquetes (software) basados en este lenguaje. Así que usaremos ese comando para instalar Sass. De nuevo, puede haber diferencias entre los distintos sistemas operativos.

Windows
Tendrás que abrir un terminal, ya sea Power Shell, "cmd" o cualquiera que te guste usar. Si no usas ninguno simplemente escribe "cmd" después de pulsar el botón de inicio. Luego tendrás que lanzar el comando.

```
gem install sass
```

Linux
Para instalar Sass, una vez tienes Ruby anteriormente, podrás hacerlo con el siguiente comando en tu terminal.

```
sudo gem install sass --no-user-install
```

Variables

Sass permite la definición de variables. Las variables comienzan con el signo de dólar ($). La asignación de variables se hace con los dos puntos (:).

SassScript permite 4 tipos de datos:
Números (incluyendo las unidades)
Strings (con comillas o sin ellas)
Colores (código, o nombre)
Booleanos

Las variables pueden ser resultados o argumentos de varias funciones. disponibles. Durante el proceso de traducción, los valores de las variables son insertados en el documento CSS de salida.
En el estilo SCSS

```
$blue: #3bbfce;
$margin: 16px;

.content-navigation {
  border-color: $blue;
```

```
  color: darken($blue, 9%);
}

.border {
  padding: $margin / 2;
  margin: $margin / 2;
  border-color: $blue;
}
```

O el estilo SASS

```
$blue: #3bbfce
$margin: 16px

.content-navigation
  border-color: $blue
  color: darken($blue, 9%)

.border
  padding: $margin/2
  margin:  $margin/2
  border-color: $blue
```

Debe compilar a:

```
.content-navigation {
  border-color: #3bbfce;
  color: #2b9eab;
}

.border {
  padding: 8px;
  margin: 8px;
  border-color: #3bbfce;
}
```

Nesting (Anidamiento)

CSS soporta anidamiento lógico, pero los bloques de código no son anidados. Sass permite que el código anidado sea insertado dentro de cualquier otro bloque.

```
table.hl {
  margin: 2em 0;
  td.ln {
    text-align: right;
  }
}

li {
  font: {
    family: serif;
    weight: bold;
    size: 1.2em;
  }
```

```
}
```

Debe compilar a:

```
table.hl {
  margin: 2em 0;
}
table.hl td.ln {
  text-align: right;
}

li {
  font-family: serif;
  font-weight: bold;
  font-size: 1.2em;
}
```

Otros tipos de anidado más complejos incluyendo namespace anidamiento y referencias al padre son discutidos en la documentación de Sass.

Mixins

CSS no soporta mixins. Cualquier código duplicado debe ser repetido en cada lugar. Un mixin es una sección de código que contiene código Sass. Cada vez que se llama un mixin en el proceso de conversión el contenido del mismo es insertado en el lugar de la llamada. Los mixin permiten una solución limpia a las repeticiones de código, así como una forma fácil de alterar el mismo.

```
@mixin table-base {
  th {
    text-align: center;
    font-weight: bold;
  }
  td, th {padding: 2px}
}

#data {
  @include table-base;
}
```

Debe compilar a:

```
#data th {
  text-align: center;
  font-weight: bold;
}
#data td, #data th {
  padding: 2px;
}
```

Argumentos

Los mixins también soportan argumentos.

```
@mixin left($dist) {
  float: left;
  margin-left: $dist;
}

#data {
  @include left(10px);
}
```

Debe compilar a:

```
#data {
  float: left;
  margin-left: 10px;
}
```

En combinación

```
@mixin table-base {
  th {
    text-align: center;
    font-weight: bold;
  }
  td, th {padding: 2px}
}

@mixin left($dist) {
  float: left;
  margin-left: $dist;
}

#data {
  @include left(10px);
  @include table-base;
}
```

Debe compilar a:

```
#data {
  float: left;
  margin-left: 10px;
}
#data th {
  text-align: center;
  font-weight: bold;
}
#data td, #data th {
  padding: 2px;
}
```

Herencia de selectores

Mientras CSS3 soporta la jerarquía Document Object Model (DOM), este no permite la herencia de selectores. La herencia se logra insertando una línea dentro de un bloque de código que use la palabra clave @extend y haga referencia a otro selector. Los atributos del selector extendido serán aplicados al selector que hace la llamada.

```
.error {
  border: 1px #f00;
  background: #fdd;
}
.error.intrusion {
  font-size: 1.3em;
  font-weight: bold;
}

.badError {
  @extend .error;
  border-width: 3px;
}
```

Debe compilar a:

```
.error, .badError {
  border: 1px #f00;
  background: #fdd;
}

.error.intrusion,
.badError.intrusion {
  font-size: 1.3em;
  font-weight: bold;
}

.badError {
  border-width: 3px;
}
```

SASS VS LESS

Coincidencias entre Sass y Less

- La principal coincidencia es que ambas son opciones válidas, ya que son preprocesadores CSS que nos van a permitir automatizar todo el proceso de creación de hojas de estilos dentro del desarrollo frontend, haciendo que este proceso sea automatizable, y permitiendo reutilizar el código CSS de un proyecto a otro.
- Ambos nos proporcionan ciertas características de los lenguajes de programación, como variables, estructuras de control, etcétera.

Ventajas de Sass frente a Less

Algunas de las ventajas que presenta Sass sobre Less son:

- Es mejor para CSS3, sobre todo si la asociamos con herramientas como Compass o Bourbon, herramientas que permiten desarrollar CSS3 de manera mucho más rápida y con muchas librerías ya estandarizadas. En Less existen recopilaciones de librerías, pero no son ni tan extensas y tan útiles como pueden ser Compass o Bourbon.
- Tiene mejores estructuras de control, como las estructuras de control condicionales como if/else, bucles como for y while, o la estructura de control para recorrer mapas each. En Less, de momento, no están todos estos tipos de estructuras de control.
- De manera nativa, permite minimizar la salida de ficheros CSS, algo que Less no permite.
- Es más usado, si atendemos a los repositorios que están en GitHub, que es un estándar a la hora de medir el uso de tecnologías, e incluso hay estudios de Developers Survey que así lo indican.

Desventajas de Sass frente a Less

- Al tener más estructuras de control, Sass tiene una sintaxis más compleja, por lo que es un poco más complejo de aprender que Less.
- En cuanto a la documentación, y esto es un aspecto un poco subjetivo, la de Less está mejor que la de Sass, que tiene todo bien documentado, pero a veces es difícil encontrar la explicación.

Conclusión Sass vs Less

Si tuviéramos que decidir entre estos dos preprocesadores, mi decisión personal sería elegir Sass.
No obstante, no está de más probar ambas herramientas y decidir cada uno la que mejor le parezca.

Bibliografía

https://es.wikipedia.org/wiki/LESS_(lenguaje_de_hojas_de_estilo)
https://amatellanes.github.io/lesscss.org/
https://lesscss.org/
https://www.genbeta.com/desarrollo/less-el-lenguaje-de-hojas-de-estilo-dinamico
https://sass-lang.com/
https://es.wikipedia.org/wiki/Sass
https://desarrolloweb.com/articulos/que-es-sass-usar-sass.html
https://openwebinars.net/blog/sass-vs-less/
https://css-tricks.com/sass-vs-less/