**SASS. CSS Preprocessor**

En una de nuestras últimas sesiones de formación, hicimos una introducción a los preprocesadores CSS y, con un poco más de detalle a **SASS** (Syntactically Awesome StyleSheet), a continuación os dejo la presentación y un poco de explicación de la misma:

[**Pre-procesadores CSS. SASS**](http://www.slideshare.net/MarionaCruz/preprocesadores-css-sass-15457152)from **[MarionaCruz](http://www.slideshare.net/MarionaCruz" \t "_blank)**

**Qué es un preprocesador css?** Un preprocesador es un programa que procesa los datos de entrada para producir una salida que se utiliza como entrada a otro programa, por tanto, un preprocesador CSS procesa el código que le entra, escrito en la sintaxis específica del mismo preprocesador y lo convierte a CSS para que sea interpretado por el navegador.

**Ventajas de usar un preprocesador:** Nos ayudará a escribir código css más rápido, organizado y fácil de mantener, para poder asegurar un buen crecimiento de nuestro site, ya que nos permite:

* **Organizar mejor el código**, por ejemplo, anidando selectores.

**.SCSS**  
a {  
color: #ce4dd6;  
&:hover { color: #ffb3ff; }  
&:visited { color: #c458cb; }  
}

**.CSS**  
a { color: #ce4dd6; }  
a:hover { color: #ffb3ff; }  
a:visited { color: #c458cb; }

* **Usar variables**

**.SCSS**  
$blue: #3bbfce;  
.content-navigation { border-color: $blue; }  
.box { background-color: $blue; }

**.CSS**  
.content-navigation { border-color: #3bbfce; }  
.box { background-color: #3bbfce; }

Las variables pueden ser tanto locales como de entorno, una variable definida dentro de una clase no será accesible fuera de esa clase o estilo.

* **Reutilizar código**. Para ello usaremos ‘mixins’, clases definidas en el documento que podemos llamar desde cualquier otra clase o desde otro mixin. A estos mixins les podremos pasar parámetros, para adaptar el resultado deseado en cada caso.

**.SCSS**  
@mixin rounded-top {  
$side: top;  
$radius: 10px;  
border-#{$side}-radius: $radius;  
-moz-border-radius-#{$side}: $radius;  
-webkit-border-#{$side}-radius: $radius;  
}  
#navbar li { @include rounded-top; }  
#footer { @include rounded-top; }

**.CSS**  
#navbar li {  
border-top-radius: 10px;  
-moz-border-radius-top: 10px;  
-webkit-border-top-radius: 10px;  
}  
#footer {  
border-top-radius: 10px;  
-moz-border-radius-top: 10px;  
-webkit-border-top-radius: 10px;  
}

* **Hacer operaciones**, de esta manera las medidas siempre serán relativas a un tamaño fijado, y solo tendremos que cambiar ese valor y se recalculará el tamaño de los demás items.

$navbar-width: 800px;  
$items: 5;  
li {  
float: left;  
width: $navbar-width/$items - 10px;  
}

* **Usar funciones.**Los preprocesadores tienen una serie de funciones nativas, que llevan a cabo acciones con las que nos podemos encontrar en el día a día y, además, podremos crear las nuestras propias.

**Ejemplos de funciones nativas:**  
darken (red, 5); //Oscurece un 5% al rojo  
lighten (red, 10); //Oscurece un 10% al rojo  
round($pi); //Redondea el número $pi

**Ejemplo de función propia:**  
@function golden-radio($int) {  
//Devuelvo el número áureo  
@return floor($int / 1.62);  
}  
$width: 900px;  
.container {  
width: $width;  
.left {  
width: golden-radio($width);  
}  
.right {  
width: ($width - golden-radio($width));  
}  
}

* **Ocultar nuestros comentarios de código.** Los comentarios en CSS son visibles para cualquier usuario que consulte el archivo descargado en el navegador, pero los preprocesadores nos permiten insertar comentarios, algo que para nosotros es muy útil a la hora de trabajar, y que estos no sean visibles en el fichero .css final.
* **Incluir archivos.**Gracias a esto, podemos tener las clases más comunes en un fichero e incluirlo siempre que nos sea necesario, sin tener que picar el código cada vez.

@import "rounded";

**Cosas a tener en cuenta** cuando vayamos a tomar la decisión de usar un preprocesador o no…

* No es para principiantes, no por la falta de conocimientos, sino porque es recomendable que alguien que va a trabajar con CSS conozca bien el código puro, los atributos, la sintaxis, etc, ya que será la mejor manera de sacarle un buen rendimiento al preprocesador.
* Definir una estrategia. Si se trabaja con un equipo de frontend developers es importante definir una guía de estilo, unas reglas de programación… sin esto es muy probable que se acabe duplicando código y no estaremos aprovechando la herramienta al máximo.
* CSS más enfocados a programación, paso de parámetros, uso de variables… eso puede hacer que la curva de aprendizaje crezca un poco.
* Rendimiento. Como siempre antes de tomar cualquier decisión sobre qué herramientas de desarrollo usar y cuales no, hay que hacer pruebas de rendimiento, ver si realmente lo nuevo nos aportará ventajas o no.

**SASS y LESS** son dos de los preprocesadores más conocidos, no entraremos en detalle en los dos, solo comentar algunas de las diferencias más relevantes:  
**SASS**: corre sobre Ruby y se procesa en el servidor.  
LESS: es una librería de JS, se procesa en cliente, aunque se puede hacer en el servidor y servirle al cliente los css finales ya procesados y, además, es el más popular entre los developers, por lo que es más fácil encontrar información y resolver nuestras dudas.  
En cuanto a la curva de aprendizaje estarían igualados, ya que para los dos hay que aprender una nueva sintaxis.

**Algunos datos sobre SASS**, fue creado por Hampton Catlin en 2007, como ya hemos comentado corre sobre Ruby, y acepta dos sintaxis, .SCSS y .**SASS**, la primera es igual que el .CSS original, y la segunda en lugar sustituye las llaves por identación en las líneas.

**Ejemplos**  
Podéis consultar los ejemplos de código en la presentación, a partir de la décima diapositiva, veréis ejemplos de lo que hemos comentado anteriormente, uso de variables, de mixins, creación de funciones, herencia…

**Para terminar…** en la presentación os dejo las sentencias para la instalación de **SASS** y para la compilación de ficheros, y una serie de links para que podáis ampliar la información.