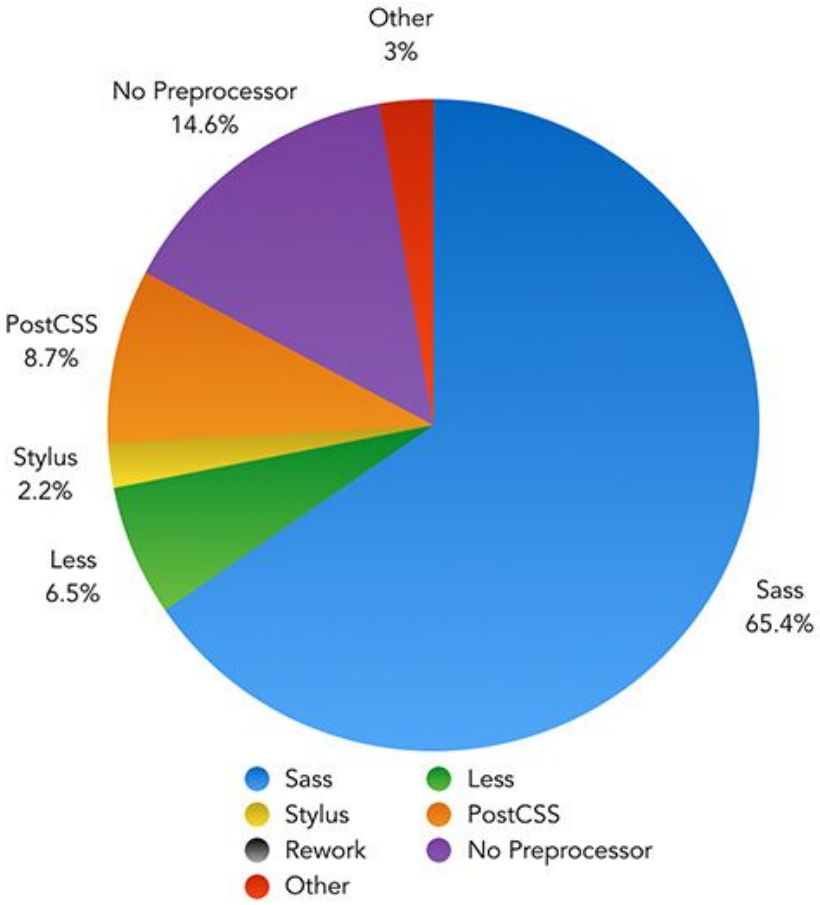
UT03.1 Preprocesadores CSS

01 de Septiembre de 2019

Preprocesadores CSS

- Existen varios procesadores en el mercado: Sass, PostCSS, LESS y Stylus
- Escribir CSS puede convertirse en una tarea repetitiva en la que tareas como cambiar los colores, cerrar etiquetas etc consumen mucho tiempo
- Un preprocesador CSS es básicamente un lenguaje de pseudo-código que extiende CSS y que se compila en un CSS normal
- Este pseudo-código se compone de variables, condiciones, bucles o funciones
- El objetivo es tener un código CSS más sencillo de mantener y editar

Preprocesadores CSS



Fuente: Encuesta de 2018: https://ashleynolan.co.uk/blog/frontend-tooling-survey-2018-results

Preprocesadores CSS

Ventajas

- Código más limpio y reutilizable con variables
- Te ahorra tiempo
- Fácil de mantener el código con fragmentos de código (snippsets) y librerías
- Permite añadir cálculos y lógica
- Más organizado y fácil de configurar

Preprocesadores CSS

Sass vs LESS

- Ambos son extensiones de CSS muy potentes
- Puedes pensar en los dos como un lenguaje de programación diseñado para hacer CSS más fácil de mantener, hacer plantillas y extenderlo
- Ambos son compatibles hacia atrás, por lo que puedes convertir ficheros CSS renombrandolos con la extensión .less o scss
- LESS está basado en JavaScript y Sass en Ruby

PostCSS vs Sass y Less

- Se puede usar junto con herramientas de automatización de tareas (gulp, grunt…)
- Hay una gran colección de plugins con los que puedes hacer cosas que no puedes con los pre-procesadores actuales
- Puedes crear tus propios plugins fácimente en javascript, si así lo necesitas.
- Puedes usarlo con archivos css «normales», pero también usando el plugin que toque con archivos de otros pre-procesadores
- Compatible con CSSNext para hacer compatible las últimas novedades de CSS3

Less

- Less es un preprocesador CSS
- Esto significa que extiende el lenguaje CSS añadiendo características que permiten variables, mixins, funciones y otras técnicas que permiten hacer CSS
  - Más mantenible
  - Más reutilizable
  - Más extendible
- Less se ejecuta dentro de Node, en el navegador y dentro de Rhino
- Existe también herramientas de terceros que permiten compilar los ficheros y ver los cambios
- La forma más rápida de probarlo es con un editor online

Less

Instalación

- Primero debemos instalar node.js en nuestro equipo
- Una vez instalado abriremos la consola de comandos node.js y ejecutaremos lo siguiente

```
$ npm install -g less
```

- En windows podemos instalar winless para que compile automáticamente cada vez que hagamos un cambio en el fichero .less

Less

Uso de la línea de comandos

- Una vez instalado puedes compilar un fichero con el siguiente comando

```
lessc styles.less
```

- Esta genera una salida de un fichero compilado CSS en la consola, para guardarlo en un fichero se ejecuta el siguiente comando

```
lessc styles.less > styles.css
```

Less

Variables

- Una de las carácteristicas de Less es la habilidad para crear variables como en los lenguajes de programación
- Puedes guardar cualquier tipo de valor que uses con frecuencia: colores, dimensiones, selectores, fuentes, URLS etc.
- La filosofía de less es reutilizar la sintaxis CSS todo lo que se pueda
- Las variables se definen utilizando una arroba @ delante del nombre y se les asigna el valor con dos puntos
- Más información http://lesscss.org/features/#variables-feature

Less

Ejemplo de Variables

```
@background-color: #ffffff;
@text-color: #1A237E;

p{
  background-color: @background-color;
  color: @text-color;
  padding: 15px;
}

ul{
  background-color: @background-color;
}

li{
  color: @text-color;
}
```

Less

Mixins

- Los mixins permiten incluir todas las propiedades de una clase en otra diferente añadiendo simplemente el nombre de la clase como una propiedad más
- El comportamiento es similar al de las variables pero con clases enteras
- Los mixins pueden comportarse también como funciones, y tomar argumentos
- Más información http://lesscss.org/features/#mixins-feature

Less

Ejemplo Mixins

less

```
.circle{
  background-color: #4CAF50;
  border-radius: 100%;
}
.small-circle{
  width: 50px;
  height: 50px;
  .circle
}
.big-circle{
  width: 100px;
  height: 100px;
  .circle
}
```

CSS

```
.circle {
  background-color: #4CAF50;
  border-radius: 100%;
}
.small-circle {
  width: 50px;
  height: 50px;
  background-color: #4CAF50;
  border-radius: 100%;
}
.big-circle {
  width: 100px;
  height: 100px;
  background-color: #4CAF50;
  border-radius: 100%;
}
```

Less

Ejemplo de ocultación de Mixin con paréntesis

less

```
.circle(){
  background-color: #4CAF50;
  border-radius: 100%;
}
.small-circle{
  width: 50px;
  height: 50px;
  .circle
}
.big-circle{
  width: 100px;
  height: 100px;
  .circle
}
```

CSS

```
.small-circle {
  width: 50px;
  height: 50px;
  background-color: #4CAF50;
  border-radius: 100%;
}
.big-circle {
  width: 100px;
  height: 100px;
  background-color: #4CAF50;
  border-radius: 100%;
}
```

Less

Ejemplo de mixin con parámetros

less

```
.circle(@size: 25px){
  background-color: #4CAF50;
  border-radius: 100%;

  width: @size;
  height: @size;
}

.small-circle{
  .circle
}

.big-circle{
  .circle(100px)
}
```

css

```
.small-circle {
  background-color: #4CAF50;
  border-radius: 100%;
  width: 25px;
  height: 25px;
}
.big-circle {
  background-color: #4CAF50;
  border-radius: 100%;
  width: 100px;
  height: 100px;
}
```

Less

Anidado

- En vez de escribir selectores con nombres largos para especificar una relación de herencia, en Less puedes anidar selectores dentro de otros como en HTML
- Ésto hace la herencia más clara y reducir el tamaño de la hoja de estilo

Less

Ejemplo de anidado

less

```
ul{
  background-color: #03A9F4;
  padding: 10px;
  list-style: none;

  li{
    background-color: #fff;
    border-radius: 3px;
    margin: 10px 0;
  }
}
```

CSS

```
ul {
  background-color: #03A9F4;
  padding: 10px;
  list-style: none;
}
ul li {
  background-color: #fff;
  border-radius: 3px;
  margin: 10px 0;
}
```

Less

Ámbito de variables

- Al igual que en los lenguajes de programación, las variables en Less reciben sus valores dependiendo de su ámbito
- Si no tiene especificado un valor en un ámbito, LESS buscará en bloque superiores hasta encontrar la declaración más cercana
- Más información http://lesscss.org/features/#features-overview-feature-scope

Less

Ejemplo de ámbito de variables

less

```
@text-color: #000000;

ul{
  @text-color: #ffffff;
  background-color: #03A9F4;
  padding: 10px;
  list-style: none;

  li{
    color: @text-color;
    border-radius: 3px;
    margin: 10px 0;
  }
}
```

CSS

```
ul {
  background-color: #03A9F4;
  padding: 10px;
  list-style: none;
}
ul li {
  color: #ffffff;
  border-radius: 3px;
  margin: 10px 0;
}
```

Less

Operaciones

- Las operaciones permiten sumar, restar, multiplicar y dividir los valores de las propiedades y colores
- De esta forma se pueden crear relaciones complejas entre propiedades
- Las operaciones deberían estar definidas entre paréntesis para asegurar la compatibilidad con CSS.

Less

Operaciones

less

```
@div-width: 100px;
@color: #03A9F4;

div{
  height: 50px;
  display: inline-block;
}

#left{
  width: @div-width;
  background-color: (@color - 100);
}

#right{
  width: (@div-width * 2);
  background-color: @color;
}
```

css

```
div {
  height: 50px;
  display: inline-block;
}

#left {
  width: 100px;
  background-color: #004590;
}

#right {
  width: 200px;
  background-color: #03a9f4;
}
```

Less

Funciones

- Less permite utilizar funciones que corresponden a funciones JavaScript para modificar valores como quieras.
- Lista de funciones disponibles:http://lesscss.org/functions/

Less

Funciones

less

```
@var: #004590;

div{
  height: 50px;
  width: 50px;
  background-color: @var;

  &:hover{
    background-color:
        fadeout(@var, 50%)
  }
}
```

css

```
div {
  height: 50px;
  width: 50px;
  background-color: #004590;
}
div:hover {
  background-color:
    rgba(0, 69, 144, 0.5);
}
```

Sass

- SASS se desarrolla en 2007
- Es el pre-procesador mas utilizado
- SASS sobresale al utilizarse con frameworks como Compass y Bourbon ofreciendo más funcionalidad
- SASS permite el uso de dos sintaxis diferentes para crear sus archivos
  - **SCSS**
  - **Sintaxis Sass**

Sass

Sintaxis SCSS

- SCSS (del inglés, Sassy CSS)
- Es capaz de entender la mayoría de hacks de CSS y la sintaxis específica de los navegadores
- Los archivos creados con esta sintaxis utilizan la extensión .scss.

Sass

Ejemplo sintaxis SCSS

```
$primary-color: hotpink; //Variable
@mixin border-radius($radius) { // Mixin
    -webkit-border-radius: $radius;
    -moz-border-radius: $radius;
    border-radius: $radius;
}
.my-element {
    color: $primary-color;
    width: 100%;
    overflow: hidden;
}
.my-other-element {
    @include border-radius(5px);
}
```

Sass

Sintaxis Sass

- También conocida como sintaxis indentada
- Permite escribir los estilos CSS de manera más concisa.
- El anidamiento de selectores se indica con tabulaciones en vez de con llaves
- Las propiedades se separan con saltos de línea en vez de con puntos y coma
- Los archivos creados con esta segunda sintaxis utilizan la extensión .sass

Sass

Ejemplo sintaxis Sass

```
$primary-color: hotpink
=border-radius($radius)
    -webkit-border-radius: $radius
    -moz-border-radius: $radius
    border-radius: $radius

.my-element
    color: $primary-color
    width: 100%
    overflow: hidden

.my-other-element
    +border-radius(5px)
```

Sass

Ejemplo sintaxis Sass

- Algunos diseñadores consideran que Sass es más sencilla de leer y más rápida de escribir que SCSS
- Las dos sintaxis tienen exactamente las mismas funcionalidades .
- Una de las ventajas de SASS es que los archivos creados con una sintaxis pueden importar cualquier archivo creado con la otra sintaxis
- Existe una utilidad para la línea de comandos para convertir de una sintaxis a otra.

```
$ sass-convert estilos.sass estilos.scss
$ sass-convert estilos.scss estilos.sass
```

Sass

Características

- 100% compatible con CSS3.
- Permite el uso de variables, anidamiento de estilos y mixins.
- Incluye numerosas funciones para manipular con facilidad colores y otros valores.
- Permite el uso de elementos básicos de programación como las directivas de control y las librerías.
- Genera archivos CSS bien formateados y permite configurar su formato.

Sass

Instalación

- Instalar la última versión de Node LTS si no lo tuvieras instalado https://nodejs.org/es/
- Abrir la consola cmd y ejecutar
  - npm install -g sass

- También puedes utilizar un programa de terceros o un plugin para facilitar:
  - https://scout-app.io/
  - http://koala-app.com/
- Más información en http://sass-lang.com/install

Sass

Uso de la línea de comandos

- Una vez instalado puedes compilar un fichero con el siguiente comando

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

- Puedes añadir la opción **--watch** para decirle a Sass que vuelva a generar el archivo CSS cada vez que se cambie la hoja de estilos original

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

Sass

Sin instalación

- Existen un gran número de aplicaciones que ponen a SASS en marcha y funcionando en pocos minuto
  - CodeKit (Pago)
  - Compass.app (Pago, Open Source)
  - Hammer (Para MacOS de Pago)
  - Koala (Open Source, también la Less)
  - LiveReload (Pago, Open Source)
  - Prepros (Pago)
  - Scout (Open Source)
- Podemos probarlo sin instalar directamente con un editor online

Sass

Variables

- De la misma forma que con Less, Sass permite utilizar variables
- Permite guardar colores, dimensiones, selectores, fuentes, URLS etc.
- La filosofía de less es reutilizar la sintaxis CSS todo lo que se pueda
- Las variables se definen utilizando una arroba $ delante del nombre y se les asigna el valor con dos puntos
- Más información http://sass-lang.com/documentation/file.SASS_REFERENCE.html#Variables_____variables_
-

Sass

Variables

scss

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

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

Sass

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

body
  font: 100% $font-stack
  color: $primary-color
```

css

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

- Algunas cosas en CSS son algo tediosas para escribir, especialmente en CSS3 con los muchos prefijos para los fabricantes que existen.
- Un mixin permite agrupar declaraciones CSS que quieres reutilizar en tu sitio e incluso pasarle valores
- Para crear un mixin en SCSS se utiliza la directiva **@mixin** y se le da un nombre y después se le llama con la directiva **@include** y el nombre del mixin
- En Sass se utiliza el igual **=** y se le da un nombre y después se le llama con el signo **+** y el nombre del mixin

Sass

Ejemplo Mixins

SCSS

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

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

Sass

```
=border-radius($radius)
  -webkit-border-radius: $radius
  -moz-border-radius:    $radius
  -ms-border-radius:     $radius
  border-radius:         $radius

.box
  +border-radius(10px)
```

CSS

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

Sass

Anidado

- En vez de escribir selectores con nombres largos para especificar una relación de herencia, en Sass puedes anidar selectores dentro de otros como en HTML
- Ésto hace la herencia más clara y reducir el tamaño de la hoja de estilo
- Más información http://sass-lang.com/documentation/file.SASS_REFERENCE.html#Nested_Rules

Sass

Ejemplo de anidado

SCSS

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

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

Sass

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

  li
    display: inline-block

  a
    display: block
    padding: 6px 12px
    text-decoration: none
```

Sass

Ámbito de variables

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

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

- Más información http://sass-lang.com/documentation/file.SASS_REFERENCE.html#Variable_Scope_and_Content_Blocks

Sass

Herencia con extend

- Una de las características más potentes de Sass es la herencia
- Utilizando @extend puedes compartir un conjunto de propiedades CSS de un selector en otro

Sass

Herencia con extend

SCSS

```
.message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}
.success {
  @extend .message;
  border-color: green;
}
.error {
  @extend .message;
  border-color: red;
}
.warning {
  @extend .message;
  border-color: yellow;
}
```

Sass

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

.success
  @extend .message
  border-color: green

.error
  @extend .message
  border-color: red

.warning
  @extend .message
  border-color: yellow
```

Sass

Operadores

- Las operaciones permiten sumar, restar, multiplicar y dividir los valores de las propiedades y colores
- De esta forma se pueden crear relaciones complejas entre propiedades
- Las operaciones deberían estar definidas entre paréntesis para asegurar la compatibilidad con CSS.

Sass

Operadores

SCSS

```
.container { width: 100%; }


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

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

Sass

```
.container
  width: 100%

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


aside[role="complementary"]
  float: right
  width: 300px / 960px * 100%
```

Sass

CSS parciales

- CSS tiene la opción de importar para poder dividir tu CSS en porciones más pequeñas y más fáciles de mantener
- El problema es que cada vez que usas @import en CSS hacer una nueva petición CSS
- Sass en vez de hacer una petición HTTP request, cogerá el fichero que quieras importar y lo combinará CSS para que el navegador le llegue un solo fichero CSS
- Las porciones empieza el nombre con guión bajo para que sepa Sass que es un archivo parcial y que no tiene que generaru un fichero CSS

Sass

Ejemplo importación

_reset.scss

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

_reset.sass

```
html,
body,
ul,
ol
  margin:  0
  padding: 0
```

base.scss

```
@import 'reset';

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

base.sass

```
@import reset

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

Sass

Referencias al elemento padre

- El código ampersand '&' hace referencia al padre

scss

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

css

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

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

Sass

Funciones

- Sass dispone de una variedad de funciones matemáticas, para manipular cadenas, transformar colores etc.
- En la siguiente página se pueden consultar todas http://sass-lang.com/documentation/Sass/Script/Functions.html

scss

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

css

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

- En Sass también podemos definir nuestras propias funciones

scss

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

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

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

css

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

Sass

Directiva de control @if

- La directiva @if coge una expresión de Sass y procesa su bloque si la expresión devuelve cualquier osa que no sea falso

Sass

```
$boolean: true !default

=simple-mixin
  @if $boolean
    @debug "$boolean is #{$boolean}"
    display: block
  @else
    @debug "$boolean is #{$boolean}"
    display: none

.some-selector
  +simple-mixin
```

CSS

```
.some-selector {
  display: block;
}
```

Sass

Directiva de control @for

- La directiva @for tiene dos formas
  - **@for $var from <comienzo> through <fin>** En la que empieza en <comienzo> e itera **hasta** llegar a su fin loops <fin>
  - **@for $var from <comienzo> to <fin>** En la que empieza en <comienzo> e itera hasta llegar a su fin, **sin ejecutar la última**
- En los dos casos $var puede ser cualquier nombre de variable

Sass

Ejemplo directiva de control @for

scss

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

scss

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

css

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

css

```
.item-1 { width: 2em; }
.item-2 { width: 4em; }
.item-3 { width: 6em; }
.item-4 { width: 8em; }
```

Sass

Directiva de control @each

- La directiva @each tiene la forma **@each $var in <list>**

Sass

```
$list: adam john wynn mason

=author-images
  @each $author in $list
    .photo-#{$author}
      background:
image-url("avatars/#{$author}.png")
no-repeat

.author-bio
  +author-images
```

CSS

```
.author-bio .photo-adam {
  background: url('/images/avatars/adam.png')
no-repeat;
}
.author-bio .photo-john {
  background: url('/images/avatars/john.png')
no-repeat;
}
.author-bio .photo-wynn {
  background: url('/images/avatars/wynn.png')
no-repeat;
}
.author-bio .photo-mason {
  background:
url('/images/avatars/mason.png') no-repeat;
}
```

Directiva de control @while

- La directiva @while coge una expresión Sass y repite el bloque de estilos que contiene hasta que se evalúa como falsa
- Muy parecida a la directiva @for, permite crear sentencias de bucles muy complejas mientras se evalúa una condición como verdadera

Sass

Ejemplo de directiva de control @while

sass

```
$types: 4
$type-width: 20px

@while $types > 0
  .while-#{$types}
    width: $type-width + $types
  $types: $types - 1
```

css

```
.while-4 {
  width: 24px;
}

.while-3 {
  width: 23px;
}

.while-2 {
  width: 22px;
}

.while-1 {
  width: 21px;
}
```

Recursos

- CSSNext: https://cssnext.github.io/
- Documentación Less http://lesscss.org/features/
- Sass y preprocesadores CSS https://www.youtube.com/watch?v=X0nV53wPcSQ
- Documentación Sass: http://sass-lang.com/
- http://getbem.com/introduction/
- Curso Sass: https://learn.freecodecamp.org/front-end-libraries/sass