



CSS preprocessor: **Sass**

# Versions

PDF file in  <http://goo.gl/HSlbLh>

HTML file: <http://www.slideshare.net/z32s/>



Esta obra está bajo una Licencia Creative Commons Atribución-CompartirlGual 4.0 Internacional.



Ahora mismo los enlaces no funcionan en chrome. Por favor, usa safari o bájate el PDF desde el drive. Thanks



<http://sass-lang.com/>

# ¿Qué es Sass?

Herramienta-sintaxis para mejorar el flujo de trabajo de las CSS.

# ¿Para qué?

Nos permitirá introducir variables, operaciones, funciones... en nuestro CSS.

**Es como meterle esteroides.**

# ¿Cómo se usa?

Básicamente tendremos un **clon de nuestro CSS** en el proyecto, donde podremos escribir con una sintaxis especial y añadir estas funciones, variables...

**Se compilará automáticamente**, y **se generará el CSS** normal resultante que es el que se subirá a producción.

# ¿Ventajas?

Más rapidez en el trabajo

Crear **código reutilizable** proyecto tras proyecto

Tener un código **fácil de mantener**



**¡Paciencia!** Todos somos  
capaces de hacerlo

# La instalación

Abrir la terminal

win: **gem install sass**

mac: **sudo gem install sass**

En windows es necesario **instalar ruby**  primero

# La compilación

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

**sass --watch dir\_scss:dir\_css**

También se puede usar para compilar: Compass<sup>↗</sup>, Grunt<sup>↗</sup>, prepros<sup>↗</sup>, codekit<sup>↗</sup>...

# Arquitectura del proyecto Sass

## Fragmentar y organizar

- Architecture for a Sass Project [↗](#)
- docssa.info [↗](#)
- Our CSS/Sass Project Architecture and Styleguide [↗](#)

# La sintaxis

---

## SCSS

```
$blue: #3bbfce ;  
$margin: 16px ;  
.content-navigation {  
    border-color: $blue ;  
    color: darken($blue, 9%) ;  
}  
.border {  
    padding: $margin / 2 ;  
    margin: $margin / 2 ;  
    border-color: $blue ;  
}
```

## SASS

```
$blue: #3bbfce;  
$margin: 16px;  
.content-navigation  
    border-color: $blue;  
    color: darken($blue, 9%);  
.border  
    padding: $margin / 2;  
    margin: $margin / 2;  
    border-color: $blue;
```

# La **base**

Variables

Mixins

Nesting

Inheritance

Import

Color functions

Operations

Functions

# Variables

---

```
$  
$color: #0982c1;  
$width: 1024px;  
$style: dotted;  
  
body {  
    color: $color ;  
    border: 1px $style $color ;  
    max-width: $width ;  
}
```

CSS compilado

```
body {  
    color : #0982c1;  
    border : 1px dotted #0982c1;  
    max-width : 1024px;  
}
```

# Mixins

---

```
@mixin error ($width: 2px) {  
    border: $width solid #F00;  
    color: #F00;  
}  
.generic-error {  
    padding: 20px;  
    @include error();  
}  
.login-error {  
    width: 100px;  
    @include error(5px);  
}
```

```
.generic-error {  
    padding: 20px;  
    border: 2px solid #f00;  
    color: #f00;  
}  
.login-error {  
    width: 100px;  
    border: 5px solid #f00;  
    color: #f00;  
}
```

[info]: Emmet  Utilizar este tipo de utilidades para crear los vendor.

# Nesting

---

```
section {  
  margin : 10px;  
  nav {  
    height : 25px;  
    a {  
      color : #0982C1;  
      & :hover {  
        text-decoration :  
        underline;  
      }  
    }  
  }  
}
```

```
section {  
  margin : 10px;  
}  
section nav {  
  height : 25px;  
}  
section nav a {  
  color : #0982C1;  
}  
section nav a :hover {  
  text-decoration :  
  underline;  
}
```

# Inheritance

---

```
.block {  
  margin: 10px 5px;  
  padding: 2px;  
}  
p {  
  @extend .block ;  
  border: 1px solid #EEE;  
}  
ul, ol {  
  @extend .block ;  
  color: #333;  
}
```

```
.block, p, ul, ol {  
  margin : 10px 5px;  
  padding : 2px;  
}  
p {  
  border : 1px solid #EEE;  
}  
ul, ol {  
  color : #333;  
}
```

# @import

---

```
/* file .scss */
body {
  background: #EEE;
}



---


@import "reset.css";
@import "file";
p {
  background: #0982C1;
}
```

```
@import "reset.css";
body {
  background: #EEE;
}
p {
  background: #0982C1;
}
```

# Color functions

```
lighten(@color, 10%); /*10% lighter than @color*/  
darken(@color, 10%); /*10% darker than @color*/  
saturation(@color, 10%); /*10% more saturated than @color*/  
desaturation(@color, 10%); /*10% less saturated than @color*/
```

---

```
$color: #0982C1;  
h1 {  
    background: $color;  
    border: 3px solid darken($color, 50%);  
}  
[doc] [article]
```

# Operations

**\$width**:960px;

**\$items**:6;

```
body {  
    margin: (14px / 2);  
    top: 50px + 100px;  
    right: 100px - 50px;  
    left: 10 * 10;  
}  
p {width: $width / 2;}  
li {  
    float: left;  
    width: $width / $items - 10px;  
    padding: 0 5px;  
}
```

# Recursos

- [sass-lang.com](http://sass-lang.com) ↗
- [Compass](#) ↗
- [Codekit](#) ↗ (sólo mac)
- [Scout](#) ↗
- [css2sass](#) ↗
- [Firesass](#) ↗
- [Sass inspector](#)

# Consejos de uso

Si tenemos una **mala estructura** de CSS el preprocesador **no lo solucionará**.

Sólo el hecho de usarlo no significa que lo estemos haciendo bien.

La **calidad** del código no lo da el preprocesador, **lo damos nosotros**.

# Dry

- Don´t Repeat Yourself
- Código limpio
- Mantenible
- Reutilizable

# Código reutilizable

- La filosofía del código reutilizable casa muy bien con los preprocesadores.
- El preprocesador **fomenta la modularización**
- Los @import nos permiten **unificar todos los archivos scss en uno**

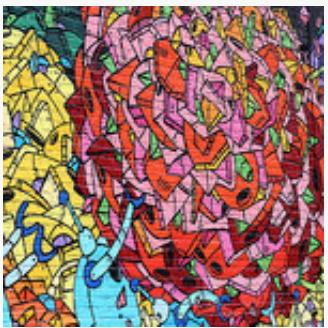
# Trampas

- **El mixin, duplica el código.** La facilidad de su uso puede hacer que abusemos de él, y tengamos un código muy hinchado.
- El **nesting** de mucho niveles causa un **código muy dependiente y poco flexible**. Hay que **evitar los anidamientos profundos**.

# Soluciones

- Mirar el **código resultante**. Si está lleno de repeticiones, plantearse **refactorizar** más el código.
- Pensar bien la acciones que tomemos. **No por el hecho de que se puedan hacer hay que aplicarlas**. Mirar lo que aporta.

# Atribuciones



Sass



