



## LESS, SASS

© 2017, ACTIBYTI PROJECT SLU, Barcelona  
Autor: Ricardo Ahumada



MINISTERIO  
DE ENERGÍA, TURISMO  
Y AGENDA DIGITAL

red.es



ESTRATEGIA DE  
EMPRENDIMIENTO Y  
EMPLEO JUVENIL  
*garantía juvenil*



UNIÓN EUROPEA

Fondo Social Europeo  
“El FSE invierte en tu futuro”

# ÍNDICE DE CONTENIDOS

1. LESS
2. SASS

1

LESS

# LESS

- CSS puede ser frustrante, para escribir el código se necesita un gran esfuerzo y disciplina para mantener CSS.
- LESS es una pequeña y gran herramienta que extiende las posibilidades de CSS añadiendo variables, mezclas, operaciones y reglas anidadas.
- Con LESS podemos escribir un código ligero y optimo de manera rápida. Luego podremos generar el CSS que se incorporará en la página.
- LESS es código abierto. Su primera versión fue basada en Ruby. Posteriormente fue reemplazado por JavaScript.



➤ <http://lesscss.org>

# Características LESS

- Código mas limpio, por lo que es mas legible y se puede escribir de manera organizada.
- Se pueden establecer perfiles y reutilizarlos en todo el código.
- Está basado en JavaScript y es un súper conjunto de CSS.
- Soluciona el problema de la redundancia del código.
  
- Se puede consultar la documentación en:
  - <http://lesscss.org/features/>



# Instalando LESS

# Instalar Node.js



- LESS esta escrito en JavaScript, por lo que es necesario node.js o algún navegador web para que funcione.
- Se puede incluir less.js en un sitio web y compilarse todas las hojas de estilo sin vínculo en tiempo real, pero no es recomendable, ya que es un proceso lento.
- Para instala node.js
  - Accede al sition <https://nodejs.org/en/download/>
  - Descarga la versión LTS e instálala
  - Una vez instalada, abre una ventana de consola y escribe

node -v

- Aparecerá la versión de node.js

```
C:\>node -v  
v6.9.2  
C:\>.
```

# Instalación



- Para instalar LESS, abriremos un terminal (ventana de consola) e instalaremos usando NPM con el siguiente comando:

```
npm install -g less
```

- El sistema mostrará los paquetes instalados

```
C:\>npm install -g less
C:\Users\Ricardo\AppData\Roaming\npm\lessc -> C:\Users\Ricardo\AppData\Roaming\npm\node_modules\less\bin\lessc
C:\Users\Ricardo\AppData\Roaming\npm
`-- less@2.7.2
    +- errno@0.1.4
    | `-- prr@0.0.0
    +- graceful-fs@4.1.11
    +- image-size@0.5.5
    +- mime@1.3.6
    +- mkdirp@0.5.1
    | `-- minimist@0.0.8
        +-- osenv@0.1.1
```

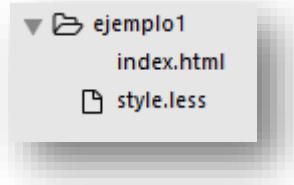


# Creando CSS con LESS



# Crea un proyecto

- Crea una carpeta para tu proyecto: less > ejemplo1
  - Añade un index.html y un archivo style.less



# Crea el HTML

```
<!DOCTYPE html>
<head>
  <link rel = "stylesheet" href = "style.css" type = "text/css" />
</head>

<body>
  <h1>Welcome to LESS</h1>
  <h3>Hello!!!!</h3>
</body>
</html>
```



# Crea las fuentes LESS

- Ahora crearemos un archivo “style.less”

```
@primarycolor: #FF7F50;  
@color:#800080;  
h1 {  
    color: @primarycolor;  
}  
  
h3 {  
    color: @color;  
}
```

# Compila LESS a CSS



- La compilación de “style.less” a “style.css”, se logra con el siguiente comando:

```
lessc style.less style.css
```

- AL ejecutar el comando en la terminal, se creara el archivo “style.css” automáticamente.

```
h1 {  
    color: #FF7F50;  
}  
  
h3 {  
    color: #800080;  
}
```

- Siempre que se modifique el archivo less, será necesario ejecutar el comando anterior para que se actualice el archivo “style.css”.

# Comprueba el resultado



- Para ver el resultado abriremos index.html con un navegador



# Reglas Anidadas

- En LESS, se puede declarar mixin del mismo modo que en CSS usando la clase o el selecto de ID.
  - <http://lesscss.org/features/#features-overview-feature-nested-rules>
- Es capaz de almacenar múltiples valores y pueden ser reutilizados en el código cuando sea necesario.
- En CSS se admite el anidamiento lógico, pero los bloques de códigos no están anidados.
- En LESS se permite anidar los selectores dentro de otros selectores , esto hace la herencia clara y las hojas de estilo mas cortas.



# Usando Reglas anidadas

# Crea el HTML



```
<html>
  <head>
    <title>Nested Rules</title>
    <link rel = "stylesheet" type = "text/css" href = "style.css" />
  </head>

  <body>
    <div class = "container">
      <h1>First Heading</h1>
      <p>LESS is a dynamic style sheet language that extends the capability of CSS.</p>
      <div class = "myclass">
        <h1>Second Heading</h1>
        <p>LESS enables customizable, manageable and reusable style sheet for web site.</p>
      </div>
    </div>
  </body>
</html>
```

# Crea el LESS

- Creamos el archivo “style.less”:

```
.container {  
    h1 {  
        font-size: 25px;  
        color:#E45456;  
    }  
    p {  
        font-size: 25px;  
        color:#3C7949;  
    }  
  
.myclass {  
    h1 {  
        font-size: 25px;  
        color:#E45456;  
    }  
    p {  
        font-size: 25px;  
        color:#3C7949;  
    }  
}  
}
```



# Compila

- Compilamos el archivo “style.less” igualmente con el comando “lessc style.less style.css” y se creara el archivo “style.css” automáticamente con el siguiente código:

```
.container h1 {  
    font-size: 25px;  
    color: #E45456;  
}  
  
.container p {  
    font-size: 25px;  
    color: #3C7949;  
}  
  
.container .myclass h1 {  
    font-size: 25px;  
    color: #E45456;  
}  
  
.container .myclass p {  
    font-size: 25px;  
    color: #3C7949;  
}
```





# Comprueba el resultado

- Abre el con un navegador

## First Heading

LESS is a dynamic style sheet language that extends the capability of CSS.

## Second Heading

LESS enables customizable, manageable and reusable style sheet for web site.

# Operaciones LESS

- Con LESS es posible realizar operaciones aritméticas tales como suma (+), resta (-), multiplicación (\*) y división (/) y pueden ser realizadas con cualquier numero, color o variable.
  - <http://lesscss.org/features/#features-overview-feature-operations>
- Utilizar operaciones en LESS puede ahorrarnos mucho tiempo cuando las utilizamos con variables.



# Usando Operaciones

# Crea el HTML



```
<html>
  <head>
    <title>Less Operations</title>
    <link rel = "stylesheet" type = "text/css" href = "style.css" />
  </head>

  <body>
    <h1>Example using Operations</h1>
    <p class = "myclass">LESS enables customizable,
      manageable and reusable style sheet for web site.</p>
  </body>
</html>
```

# Crea el LESS



- Creamos el archivo “style.less” :

```
@fontSize: 10px;  
.myclass {  
  font-size: @fontSize * 2;  
  color:green;  
}
```

- Compilamos el archivo para crear automáticamente el archivo “style.css”:

```
.myclass {  
  font-size: 20px;  
  color: green;  
}
```

- Comprueba el resultado

# Funciones LESS

- Las funciones son transformaciones con código javascript, permitiendo la manipulación de valores.
  - <http://lesscss.org/functions/>
- LESS proporciona varias funciones para manipular colores, detectar el tamaño de las imágenes, etc.
- Estas funciones pueden ser las de redondeo, la función de porcentaje, entre otras.



# Usando funciones

# Crea el HTML



```
<html>
  <head>
    <title>Less Functions</title>
    <link rel = "stylesheet" type = "text/css" href = "style.css" />
  </head>

  <body>
    <h1>Example using Functions</h1>
    <p class = "mycolor">LESS enables customizable,
      manageable and reusable style sheet for web site.</p>
  </body>
</html>
```

# Crea el LESS



- Crea el archivo “style.less”:

```
@color: #FF8000;  
@width:1.0;  
.mycolor {  
    color: @color;  
    width: percentage(@width);  
}
```

- Compilamos el archivo para crear automáticamente el archivo “style.css”:

```
.mycolor {  
    color: #FF8000;  
    width: 100%;  
}
```

- Comprueba el resultado

# Mixins

- Permiten mezclar propiedades de estilos existentes
- Se pueden combinar selectores de clases y selectores

```
.a, #b {  
  color: red;  
}  
.mixin-class {  
  .a();  
}  
.mixin-id {  
  #b();  
}
```



```
.a, #b {  
  color: red;  
}  
.mixin-class {  
  color: red;  
}  
.mixin-id {  
  color: red;  
}
```

- Cuando se llama a un mixin, los paréntesis son opcionales. Estas dos opciones producen la misma salida

.a();

.a;

# Selectores

- Mixins puede contener más que sólo propiedades, pueden contener selectores también.

```
.my-hover-mixin() {  
  &:hover {  
    border: 1px solid red;  
  }  
}  
button {  
  .my-hover-mixin();  
}
```



```
button:hover {  
  border: 1px solid red;  
}
```

# Namespaces

- Si desea mezclar propiedades dentro de un selector más complicado, puede apilar múltiples identificaciones o clases.
- El “>” y espacios en blanco son opcionales

```
#outer > .inner;  
#outer > .inner();  
#outer .inner;  
#outer .inner();  
#outer.inner;  
#outer.inner();
```

- Namespacing: se puede poner sus mixins bajo un selector id y esto asegura de que no entrará en conflicto con otra biblioteca.

```
#outer {  
  .inner {  
    color: red;  
  }  
}  
.c {  
  #outer > .inner;  
}
```

```
#my-library {  
  .my-mixin() {  
    color: black;  
  }  
}  
// which can be used like this  
.class {  
  #my-library > .my-mixin();  
}
```

# La palabra clave *!important*

- Utilice la palabra clave *!important* después de la llamada de mixin para marcar todas las propiedades heredadas por ella como importante:

```
.foo (@bg: #f5f5f5, @color: #900) {  
  background: @bg;  
  color: @color;  
}  
.unimportant {  
  .foo();  
}  
.important {  
  .foo() !important;  
}
```



```
.unimportant {  
  background: #f5f5f5;  
  color: #900;  
}  
.important {  
  background: #f5f5f5 !important;  
  color: #900 !important;  
}
```

# Parametric Mixins

- Los mixins también puede recibir argumentos, que son variables pasadas al bloque de selectores cuando se mezcla.
- Definición del mixin con valor por defecto
- Uso del mixin con parámetro

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

```
#header {  
  .border-radius(4px);  
}  
.button {  
  .border-radius(6px);  
}
```

# Mixins con multiples parámetros

- Los parámetros se separan por “,” o “;”. Se recomienda usar punto y coma.
- Es válido definir múltiples mixins con el mismo nombre y número de parámetros.

```
. mixin(@color) {  
    color-1: @color;  
}  
. mixin(@color; @padding: 2) {  
    color-2: @color;  
    padding-2: @padding;  
}  
. mixin(@color; @padding; @margin: 2) {  
    color-3: @color;  
    padding-3: @padding;  
    margin: @margin @margin @margin @margin;  
}  
. some .selector div {  
    . mixin(#008000);  
}
```



```
.some .selector div {  
    color-1: #008000;  
    color-2: #008000;  
    padding-2: 2;  
}
```

# Mixins como funciones

- Las variables y mixins definidos en un mixin son visibles y se pueden usar en el scope del caller.
- Sólo hay una excepción, una variable no se copia si el caller contiene una variable con el mismo nombre (que incluye variables definidas por otra llamada de mixin).
- Sólo se protegen las variables presentes en el ámbito local de los caller. Las variables heredadas de los ámbitos primarios se anulan.

```
.mixin() {  
  @width: 100%;  
  @height: 200px;  
}
```

```
.caller {  
  .mixin();  
  width: @width;  
  height: @height;  
}
```



```
.caller { width: 100%; height: 200px; }
```

# Alcance

- El alcance de la variable especifica el lugar de disponibilidad de la variable.
  - <http://lesscss.org/features/#features-overview-feature-scope>
- Las variables se buscaran desde el ámbito local y si no están disponibles, entonces el compilador buscara desde el ámbito padre.



# Comprobando el alcance

# Crea el HTML



```
<html>
  <head>
    <title>Less Scope</title>
    <link rel = "stylesheet" type = "text/css" href = "style.css" />
  </head>

  <body>
    <h1>Example using Scope</h1>
    <p class = "myclass">LESS enables customizable,
      manageable and reusable style sheet for web site.</p>
  </body>
</html>
```

# Crea el LESS

- Crea el archivo “style.less”:

```
@var: @a;  
@a: 15px;  
  
.myclass {  
    font-size: @var;  
    @a:20px;  
    color: green;  
}
```

- Compila y comprueba el resultado

```
@var: @a;  
@a: 15px;  
  
.myclass {  
    font-size: @var;  
    @a:20px;  
    color: green;  
}
```



# Importación

- LESS ofrece varias extensiones a @import de CSS para proporcionar más flexibilidad sobre lo que puede hacer con archivos externos.  
`@import (keyword) "filename";`
- Se han implementado las siguientes directivas de importación:
  - **reference**: utiliza un archivo Less pero no lo muestra
  - **inline**: incluye el archivo de origen en la salida pero no lo procesa
  - **less**: trata el archivo como un archivo less, sin importar la extensión de archivo
  - **css**: trata el archivo como un archivo CSS, sin importar la extensión de archivo
  - **once**: sólo incluye el archivo una vez (este es el comportamiento predeterminado)
  - **multiple**: incluye el archivo varias veces
  - **opcional**: continúa compilando cuando no se encuentra el archivo

`@import (optional, reference) "foo.less";`

# Loops

- En Less, un mixin puede llamarse a sí mismo.
- Dichas mezclas recursivas, cuando se combinan con Guard Expressions y Pattern Matching, se pueden usar para crear varias estructuras iterativas o bucles.

```
.loop(@counter) when (@counter > 0) {  
  .loop(@counter - 1)); // next iteration  
  width: (10px * @counter); // code for each  
  iteration  
}  
  
div {  
  .loop(5); // launch the loop  
}
```



```
div {  
  width: 10px;  
  width: 20px;  
  width: 30px;  
  width: 40px;  
  width: 50px;  
}
```

2

SASS

# SASS

- SASS (hoja de estilo sintácticamente impresionante), es un preprocesador de CSS, es mas estable y potente que el lenguaje de extensión CSS que describe el estilo de documento estructuralmente.
- <http://sass-lang.com/>
- SASS hace que escribir CSS se mantenable de manera más fácil. Se pueden hacer más cosas, con menos código, más legible y en menos tiempo.



# Características

- Es mas estable potente y compatible con versiones de CSS.
- Es un súper conjunto de CSS y se basa en JavaScript.
- Utiliza su propia sintaxis y compila a CSS legible.
- Es posible escribir CSS en menos código y en menos tiempo.
- Utiliza métodos reutilizables, instrucciones lógicas e incorpora algunas funciones como la manipulación de color, matemáticas y listas de parámetros.

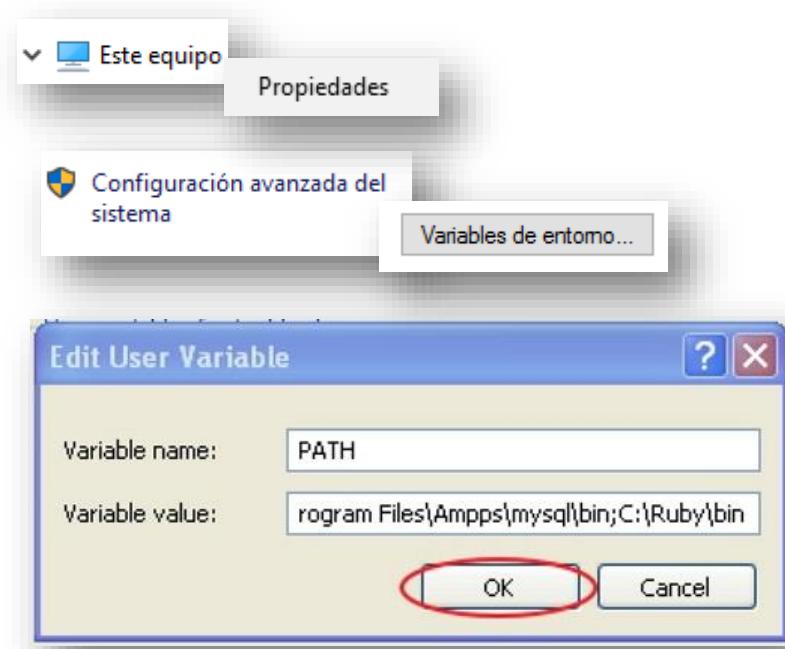


# Instalando SASS



# Instala Ruby

- SASS necesita Ruby para compilar los archivos scss.
- Descarga el archivo comprimido de la versión estable de Ruby del sitio:
  - <https://rubyinstaller.org/>
  - Ejecuta el archivo de instalación
- Añade Ruby al “Path” del sistema
  - Botón derecho en Mi PC → Propiedades → Avanzado y hacemos clic en “Variables del entorno”.
  - En la ventana de variables del entorno, hacemos doble clic en la entrada “PATH”
  - Añadimos “;” y la ruta de la carpeta “bin” de ruby en el campo de “Valor de variable” (ej. ;C:\Ruby\bin) → OK



# Instala Sass



- Abre una ventana de comandos y ejecuta la siguiente orden

```
gem install sass
```

```
C:>gem install sass
Fetching: sass-3.4.24.gem (100%)
Successfully installed sass-3.4.24
Parsing documentation for sass-3.4.24
Installing ri documentation for sass-3.4.24
Done installing documentation for sass after 13 seconds
1 gem installed
```



# Creando un proyecto Sass



# Crea el HTML

- Crea una carpeta de proyecto y añade un index.html

```
<html>
  <head>
    <title> Import example of sass</title>
    <link rel = "stylesheet" type = "text/css" href = "style.css"/>
  </head>

  <body>
    <h1>Primer ejemplo</h1>
    <h3>Welcome to Sass</h3>
  </body>
</html>
```

# Crea el archivo SASS



- Crear el archivo “style.scss”, que tendrá una estructura similar a la del archivo CSS, solo que con la extensión **.scss**.

```
h1{  
    color: #AF80ED;  
}
```

```
h3{  
    color: #DE5E85;  
}
```

# Compilar SASS



- Abre una ventana de comando en la carpeta de proyecto
- Ejecuta la orden

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

- Esta orden estará pendiente de los cambios en el archivo style.scss y generará automáticamente el archivo “style.css”

```
D:\sass\ejemplo1> sass --watch style.scss:style.css
>>> Sass is watching for changes. Press Ctrl-C to stop.
      write style.css
      write style.css.map
>>> Change detected to: style.scss
      write style.css
      write style.css.map
```

- Para terminar el comando pulsa **CTRL+C**



# Comprueba el resultado

- Abre el archivo index.html en un navegador



# Sintaxis SASS

- SASS soporta dos tipos de sintaxis, SCSS y sintaxis con sangría.
- SCSS (Sassy CSS) es una extensión de la sintaxis de CSS, lo que significa que un archivo CSS es valido también como SCSS.
- La mayoría de los archivos CSS y SCSS utilizan la extensión .scss.
- La sintaxis con sangría es la sintaxis mas antigua y en muchas ocasiones llamada sencillamente SASS, los archivos SASS utilizan la extensión .sass.

# Sintaxis SASS

- La sintaxis con sangría utiliza sangría en lugar de { y } para delimitar bloques.
- Para estados separados, se utiliza el salto de línea en lugar de punto y coma (;).
- La declaración y selectores de propiedades deben ser colocados en su propia línea y los estados dentro de { y } deben ser colocados en la nueva línea con sangría.
- Veamos un ejemplo:

```
.myclass {  
    color = red;  
    font-size = 0.2em;  
}
```

# Selectores de varias líneas

- En la sintaxis con sangría, los selectores pueden ser colocados en una nueva línea, siempre que aparezcan después de comas. Veamos un ejemplo:

```
.class1,  
.class2{  
    color:red;  
}
```

# Extensiones de CSS

- Las extensiones de CSS se pueden utilizar para mejorar la funcionalidad de las paginas.
- Las extensiones mas usadas son:
  - Reglas anidadas
  - Referencia al selector padre: &
  - Propiedades anidadas
  - Selector de marcador de posición (placeholder)

# Reglas anidadas

- Es una forma de combinar varias reglas CSS dentro de otra.

## ➤ SCSS

```
.container{  
  h1{  
    font-size: 25px;  
    color:#E45456;  
  }  
  
  p{  
    font-size: 25px;  
    color:#3C7949;  
  }  
  
  .box{  
    h1{  
      font-size: 25px;  
      color:#E45456;  
    }  
  
    p{  
      font-size: 25px;  
      color:#3C7949;  
    }  
  }  
}
```



## ➤ CSS

```
.container h1 {  
  font-size: 25px;  
  color: #E45456;  
}  
  
.container p {  
  font-size: 25px;  
  color: #3C7949;  
}  
  
.container .box h1 {  
  font-size: 25px;  
  color: #E45456;  
}  
  
.container .box p {  
  font-size: 25px;  
  color: #3C7949;  
}
```

# Selectores padre: &

- Se puede seleccionar el selector padre usando el carácter &. Indica dónde se debe insertar el selector padre.

## ➤ SCSS

```
a {  
  font-size: 20px;  
  &:hover { background-color:  
    yellow; }  
}
```



## ➤ CSS

```
a {  
  font-size: 20px;  
}  
a:hover {  
  background-color: yellow;  
}
```

# Ejemplo de propiedades anidadas

- Permite anidamiento de propiedades en otras propiedades que conducen a la agrupación de otro código relacionado.

## ➤ SCSS

```
.line {  
  font: {  
    family: Lucida Sans Unicode;  
    size: 20px;  
    weight: bold;  
    variant: small-caps;  
  }  
}
```



## ➤ CSS

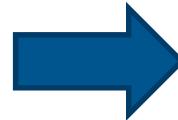
```
.line {  
  font-family: Lucida Sans  
  Unicode;  
  font-size: 20px;  
  font-weight: bold;  
  font-variant: small-caps;  
}
```

# Selector placeholder

- Sass soporta el selector placeholder usando la clase o el selector ID, usando la directiva “@extend”.

## ➤ SCSS

```
.frst_para {  
  color: green;  
}  
.sec_para {  
  @extend .frst_para;  
  font-size:20px;  
}
```



## ➤ CSS

```
.frst_para, .sec_para {  
  color: green;  
}  
.sec_para {  
  font-size: 20px;  
}
```

# Comentarios

- SASS es compatible con dos tipos de comentarios:
- Comentarios multilinea:
  - Estos comentarios se escriben con “/\*” y “\*/”. Los comentarios multilineas se conservan en la salida de CSS.
- Comentarios de una linea:
  - Estos se escriben utilizando “//” seguido del comentario. Los comentarios de una linea no se conservan en la salida CSS.

```
/* This comment is
 * more than one line long
 * since it uses the CSS comment syntax,
 * it will appear in the CSS output. */
body { color: black; }

// These comments are in single line
// They will not appear in the CSS output,
// since they use the single-line comment syntax.
a { color: blue; }
```



```
/* This comment is
 * more than one line long
 * since it uses the CSS comment syntax,
 * it will appear in the CSS output. */
body {
  color: black; }

a {
  color: blue; }
```

# Directivas Mixins

- Los mixins permite crear un grupo de estilos, que se pueden reutilizar en toda la hoja de estilos.
- Pueden almacenar múltiples valores o parámetros y funciones de llamada, lo que ayuda a evitar repetir código.
- La siguiente tabla muestra

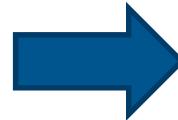
Directiva	Descripción
Definición de un mixin	
Inclusión de un mixin	La directiva @include incluye un mixin en el documento
Argumentos	Los valores de SassScript pueden tomarse como argumentos en los mixins. Estos se insertan cuando se incluye mixin. En el mixin están disponibles como variable.
	Los bloques de estilo son pasados al mixin.

# Definición de un mixin

- Se usas la directiva **@mixin** para definir el mixin
- Con **@include** se llama al mixin

## ➤ SCSS

```
@mixin style {  
  .cont{  
    color: #77C1EF;  
  }  
}  
  
@include style;
```



## ➤ CSS

```
.cont {  
  color: #77C1EF;  
}
```

# Argumentos de un mixin

- Los valores de SassScript se pueden tomar como argumentos en mixins, que se pasan cuando se incluye mixin y están disponibles como variables dentro del mismo.
- El argumento es el nombre de una variable, que está separada por comas al definir un mixin.
- Hay dos tipos de argumentos:
  - Argumentos de palabras clave
  - Argumentos Variables

# Argumentos de palabras clave

- Los argumentos se pueden pasar en cualquier orden y se pueden omitir los valores predeterminados de argumento.

## ➤ SCSS

```
@ mixin bordered($color,  
$width: 2px) {  
  color: #77C1EF;  
  border: $width solid black;  
  width: 450px;  
}
```

```
.style {  
  @include  
  bordered($color:#77C1EF,  
  $width: 2px);  
}
```



## ➤ CSS

```
.style {  
  color: #77C1EF;  
  border: 2px solid black;  
  width: 450px;  
}
```

# Argumentos Variables

- Los argumentos variables se utilizan para pasar un número no definido de argumentos al mixin.
- Los argumentos de palabras clave pasados al mixin se pueden acceder usando la función `keywords($args)` que devuelven valores asignados a String.

## ➤ SCSS

```
@mixin colors($background) {  
  background-color: $background;  
}  
  
$values: magenta, red, orange;  
.container {  
  @include colors($values...);  
}
```



## ➤ CSS

```
.container {  
  background-color: magenta;  
}
```

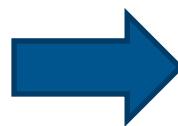
# Pasando bloques de contenido a un mixin

- Un bloque de estilos se puede pasar al mixin para su inserción como estilo.
- Los estilos se incluyen en la ubicación de directiva @content.

## ➤ SCSS

```
@mixin element {  
  @content;  
}
```

```
@include element {  
  .block {  
    color: green;  
  }  
}
```



## ➤ CSS

```
.block {  
  color: green;  
}
```

# Directivas de función

- En SASS se puede crear una función propia y utilizarla en el contexto del script o con cualquier valor.
- Las funciones se llaman usando el nombre de la función, con sus parámetros.



# Usando directiva de funciones

# EI HTML



## ➤ Creamos el index.html

```
<html>

<head>
  <title>Nested Rules</title>
  <link rel = "stylesheet" type = "text/css" href = "style.css" />
</head>

<body>

<div class = "container" id = "set_width">
  <h2>Example for Function directives</h2>
  <p>SASS stands for Syntactically Awesome Stylesheet. </p>
</div>

</body>

</html>
```

# El .scss



- Definimos le archivo style.scss

```
$first-width: 5px;  
$second-width: 5px;  
  
@function adjust_width($n) {  
  @return $n * $first-width + ($n - 1) * $second-width;  
}  
  
#set_width { padding-left: adjust_width(10); }
```

# Compilamos



- Cada vez que se modifique el archivo .scss, se debe ejecutar el comando:

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

- Se generara o actualizara automáticamente el archivo .css:

```
#set_width {  
    padding-left: 95px;  
}
```

- Comprobamos abriendo el index.html en el navegador

# Estilos de Salida

- El archivo CSS que SASS genera tiene el estilo de CSS por defecto, que se refleja en la estructura del documento.
- La salida por defecto es buena, pero en ocasiones podría no ser adecuada para todas las situaciones, por lo que SASS es compatible con otros estilos de salida.
- Los estilos disponibles son:
  - :nested
  - :expanded
  - :compact
  - :compressed
- La orden para definir la salida se invoca de la siguiente manera:

```
sass --watch style.scss:style.css --style compressed
```

# Estilos de Salida: fuente scss

- Consideremos el siguiente código Sass para observar los distintos tipos de salida

```
.widget-social {  
    text-align: right;  
  
    a,  
    a:visited {  
        padding: 0 3px;  
        color: #222222;  
        color: rgba(34, 34, 34, 0.77);  
    }  
  
    a:hover {  
        color: #B00909;  
    }  
}
```

# :nested

- Es el estilo por defecto de SASS.
- Es muy útil cuando el archivo CSS es demasiado grande, esto hace que la estructura del archivo sea mas legible y pueda ser comprendido fácilmente.
- Por ejemplo:

```
.widget-social {  
    text-align: right; }  
.widget-social a,  
.widget-social a:visited {  
    padding: 0 3px;  
    color: #222222;  
    color: rgba(34, 34, 34, 0.77); }  
.widget-social a:hover {  
    color: #B00909; }
```

# :expanded

- Se necesita mas espacio en comparación con estilo anidado.
- La sección de reglas consiste en propiedades, las cuales están todas dentro de las reglas, mientras que las reglas no sigan sumando sangría.
- Ejemplo:

```
.widget-social {  
    text-align: right;  
}  
.widget-social a,  
.widget-social a:visited {  
    padding: 0 3px;  
    color: #222222;  
    color: rgba(34, 34, 34, 0.77);  
}  
.widget-social a:hover {  
    color: #B00909;  
}
```

# :compact

- Ocupa menos espacio que las otras formas.
- Se centra principalmente en los selectores (en vez de en sus propiedades)
- Cada selector y sus propiedades se colocan en la misma salida.
- Ejemplo:

```
.widget-social { text-align: right; }
.widget-social a, .widget-social a:visited { padding: 0 3px; color: #222222;
color: rgba(34, 34, 34, 0.77); }
.widget-social a:hover { color: #B00909; }
```

## :compressed

- Ocupa el menor espacio en comparación con los otros estilos mencionados anteriormente.
- Proporciona espacios en blanco solo a los selectores y nueva línea al final del archivo.
- Este forma de salida es confusa y no es fácil de leer.
- Ejemplo

```
.widget-social{text-align:right}.widget-social a,.widget-social a:visited{padding:0  
3px;color:#222222;color:rgba(34,34,34,0.77)}.widget-social a:hover{color:#B00909}
```



## BananaTube: Decisión en equipo

- Discute en equipo la conveniencia de usar Less, Sass o CSS plano para BananaTube
- Tomad nota de las fortalezas y debilidades que hayas observado en cada caso
- Implementa la solución en función de la decisión



[...]**netmind**

WeKnowIT

Barcelona

C. Almogàvers, 123  
08018 Barcelona  
Tel. 93 304.17.20  
Fax. 93 304.17.22

Madrid

Plaza Carlos Trías Bertrán, 7  
28020 Madrid  
Tel. 91 442.77.03  
Fax. 91 442.77.07

[www.netmind.es](http://www.netmind.es)



GOBIERNO  
DE ESPAÑA

MINISTERIO  
DE ENERGÍA, TURISMO  
Y AGENDA DIGITAL

**red.es**



ESTRATEGIA DE  
EMPRENDIMIENTO Y  
EMPLEO JUVENIL  
*garantía juvenil*



Agenda Digital para España



**UNIÓN EUROPEA**

Fondo Social Europeo  
*"El FSE invierte en tu futuro"*