

SASS, UN PREPROCESADOR CSS

ADOLFO SANZ DE DIEGO

SEPTIEMBRE 2016

1 EL AUTOR

1.1 ADOLFO SANZ DE DIEGO

- Empecé desarrollando aplicaciones web, hasta que di el salto a la docencia.
- Actualmente soy Asesor Técnico Docente en el servicio TIC de la D.G de Infraestructuras y Servicios de la Consejería de Educación, Juventud y Deporte de la Comunidad de Madrid.
- Además colaboro como formador especializado en tecnologías de desarrollo.

1.2 ALGUNOS PROYECTOS

- Hackathon Lovers <http://hackathonlovers.com>: un grupo creado para emprendedores y desarrolladores amantes de los hackathones.
- Password Manager Generator <http://pasmangen.github.io>: un gestor de contraseñas online.
- MarkdownSlides
<https://github.com/asanzdiego/markdownslides>: un script para crear slides a partir de ficheros MD.

1.3 ¿DONDE ENCONTRARME?

- Mi nick: asanzdiego
 - AboutMe: <http://about.me/asanzdiego>
 - GitHub: <http://github.com/asanzdiego>
 - Twitter: <http://twitter.com/asanzdiego>
 - Blog: <http://asanzdiego.blogspot.com.es>
 - LinkedIn: <http://www.linkedin.com/in/asanzdiego>
 - Google+: <http://plus.google.com/+AdolfoSanzDeDiego>

2 INTRODUCCIÓN

2.1 ¿QUÉ ES?

- Less es un pre-procesador de CSS.
- Añade características como variables, mixins, funciones, etc.

2.2 VENTAJAS

- El CSS es así más fácil de mantener, personalizable y extensible.
- Sass tiene una sintaxis parecida a CSS.
- Es el preprocesador que usa Bootstrap 4.0.

3 INSTALACIÓN

3.1 INSTALAR RUBY

- Ubuntu:

```
sudo apt-get install ruby-full
```

- Windows: <http://rubyinstaller.org/>
- Mac: incluido

3.2 INSTALAR SASS

- Después de instalar Ruby, ejecutar

```
gem install sass
```

3.3 PREPROCESAR

- Un fichero

```
sass --watch input.scss:output.css
```

- Un directorio

```
sass --watch input/dir:output/dir
```

3.4 PROBAR SIN INSTALAR

- <http://www.sassmeister.com/>

4 CARACTERÍSTICAS

4.1 VARIABLES (I)

- El siguiente código:

```
$font-stack:    Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

4.2 VARIABLES (II)

- Se compila a:

```
body {  
  font: 100% Helvetica, sans-serif;  
  color: #333;  
}
```

4.3 VALORES POR DEFECTO (I)

- El siguiente código:

```
$content: "First content";  
$content: "Second content?" !default;  
$new_content: "First time reference" !default;  
  
#main {  
  content: $content;  
  new-content: $new_content;  
}
```

4.4 VALORES POR DEFECTO (I)

- Se compila a:

```
#main {  
  content: "First content";  
  new-content: "First time reference";  
}
```


4.5 SCOPE

- Los **ámbitos de las variables** en Sass es muy similar a otros lenguajes:

```
$var: red;

#page {
  $var: white;
  #header {
    color: $var; // white
  }
}
```

4.6 REGLAS ANIDADAS (I)

- El siguiente código:

```
nav {  
  ul {  
    margin: 0;  
    padding: 0;  
    list-style: none;  
  }  
  
  li { display: inline-block; }  
}
```

4.7 REGLAS ANIDADAS (II)

- Se compila a:

```
nav ul {  
  margin: 0;  
  padding: 0;  
  list-style: none;  
}  
  
nav li {  
  display: inline-block;  
}
```

4.8 PARCIALES

- Es un archivo con un guión bajo:

```
_partial.scss
```

- Con esto Sass entiende que es un archivo parcial y que no debe generar CSS.

4.9 IMPORTS (I)

- Imaginemos el archivo `**_reset.scss**`:

```
html, body, ul {  
  margin: 0;  
  padding: 0;  
}
```


4.10 IMPORTS (II)

- Y el archivo `base.scss`:

```
@import 'reset';

body {
  font: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

4.11 IMPORTS (III)

- Se compila a:

```
html, body, ul, ol {  
  margin: 0;  
  padding: 0;  
}  
  
body {  
  font: 100% Helvetica, sans-serif;  
  background-color: #efefef;  
}
```

4.12 PARENT (I)

- El código '&' hace referencia al padre:

```
a {  
  font-weight: bold;  
  text-decoration: none;  
  &:hover { text-decoration: underline; }  
  body.firefox & { font-weight: normal; }  
}
```

4.13 PARENT (II)

- Se compila a:

```
a {  
  font-weight: bold;  
  text-decoration: none;  
}  
a:hover { text-decoration: underline; }  
body.firefox a { font-weight: normal; }
```

4.14 MIXINS (I)

- El siguiente código:

```
@mixin border-radius($radius) {  
    -webkit-border-radius: $radius;  
    -moz-border-radius: $radius;  
    -ms-border-radius: $radius;  
    border-radius: $radius;  
}  
  
.box {  
    @include border-radius(10px);  
}
```


4.15 MIXINS (II)

- Se compila a:

```
.box {  
  -webkit-border-radius: 10px;  
  -moz-border-radius: 10px;  
  -ms-border-radius: 10px;  
  border-radius: 10px;  
}
```

4.16 ARGUMENTOS VARIABLES (I)

- El siguiente código:

```
@mixin box-shadow($shadows...) {  
  -moz-box-shadow: $shadows;  
  -webkit-box-shadow: $shadows;  
  box-shadow: $shadows;  
}  
  
.shadows {  
  @include box-shadow(0px 4px 5px #666, 2px 6px 10px #999);  
}
```

4.17 ARGUMENTOS VARIABLES (II)

- Se compila a:

```
.shadows {  
  -moz-box-shadow: 0px 4px 5px #666, 2px 6px 10px #999;  
  -webkit-box-shadow: 0px 4px 5px #666, 2px 6px 10px #999;  
  box-shadow: 0px 4px 5px #666, 2px 6px 10px #999;  
}
```

4.18 PASAR CONTENIDO A MIXIN (I)

- El siguiente código:

```
@mixin apply-to-ie6-only {  
  * html {  
    @content;  
  }  
}  
@include apply-to-ie6-only {  
  #logo {  
    background-image: url(/logo.gif);  
  }  
}
```

4.19 PASAR CONTENIDO A MIXIN (II)

- Se compila a:

```
* html #logo {  
  background-image: url(/logo.gif);  
}
```


4.20 INTERPOLACIÓN (I)

- Podemos sustituir el valor de una variable en nombre de propiedades o selectores con `#{$variable}`:

```
$position: left;
@mixin car($brand, $color) {
  .coche.#{ $brand } {
    border-#{ $position }: 2px;
    background-color: $color;
    background-image: url('img/#{ $brand }-#{ $color }.png');
  }
}
@include car('audi', 'green');
```

4.21 INTERPOLACIÓN (II)

- Se compila a:

```
.coche.audi {  
  border-left: 2px;  
  background-color: green;  
  background-image: url('img/audi-green.png');  
}
```

4.22 EXTEND (I)

- El siguiente código:

```
.message {  
  color: #333;  
}  
.success {  
  @extend .message;  
  border-color: green;  
}  
.error {  
  @extend .message;  
  border-color: red;  
}
```

4.23 EXTEND (II)

- Se compila a:

```
.message, .success, .error, .warning {  
  color: #333;  
}  
.success {  
  border-color: green;  
}  
.error {  
  border-color: red;  
}
```

4.24 OPERADORES (I)

- El siguiente código:

```
article[role="main"] {  
  float: left;  
  width: 600px / 960px * 100%;  
}  
  
aside[role="complementary"] {  
  float: right;  
  width: 300px / 960px * 100%;  
}
```


4.25 OPERADORES (II)

- Se compila a:

```
article[role="main"] {  
  float: left;  
  width: 62.5%;  
}  
  
aside[role="complementary"] {  
  float: right;  
  width: 31.25%;  
}
```

4.26 FUNCIONES (I)

- Sass dispone de una variedad de funciones matemáticas, que manipulan cadenas, y que transforman los colores:

```
@base: #f04615;  
@list: 200, 500, 1200;  
  
.class {  
  width: extract(@list, 3);  
  color: saturate(@base, 5%);  
  background-color:  
    lighten(@base, 25%);  
}
```

4.27 FUNCIONES (II)

- Se compila a:

```
.class {  
  width: 1200;  
  color: #f6430f;  
  background-color: #f8a58d;  
}
```

4.28 DEFINIR FUNCIONES (I)

- Podemos definir nuestras propias funciones:

```
$grid-width: 40px;  
$gutter-width: 10px;  
  
@function grid-width($n) {  
  @return $n * $grid-width + ($n - 1) * $gutter-width;  
}  
  
#sidebar { width: grid-width(5); }
```

4.29 DEFINIR FUNCIONES (II)

- Se compila a:

```
#sidebar {  
  width: 240px;  
}
```


4.30 IF (I)

- El siguiente código:

```
$animal: cat;  
p {  
  @if 1+1 == 2 { border: 2px solid black; }  
  @if $animal == cat { color: white; }  
}
```

4.31 IF (I)

- Se compila a:

```
p {  
  border: 2px solid black;  
  color: white;  
}
```

4.32 FOR (I)

- El siguiente código:

```
@for $i from 1 to 4 {  
  .item-#{$i} { width: 2em * i; }  
}
```

4.33 FOR (II)

- Se compila a:

```
.item-1 { width: 2em; }  
.item-2 { width: 4em; }  
.item-3 { width: 6em; }
```

4.34 EACH (I)

- El siguiente código:

```
@each $animal in puma, tiger, salamander {  
  .#{$animal}-icon {  
    background-image: url('/images/#{$animal}.png');  
  }  
}
```


4.35 EACH (II)

- Se compila a:

```
.puma-icon {  
  background-image: url('/images/puma.png'); }  
.tiger-icon {  
  background-image: url('/images/tiger.png'); }  
.salamander-icon {  
  background-image: url('/images/salamander.png'); }
```

4.36 WHILE (I)

- El siguiente código:

```
$i: 6;  
@while $i > 0 {  
  .item-#{$i} { width: 2em * $i; }  
  $i: $i - 2;  
}
```

4.37 WHILE (II)

- Se compila a:

```
.item-6 { width: 12em; }  
.item-4 { width: 8em; }  
.item-2 { width: 4em; }
```

4.38 LOGS

- El siguiente código:

```
@debug 10em + 12em;
```

- Saca por pantalla a la hora de compilar:

```
Line 1 DEBUG: 22em
```

5 ACERCA DE

5.1 LICENCIA

- Estas transparencias están hechas con:
 - MarkdownSlides:
<https://github.com/asanzdiego/markdownslides>
- Estas transparencias están bajo una licencia Creative Commons Reconocimiento-CompartirIgual 3.0:
 - <http://creativecommons.org/licenses/by-sa/3.0/es>

5.2 FUENTES

- Transparencias:
 - <https://github.com/asanzdiego/curso-interfaces-web-2016/tree/master/04-preprocesadores-css/04.02-sass/slides>
- Código:
 - <https://github.com/asanzdiego/curso-interfaces-web-2016/tree/master/04-preprocesadores-css/04.02-sass/src>

5.3 BIBLIOGRAFÍA

- Documentación oficial de Sass
 - <http://sass-lang.com/>
- Para probar Sass
 - <http://www.sassmeister.com/>