Introducción

Introducción

SASS es un preprocesador y extensión del lenguaje CSS

Añade características dinámicas a CSS como anidamiento, variables y funciones

Ventajas:

- Desarrollo de CSS más rápido y conciso
- Mejor mantenimiento del código

Inconvenientes

- Hay que procesar SASS para convertirlo a CSS

Nuestro primer SASS

hello.scss

```
$main-text-color: blue;

body {
    color: $main-text-color;
}
```

hello.scss

```
$ node-sass hello.scss

body {
  color: blue;
}
```

Sintaxis

¿SASS o SCSS?

SASS vs SCSS

SASS Syntax

```
nav
  ul
    margin: 0
    padding: 0
    list-style: none
```

SCSS Syntax

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
}
```

SASS vs SCSS

Ambas pueden convertirse fácilmente a CSS.

SCSS es en realidad una versión de SASS (Sassy CSS) y es la más utilizada debido a:

- Es una extensión de CSS y su sintaxis es 100% compatible con él. Podemos cambiar la extensión de un archivo .css a .scss y funcionará. No pasa lo mismo con SASS.

- Menor barrera de entrada conociendo la sintaxis de CSS

Variables

Podemos definir variables para almacenar valores o incluso para utilizar como nombres de propiedades CSS o rutas a ficheros.

Variables

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

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

Resultado

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

Anidamiento

Podemos anidar las reglas unas dentro otras, lo que nos ahorrará mucho código repetitivo

Anidamiento

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
}
```

Resultado

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
```

Especialización de un elemento

```
a {
  font-weight: bold;
  text-decoration: none;

  &.redLink { color: red }

  &:hover { text-decoration: underline; }

  body.firefox & { font-weight: normal; }
}
```

Resultado

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

Comentarios

Comentarios

```
/* comentarios clásicos
o multilínea */

// comentario de una sola línea
```

Mixins

Los mixins nos permiten mezclar propiedades de clases y también definir funciones de generación de código CSS.

Mixin sin parámetros

```
@mixin clearfix {
  display: inline-block;
  &:after {…}
  * html & { height: 1px }
}

.menu { @include clearfix; }
```

Debemos utilizar la directiva ***@include*** para utilizar un ***mixin***.

Resultado

```
.menu { display: inline-block; }
.menu:after {…}
* html .menu { height: 1px }
```

Mixin con parámetro

```
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}

.box { @include border-radius(10px); }
```

Resultado

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

Mixin como bloque

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

Resultado

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

Útil para responsive

```
$desktop-width: 1024px;
$tablet-width: 768px;

@mixin tablet {
  @media (min-width: #{$tablet-width}) and (max-width: #{$desktop-width - 1px}) {
    @content;
  }
}
```

Útil para responsive

```
p {
  font-size: 16px;
  @include tablet {
    font-size: 18px;
  }
}
```

Resultado

```
p {
  font-size: 16px;
}
@media (min-width: 768px) and (max-width: 1023px) {
    p {
        font-size: 18px;
    }
}
```

Operadores

Al poder definir variables podemos realizar operaciones básicas con las mismas

Operadores

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

Operadores

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

Funciones built-in

SASS incorpora una serie de funciones built-in para realizar diferentes operaciones

opacify & transparentize

```
$translucent-red: rgba(255, 0, 0, 0.5);

p {
  color: opacify($translucent-red, 0.3);
  background: transparentize($translucent-red, 0.25);
}
```

Añaden o quitan transparencia a un color

opacify & transparentize

```
p {
  color: rgba(255, 0, 0, 0.8);
  background-color: rgba(255, 0, 0, 0.25);
}
```

Funciones built-in

http://sass-lang.com/documentation/Sass/Script/Functions.html

@import

La directiva @import se comporta de diferentes maneras en función de la extensión del archivo que importemos.

Se mantiene intacta cuando…

- Si la extensión del archivo es .css
- Si el archivo empieza por “http”
- Si el archivo es una url()
- Si la instrucción tiene media queries

@import

```
@import "foo.css";
@import "foo" screen;
@import "http://foo.com/bar";
@import url(foo);
```

Resultado del @import

```
@import "foo.css";
@import "foo" screen;
@import "http://foo.com/bar";
@import url(foo);
```

@import

```
// _example.scss
.red-color {
  color: red;
}

// base.scss
#main {
  @import "example";
}
```

Resultado de @import anidado

```
#main .example {
  color: red;
}
```

Partials

Archivos SASS que queremos cargar pero no compilar
(no queremos crear su respectivo archivo CSS)

Partials

_reset.scss

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

base.scss

```
@import 'reset';

body {
  font: 100% Helvetica;
  background-color: white;
}
```

Partials

El nombre de archivo un ***partial*** deben comenzar por _ seguido del nombre del partial (y su extensión).

Para usarlo, debemos utilizar la instrucción ***@import*** seguido del nombre del partial.

Nombre	Archivo	Importación
reset	_reset.scss	@import “reset”

@media

Funciona igual que en CSS…pero pueden anidarse!

Especializando el comportamiento de una clase

```
.sidebar {
  width: 300px;
  @media screen and (orientation: landscape) {
    width: 500px;
  }
}
```

Resultado

```
.sidebar {
  width: 300px;
}
@media screen and (orientation: landscape) {
    .sidebar {
      width: 500px;
    }
}
```

Anidándose unas con otras

```
@media screen {
  .sidebar {
    @media (orientation: landscape) {
      width: 500px;
    }
  }
}
```

Resultado

```
@media screen and (orientation: landscape) {
  .sidebar {
    width: 500px;
  }
}
```

@extend

Permite heredar las reglas de un estilo

Herencia

```
.message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  @extend .message;
  border-color: green;
}
```

Resultado

```
.message, .success {
  border: 1px solid #cccccc;
  padding: 10px;
  color: #333;
}

.success {
  border-color: green;
}
```

@if

Permite escribir reglas condicionales

@if

```
p {
  @if 1 + 1 == 2 { border: 1px solid;  }
  @if 5 < 3      { border: 2px dotted; }
  @if null       { border: 3px double; }
}
```

Resultado

```
p {
  border: 1px solid;
}
```

@for

Permite hacer un bucle para escribir reglas

@for

```
@for $i from 1 through 3 {
  .item-#{$i} { width: 2em * $i; }
}
```

Resultado

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

@each

Permite recorrer una lista de valores

@each

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

Resultado

```
.puma-icon {
  background: url('/images/puma.png');
}
.sea-slug-icon {
  background: url('/images/sea-slug.png');
}
.egret-icon {
  background: url(‘/images/bird.png');
}
```

@while

Permite realizar un bucle mientras se cumple una condición

@while

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

Resultado

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

@function

Permite definir nuestras propias funciones

@function

```
$grid-width: 40px;
$gutter-width: 10px;

@function grid-width($n) {
  @return $n * $grid-width + ($n - 1) * $gutter-width;
}

#sidebar { width: grid-width(5); }
```

Resultado

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

GRACIAS

www.keepcoding.io