[DNN Spain]
– Guía de estilos -

Guía para la codificación de las hojas de estilo

FOREIGN EXCHANGE SERVICES

Departamento: Tecnología

Responsable: Juan José Alonso Sánchez

Fecha: 03/08/2022

ÍNDICE

1.- INTRODUCCIÓN

Lo importante al escribir una hoja de estilo es seguir siempre los mismos criterios, para que sea más fácil detectar errores o poder reutilizarla en otros proyectos.

En los proyectos de Global Exchange se prevé que varios equipos de trabajo se encarguen de la edición y mantenimiento de sus aplicaciones, por lo que se plantea la siguiente guía de estilo para un óptimo desarrollo.

2.- OBJETIVO

Esta guía servirá de manual para implementar un diseño uniforme en sus aplicaciones y una codificación más limpia.

3.- ¿QUÉ ES?

Es un manual de referencia para poder elaborar un diseño uniforme de las aplicaciones de Global Exchange.

Esta guía resulta necesaria para poder presentar el trabajo realizado por los distintos equipos de desarrollo como una única unidad y ofrece directrices acerca de cómo se tiene que codificar para poder ofrecer un desarrollo mantenible.

4.- PRINCIPIOS

- Mantener un código limpio y simple y poder reusarlo lo máximo posible
- Implementar un código que luzca como si lo hubiera escrito una única persona
- Codificar para obtener una mayor escalabilidad y visibilidad

5.- ARQUITECTURA

La arquitectura usada para organizar las carpetas del código fuente de los estilos CSS es **La regla del 7+1**. Esta arquitectura también es conocida como **patrón 7–1**. Consiste en tener todas nuestras hojas de estilo organizadas en 7 carpetas y un único archivo en la raíz para importarlas.

A continuación, se muestra la estructura jerárquica:

```
scss/                       //Estilos a procesar
|- style.scss               //Archivo fuente
|
|- base/                    //Código base
|   |- _global.scss
|   |- _reset.scss
|   |- _tipografia.scss
|
|- components/              //Estilos componentes
|   |- _buttons.scss
|   |- _texts.scss
|
|- layouts/                 //Estilos estructura
|   |- _footer.scss
|   |- _header.scss
|
|- themes/                  //Estilos diferentes temas
|   |- _admin.scss
|   |- _theme.scss
|
|- utils/                   //Código funcionalidad
|   |- _extends.scss
|   |- _funciones.scss
|   |- _mixins.scss
|   |- _variables.scss
|
|- views/                   //Estilos vistas
|   |- _home.scss
|
|- vendors/                 //Librerías de terceros
```

- **scss**: Este directorio contiene el código fuente de todos los estilos de aplicación agrupados en partials1, para que posteriormente sean mapeados y procesados a una única hoja de estilo interpretada por los navegadores.
- **base**: Contiene el código base del proyecto como las reglas de tipografía o los resets necesarios para sobrescribir las reglas por defecto de los navegadores.
- **components**: Contiene los estilos de los distintos componentes que emplearemos dentro de nuestra aplicación como pueden ser botones, sliders, carruseles… Separar cada componente en su propio archivo nos permitirá reutilizarlos en otros proyectos.
- **layouts**: En esta carpeta aparecen los estilos relacionados con la estructura de la aplicación, como los estilos de la cabecera o el pie de página.

[1] Una de las múltiples ventajas que nos proporciona Sass es poder dividir nuestras hojas de estilo en pequeños archivos (denominados *partials)* que posteriormente podemos importar en nuestras hojas de estilo principales.

- **themes**: Almacena los estilos referidos a diferentes themes que pueda adoptar nuestro proyecto en función, por ejemplo, del tipo de usuario o la sección que esté visualizándose.
- **utils**: En esta carpeta se encuentran las funciones, variables y mixins empleados en el resto de ficheros. Estos archivos no generan ninguna regla CSS al ser compilados, sino que añaden funcionalidad extra para ser empleada por los otros partials.
- **views**: Contiene los estilos específicos para determinadas vistas (páginas) de la aplicación como puede ser la página de inicio o la de login.
- **vendors**: Contiene librerías de terceros.

Cabe destacar que el archivo **style.scss** situado en la raíz de la carpeta de desarrollo, importará todos los estilos (partials) alojados en dicha carpeta.

6.- METODOLOGÍA

La metodología usada para la codificación de las hojas de estilo se llama BEM. Esta metodología divide la interfaz de usuario en bloques independientes para crear componentes escalables y reutilizables.

Con BEM modularizamos lo máximo posible cada uno de los bloques de código dispuesto. Se centra en tres parámetros o variables posibles:

- **Bloques**

```
<section>
   <article class=”noticia”>
        <! -- Bloque contenedor -- >
   </article>
</section>
```

- **Elementos**

```
<section>
   <article class=”noticia”>
        <h1 class=”noticia__titulo”>Título de la noticia</h1>
   </article>
</section>
```

- **Modificadores**

```
<section>
   <article class=”noticia”>
        <h1 class=”noticia__titulo--mayuscula”>Título</h1>
   </article>
</section>
```

7.- REGLAS GENERALES

- Los nombres de las hojas de estilo que vayan dentro de las carpetas components, layouts y utils irán en plural y siempre deben comenzar con el símbolo _. El resto de hojas de estilo se nombrarán en singular (*Ejemplo: _botones.scss*).
- Las imágenes se nombran en referencia a su bloque y están separadas por un guión (*Ejemplo: background-header.png*)
- Las clases deben ir en singular y minúsculas (*Ejemplo: .galeria__boton*).
- El estilo de escritura es Kebab Case (*Ejemplo: .main-header*).

8.- SINTAXIS

- Uso siempre de palabras en minúscula.
- Uso de comillas dobles preferiblemente en lugar de comillas simples.
- El selector y la llave de apertura deben ir en la misma línea, sin sangrar.
- 1 espacio en blanco entre el selector y la llave.
- Cada propiedad debe ir en una línea, completando la línea con un punto y coma.
- 1 tabulación para identación.
- 0 espacios antes de los dos puntos que separan la propiedad y su valor.
- 1 espacio después de los dos puntos que definen de la propiedad.
- 0 espacios en blanco antes del punto y coma final de cada línea de propiedad.

- La llave de cierre debe ir en una línea, sin sangrar. En esa línea sólo estará la llave de cierre.
- Bloques de CSS separados por una línea en blanco (2 saltos de línea).
- Selectores dentro del mismo bloque separados por 1 salto de línea.
- Última propiedad termina siempre con punto y coma.
- Evitar abuso de anidaciones. Límite 1 nivel.
- Evitar el uso de identificadores (*id/#*) salvo en especificaciones muy concretas como el uso de anclas.
- Siempre que se pueda usar clases (*class/.*).
- Uso de mixins para tamaño, estilos y media queries.
- Uso de funciones para cálculos (*valores numéricos o el tamaño de fuente*).
- 1 espacio entre parámetros en mixins y funciones.
- 1 espacio antes del selector de parámetros () en mixins y funciones.
- Uso de propiedades cortas cuando sea posible. *Ejemplo border: 1px solid #000*
- Se deben omitir los 0 cuando sea posible. *Ejemplo Escribir .8 en lugar de 0.8*
- Uso de 3 caracteres haxadecimales cuando sea posible en lugar de 6. *Ejemplo #000*

9.- PROPIEDADES

Escribir las sentencias en el orden en que aparecen los elementos en la página web. Además, las propiedades y selectores deberán ordenarse de la siguiente manera:

- Propiedades del modelo de caja (*display, width, height, margin, padding, border*…).
- Posicionamiento (*position, left, top*…).
- Tipografía (*text-transform, font-size*…).
- Decoración (*color, background*…).
- Variables.
- Extends.
- Mixins.
- Mixins con @content

```
 .button {
    display: block;
    width: 220px;
    height: 40px;
    position: relative;
    text-transform: uppercase;
    background-color: #fff;
    font-weight: $semibold;
    @include font-size (13px);
}
```