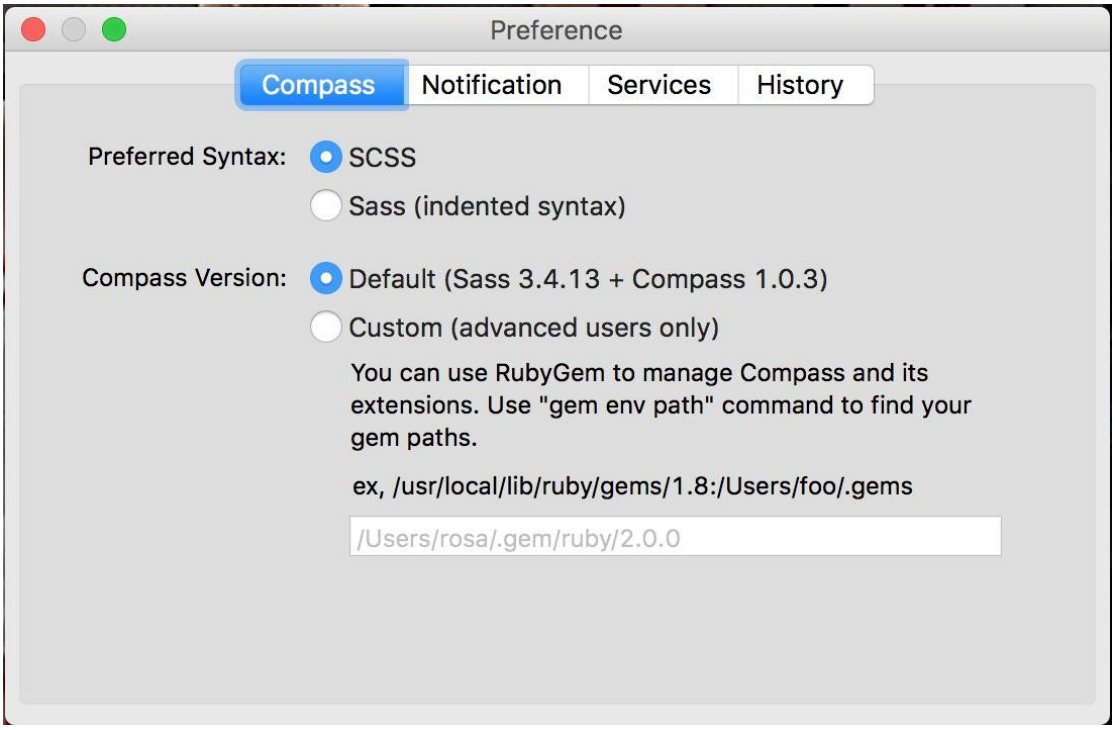
Mas sobre Compass

Profesora: **Rosa María Medina Gómez**
Curso de Formación del Profesorado a distancia
Cefire Específico de FP de Cheste
Generalitat Valenciana
Curso 2018-2019

Más sobre compass

¿Cómo modificar las preferencias de la sintaxis en Compass.App?

Desde botón derecho sobre el icono del sistema podemos modificar las preferencias de Compass.App:



Esta captura ha sido realizada con MacOSX, pero es exactamente igual en Linux y Windows.

Si lo vamos a hacer desde consola, descomentaremos en el archivo de configuración la línea: `preferred_syntax = :sass`

Y ejecutaremos desde el directorio donde tenemos nuestro proyecto (con esto se cambiarán nuestros archivos .scss a .sass):

```
sass-convert -R --from scss --to sass sass scss && rm -rf sass && mv scss sass
```

Sintaxis indentada

Una de las diferencias que podemos encontrar entre sass y scss es la indentación. En lugar de utilizar las llaves como señal de que el

contenido que hay entre las llaves pertenece al mismo ámbito/bloque, en sass se utiliza la indentación de la misma forma que se hace con Python. Ejemplo:

application.scss	**application.sass**

tent{
.con border: 1px solid #ccc;
padding: 20px; h2{
font-size: 3em; margin:
20px 0;
}
 	
}
	.content
	border: 1px solid #ccc
	padding: 20px
	h2
	font-size: 3em margin: 20px
	0

¡Importante! Si os fijáis cada sentencia no se separa por punto y coma ya en .sass

Sintaxis indentada + mixin

Otra de las ventajas ya que es "más eficiente" es el uso y declaración del mixin.

application.scss	**application.sass**

@mixin box-sizing($x){
-webkit-box-sizing: $x; -moz-box-
sizing: $x;
box-sizing: $x;
}
.content{
@include: box-sizing(border-
box); }
	=box-sizing($x)
	-webkit-box-sizing: $x -moz-
	box-sizing: $x
	box-sizing: $x
	.content
	+box-sizing(border-box)

El + lo usamos para indicar que es un include y el = una definición de un mixin.

Listas

Los elementos de una lista pueden estar separados por un espacio o por una coma.

application.sass

```
$authors: rosa jose lucia pablo
$designers: manolo, pepe, maria
```

Y para recorrer el listado de elementos, sería:

application.sass

```
$authors: rosa jose lucia pablo
$each $a in $authors
     .author-#{$a}
          background: url(#{$a}.jpg)
```

De forma que obtendremos:

application.css

```
.author-rosa{
     background: url(rosa.jpg);
}
.author-jose{
     background: url(jose.jpg);
}
.author-lucia{
     background: url(lucia.jpg);
}
.author-pablo{
     background: url(pablo.jpg);
}
```

Además, también tenemos un conjunto de funciones para trabajar con listas, como puede ser:

- **length**: devuelve el número de elementos de una lista
- **append**: Devuelve la primera lista, añadiendo al final el nuevo ítem que le pasamos como segundo parámetro
- **join**: Combina dos listas
- **index**: Devuelve la posición donde se encuentra un determinado ítem y si no existe devuelve false. **¡OJO!**, a diferencia de otros lenguajes la primera posición es 1.
- **nth**: Devuelve el elemento n de una lista
- **zip**: Combina dos listas, devolviendo la combinación de cada una de ellas ítem a ítem y separadas por comas (primer ítem de la lista 1 con el primer ítem de la lista 2, segundo ítem de la lista 1 con el segundo ítem de la lista 2, etc.=

Ejemplo:

application.sass

```
$authors: rosa jose lucia pablo
length($authors); // Devuelve 4
append($authors, marga); // rosa jose lucia pablo marga
```

```
$list: arale goku
$list2: doraemon pokemon
join($list, $list2); // arale goku doraemon pokemon
index($authors, lucia); // 3, IMPORTANTE NO EMPIEZA EN
0 index($authors, paco); // false nth($authors, 3); //
lucia
zip($list, $list2); // arale doraemon, goku pokemon
```

¿Cuándo nos puede resultar útil usar el método zip?
Si tenemos dos listas como pueden ser los autores y dependiendo del autor tenemos que usar un color de fondo distinto, podríamos hacer:

```
$authors: nick aimee dan drew
$colors: green blue red yellow

$author-style: zip($authors, $colors)

@each $a in $author-style     .author-
#{nth($a,1)}
     background: nth($a,2)
```

Obteniendo el css:

```
.author-nick{
     background: green;
}
.author-aimee{
     background: blue;
}
.author-dan{
     background: red;
}
.author-drew{
     background: yellow;
}
```

Escalado de colores

En la sesión anterior vimos funciones para trabajar con los colores, como lighten y darken, de forma que, si le aplicábamos un aclarado al color #eee del 7%, esta función nos devolvía el color White, siendo muy fácil alcanzar este valor con un porcentaje bastante bajo.
Sin embargo, si usamos la función scale_color vamos a obtener el color relativo. Por ejemplo:

application.sass

```
.content
```

```
    color: scale_color(#eee, $lightness: 7%)
```

Por tanto, ahora en lugar de darnos directamente el color blanco, nos genera: **application.css**

```
.content{
    color: #efefef; /*7% entre #eee y
white*/ }
```

Condicionales simples

Al principio de la unidad 4, vimos cómo hacer condicionales if-else, en este caso, vamos a ver cómo hacerlas, pero como ternarias. if(condición, "valor_cierto", "valor_falso")

application.scss	application.sass
`$theme: light; header{ @if $theme == dark { bakcground: #000; } @else { background: #fff: } }`	`$theme: light header background: if($theme == dark, #000, #fff)`

Compass

En la unidad anterior, importamos "compass", pero ¿Qué contiene realmente eso? Al importar compass estamos añadiendo 5 módulos principales: utilities, typography, css3, layout y reset.

Una vez que hemos instalado desde consola compass, podemos importar compass, o únicamente los módulos que necesitemos:

```
@import "compass"
@import "compass/layaut"
```

Cómo simplificar los prefijos de las propiedades de CSS con mixin

También vimos cómo usar un mixin para asignar un valor dependiendo del tipo de navegador que estamos usando:

application.sass	**application.css**

```
=box-sizing($x)
  -webkit-box-sizing: $x   -moz-box-sizing: $x
     box-sizing: $x
.content
     +box-sizing(border-box)
```

```
.content{
     -webkit-box-sizing:border-box;
     -moz-box-sizing: border-box;
     box-sizing: border-box;
}
```

Al incluir compass esto ya no es necesario, si usamos propiedades como box-sizing nos genera un CSS donde aparecen especificados las propiedades para cada navegador.

application.sass

```
@import "compass"
.content
     +box-sizing(border-box)
```

application.css

```
.content{
     -webkit-box-sizing:border-box;
     -moz-box-sizing: border-box;
     box-sizing: border-box;
}
```

Otro de los usos de los mixin en compass puede ser: background (gradients), border-radius, box-shadow, columns, transform, transition, pero siempre debemos asegurarnos que el CSS que se genera es soportado para el proyecto que estamos desarrollando. Por ejemplo, si queremos obtener un gradiente:

application.css

```
.content {
  background: -webkit-gradient(linear, 50% 0%, 50% 100%, color-stop(0%, #9b9592), color-stop(100%, #3c3733));
  background: -moz-linear-gradient(top, #9b9592, #3c3733);
background: -webkit-linear-gradient(top, #9b9592, #3c3733);
background: linear-gradient(to bottom, #9b9592, #3c3733); }
```

Nos bastaría con poner en sass:

application.sass

```
@import compass

.content
    +background(linear-gradient(top, #9b9592, #3c3733))
```

En la unidad anterior vimos cómo se utilizaban los métodos scalelightness o scale_color. Si queremos "escalar" un color dándole un toque de aclarado, podemos:

application.sass

```
.content
    color: scale-lightness(#eee, 7%)
```

Del mismo modo, podíamos haberle puesto un valor negativo al porcentaje del aclarado consiguiendo el efecto de oscurecer en lugar de aclarar.

Otra función como es, scale-saturation también es un atajo de scale_color:

application.sass

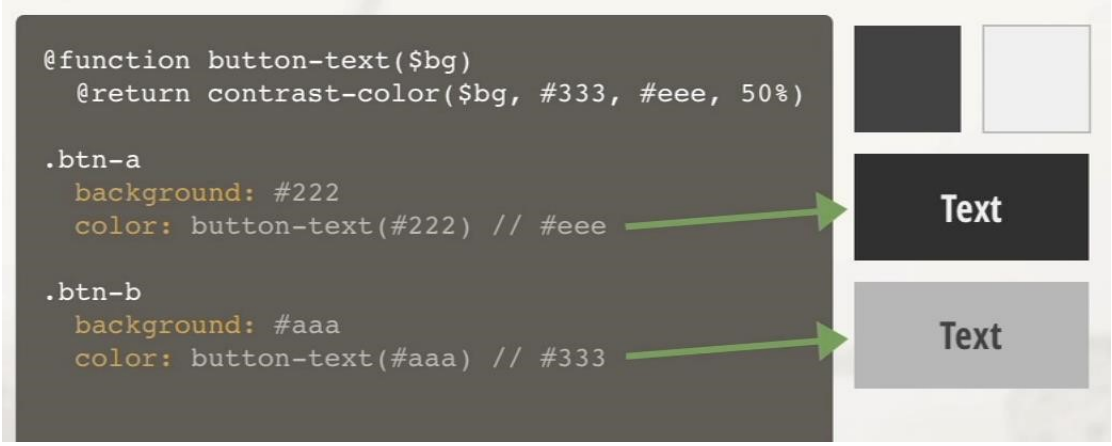
```
.content
    color: scale-saturation(#a99f88, 30%)
```

La función contrast-color, nos devuelve un color oscuro o claro basado en el color que le pasamos por argumento y el porcentaje:

application.sass

```
 contrast-color(input-color, dark-color, light-color,
threshold)
```

Con este método podemos hacer que, en función del color de fondo de un botón, el color del texto varíe. Ejemplo:



Dentro del bloque de funciones para los colores, Compas incluye dos nuevas funciones: shade y tint. Más funciones: http://compassstyle.org/reference/compass/helpers/colors/

stretch

Asigna la distancia de cada lado de un elemento en relación a su contenedor:
stretch(0,0,0,0) // top, right, bottom, left

Dimensiones de una imagen

Tenemos las funciones image-height e image-width, estos métodos son muy útiles cuando queremos insertar un logo dentro de nuestra web ya que, si este es modificado por otro, no deberemos indicar a nuestro CSS las dimensiones exactas de nuestra imagen, sino que con estos métodos se autocalculan.
Ejemplo:

application.sass

```
.logo
    background: image-
url('sass.png')     height: image-
height('sass.png')     width: image-
width('sass.png')
```

Espaciado

Con sass, podemos estalecer un tamaño de fuente y un espaciado en el texto o "tamaño de una línea", para poder conseguir esto, deberemos utilizar `+establish-baseline`, ejemplo:

application.sass

```
$base-font-size: 18px
$base-line-height: 30px
+establish-baseline

blockquote
    // El contenido del blockquote voy a hacer que tenga el espacio de 2 líneas
    +adjust-leading-to(2)
background: #ccc
```

En relación con el tamaño de letra de una web, podemos utilizar `+adjust-font-size-to` de forma que la letra estará determinada a la resolución, ya que le indicaremos el número de píxeles que ocupará. Ejemplo:

application.sass

```
h1
    +adjust-font-size-to(50px)
```

Pero ojo, si vamos a usar `adjust-font-size-to` y `adjust-leadingto` debemos especificarle como segundo parámetro que se ajuste al tamaño de letra especificado para que quede bien ajustado:

application.sass

```
blockquote
    +adjust-font-size-to(20px)     +adjust-
leading-to(2, 20px)     background: #ccc
```

Otra de las funciones dentro de este apartado son los mixin leader y trailer, donde te dejan el espacio correspondiente a una (o varias) línea al principio de un párrafo (leader) o al final (trailer)

application.sass

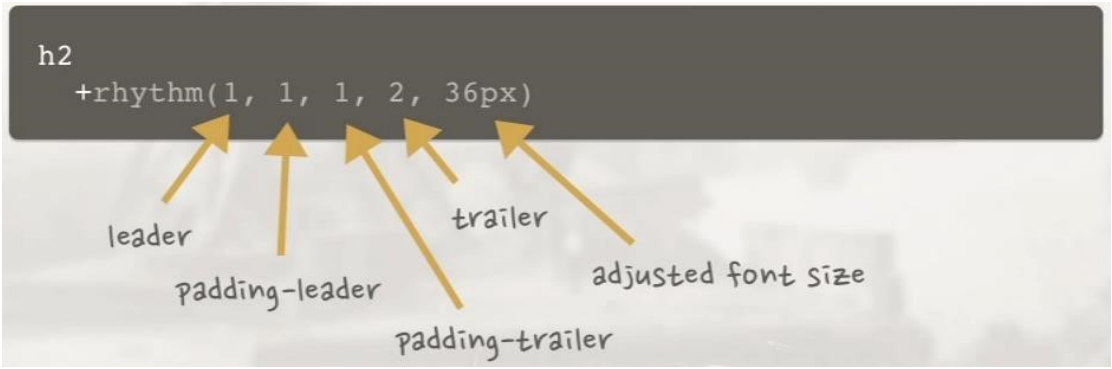
```
p
    +leader(1)
    +trailer(1)
```

Algo parecido ocurre con trailer si usamos adjust-font-size-to, debemos indicarle los píxeles que empleamos para que ocupe la parte correspondiente.

application.sass

```
h1
    +adjust-font-size-to(50px)
    +trailer(2, 50px)
```

Otros mixin sobre el espaciado son: padding-leader, padding-trailer y rythm, donde si vamos a usar leader, padding-leader, trailer, y/o padding-trailer lo más correcto sería pasar a usar rythm



Como vemos, tenemos muchos métodos para ajustar de forma vertical el texto en nuestra web.
http://compass-style.org/reference/compass/typography/vertical_rhythm/

Bibliografía y/o páginas de interés

- Documentación de Sass: http://sasslang.com/documentation/file.SASS_REFERENCE.html#_5
- Sass: http://sass-lang.com/ • Ruby: https://www.ruby-lang.org/en/ • Sass Wikipedia: https://en.wikipedia.org/wiki/Sass_(stylesheet_language)
- Less wikipedia: https://en.wikipedia.org/wiki/Less_(stylesheet_language)
- Stylus wikipedia: https://en.wikipedia.org/wiki/Stylus_(stylesheet_language)
- compass: http://compass.kkbox.com/
- scout: http://mhs.github.io/scout-app/
- Koala: http://koala-app.com/