

Tecnologías Web: Cliente

Sass.

Pre-procesadores CSS

- Extienden el lenguaje de hojas de estilo CSS
- CSS más limpio y flexible, facilita el mantenimiento.
- Reduce la repetición, es más productivo que CSS
- Características para especificar estilos de forma modular.
- No son reconocidos por el navegador, necesitamos instalar el procesador y generarlo antes del despliegue.
- Sass, Less, Stylus



Sass

- Dos tipos de sintaxis:
 - .scss Similar a CSS
 - .sass Sintaxis indentada en lugar de corchetes y saltos de línea en lugar de ;
- Instalación:
 - Gestor de paquetes de ruby: gem

```
$ sudo apt-get install ruby  
  
$ sudo gem install sass  
  
$ sass -v
```



Variables

- Las variables nos permiten declarar - almacenar información que se puede reutilizar más adelante
- Tipos admitidos:
 - strings, números, colores, booleanos, listas, nulos
- Sintaxis:
 - `$nombre: valor`

```
$myFont: sans-serif;  
$myFontSize: 16px;  
  
body {  
  font-family: $myFont;  
  font-size: $myFontSize;  
}
```

```
body {  
  font-family: sans-serif;  
  font-size: 16px;  
}
```



Anidamiento

- Sass permite anidar selectores como en HTML
- Simplificar sintaxis de propiedades con el mismo prefijo mediante anidamiento.
 - **font-size, font-family, font-weight,**

```
prefijo
{
    sufijo1: valor1
    sufijo2: valor2
}
```

```
p {
  font: {
    family: sans-serif;
    size: 16px;
    weight: bold;
  }
}
```

```
p {
  font-family: sans-serif;
  font-size: 16px;
  font-weight: bold;
}
```



Operadores

- Matemáticos: **+**, **-**, *****, **/**, **%**
- **&**: Referencia a su elemento padre.

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

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



Parciales - @import

- Ficheros con poco código que se incluyen en otros ficheros Sass.
- Se importan en cualquier fichero Sass con **@import**
- Ejemplos: Colores, fuentes, variables, resets, etc.
- Si no queremos que se transpile: **_nombrefichero.scss**

```
html,  
body,  
ul,  
ol {  
  margin: 0;  
  padding: 0;  
}  
  
reset.scss, _reset.scss
```

```
@import "reset.scss"  
  
body{  
  font-size: 16px  
}
```

```
html,  
body,  
ul,  
ol {  
  margin: 0;  
  padding: 0;  
}  
  
body{  
  font-size: 16px  
}
```



Reutilizar código

- Reutilizar código en el sitio web:
 - Declarar el módulo con **@mixin**
 - Un mixin puede incluir otros mixin
 - Importar el módulo con **@include**

```
@mixin name {  
  property: value;  
  property: value;  
  ...  
}
```

```
@mixin error-text {  
  color: red;  
  font-size: 16px;  
  font-weight: bold;  
}
```

```
.error {  
  @include error-text;  
  background-color: green;  
}
```



Herencia

- Se usa para estilos que son compartidos, en su mayoría, y que se aplican con alguna diferencia según qué elemento:
 - Declarar el estilo base
 - Incorporarlo en el estilo que hereda con **@extend**
 - También se puede usar con **%nombre_estilo**

```
.button-basic {  
  border: none;  
  padding: 15px 30px;  
  text-align: center;  
}  
  
.button-report {  
  @extend .button-basic;  
  background-color: red;  
}  
  
.button-submit {  
  @extend .button-basic;  
  background-color: green;  
}
```

```
.button-basic, .button-report, .button-submit  
{  
  border: none;  
  padding: 15px 30px;  
  text-align: center;  
}  
  
.button-report {  
  background-color: red;  
}  
  
.button-submit {  
  background-color: green;  
}
```



Funciones

- Color: set, get o manipulación
 - `rgb(red, green, blue)`, `rgba(red, green, blue, alfa)`, `hsl(hue, saturation, lightness)`, `hsla(hue, saturation, lightness, alpha)`, `grayscale(color)`, `mix(color1, color2, weight)`, `adjust-color(color, red, green, blue, hue, saturation, lightness, alpha)`, ...
- Cadenas:
 - `quote(string)`, `str-length(string)`, `str-index(string, substring)`, `to-lower-case(string)`, ...
- Números:
 - `abs(number)`, `ceil(number)`, `comparable(num1, num2)`, `max(number...)`, ...



Mapas:

- Lista de pares: clave-valor.
- `$map: (key1: value1, key2: value2, key3: value3);`
- Funciones:
 - `map-get(map, key)`, `map-has-key(map, key)`, `map-merge(map1, map2)`,
`map-remove(map, keys...)`

Sentencias de control:

```
@if expresión {           @each $var in <lista o mapa>{
    ...                     ...
}                             }
@else {
    ...
}                             $var from <inicio> through[to] <final> {
                                ...
                                }
```



Ejemplos:

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

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

```
@each $publicacion in libro, revista, apuntes {  
  .#{$publicacion}-icon {  
    background-image:  
      url('/images/#{ $publicacion }.png');  
  }  
}
```

```
.libro-icon {  
  background-image: url("/images/libro.png");  
}  
  
.revista-icon {  
  background-image: url("/images/revista.png");  
}  
  
.apuntes-icon {  
  background-image: url("/images/apuntes.png");  
}
```



