#Sass

SASS (Syntactically Awesome Stylesheets) es un preprocesador de **CSS** que ermite escribir estilos de forma un poco diferente a **CSS** y que luego son procesados y convertidos de modo tal que el navegador los pueda interpretar como **CSS**. También permite hacer otras cosas que con la sintaxis de **CSS** puro son inposibles de hacer.

Instalación

La instalación la vamos a hacer a través de otra aplicación, para ello primero vamos a descargar e instalar Node.js.
Una vez instalado, abrimos la consola de **Git Bash** e introducimos el siguiente comando:

```
npm install -g sass
```

Ya descargado e instalado, verificamos que efectivamente se haya instalado de manera correcta con el comando:

```
sass --version
```

Y deberíamos ver qué versión tenemos instalada en nuestro sistema.

Compilando archivos

Para que SASS convierta automáticamente los los archivos con extensión `.scss` a archivos `.css`, debemos indicarle cuáles son los archivos a los que debe prestar atención y dónde queremos que los guarde.
Existen dos maneras de hacer esto:

- Dándoles la instrucción a Sass de que *vigile* un archivo en particular:

```
sass --watch styles.scss styles.css
```

Donde `styles.scss` es el archivo **SASS** y `styles.css` es el archivo que va a compilar nuestro código. Este comando los ejecutamos desde **Git Bash** o desde la consola de Visual Studio, siempre teniendo cuidado de estar ubicadas en el directorio de nuestro proyecto.
Los nombres de los archivos son rutas relativas, por lo tanto debemos tener en cuenta la estructura de nuestro directorio, es decir, la carpeta en la cual estamos posicionadas al momento de ejecutar el comando.
Suponiendo que el archivo `styles.css` está ubicado dentro de una capeta llamada css, el comando será el siguiente:

```
sass --watch styles.scss css/styles.css
```

- Dándole la instrucción a Sass de que *vigile* carpetas directamente.
  En este caso, va a buscar todos los archivos no-parciales con extensión `.scss` que se encuentren ubicados en la carpeta de origen y los convertirá a sus equivalentes archivos `.css` en la carpeta de destino:

```
sass --watch origen:destino
```

Donde `origen` y `destino` son los nombres de las carpetas respectivas
Suponiendo que queremos vigilar el directorio `styles` y compilar los archivos en la carpeta `styles/css`, el comando será el siguiente:

```
sass --watch styles:styles/css
```

Variables

Una de las características de Sass es la implementación de variables. Las viariables son etiquetas a las cuales se les asigna un valor, por ejemplo una tipografía, y cuando queramos modificar elemento, en lugar de utilizar directamente esta fuente, podemos asignar la variable como valor

de la propiedad. Cada elemento que utilice esta variable luego es convertido a css utilizando el valor que ésta contiene, de este modo, si luego necesitamos cambiar la tipografía, simplemente cambiamos el valor de la variable, y todos los elementos que la estén utilizando se actualizarán automáticamente.

Una variable se declara de la siguiente manera:

```
$nombre-variable: valor;
```

Por ejemplo:

```
$main-font: "Roboto";
```

Y se utiliza simplemente llamándola con su nombre, anteponiendo el signo *$*:

Sass CSS

```
h1{
    font-family: $main-font;
}
```

Lo destacable de las variables es que al tener un nombre le dan un valor más semántico a nuestros estilos, es decir, es más fácil entender qué es lo que estamos estilando porque los valores tienen cierto significado más allá de sus valores numéricos o de texto.
Dentro de una variable podemos guardar cualquier tipo de valores que usamos dentro de propiedades de **css**, por ejemplo:

```
$main-font-size: 15px;
$button-border: 2px solid
$main-color: #bc3b69;
$column-padding: 20px 30px;
$background-transition: transition 1s ease-in 0.1s;
```

Selectores anidados

Selectores anidados

Otra de las características de **Sass** es que nos permite anidar selectores. Esto nos da una visualización más clara y evidente de la jerarquía de los elementos.
Para anidad un selector dentro de otro, simplemente debemos declararlo dentro de las llaves de éste:

Sass CSS

```
main {
    width: 600px;

    article{
        width: 300px;
    }
}
```

Podemos anidar todos los selectores que queramos dentro de otro selector, por ejemplo:

Sass CSS

```
main{
    width: 600px;

    h1{
        font-family: $main-font;
    }

    h2{
        font-family: $secondry-font;
    }

    article{
        width: 300px;
    }
}
```

Podemos anidar selectores dentro de selectores todos los niveles que queramos, si bien esto es perfectamente válido, no es muy conveniente anidar más de 3 o 4 niveles máximo, ya que de otra forma nos queda una jerarquía muy rígida, y si modificamos el orden en nuestro **html**, los estilos se van a romper ya que la jerarquía no aplica más.
En el siguiente bloque de código vemos un ejemplo de como la estructura de Sass sigue la estructura del html:

HTML Sass

```
<main>
    <section>
        <div class="column">
            <div class="news"></div>
        </div>
    </section>
</main>
```

Si ahora quisiéramos utilizar `.news` en otra sección, los estilos que declaramos para `.news` no se van a aplicar a los que se encuentren dentro de `secondary-news`, ya que en la jerarquía que declaramos en nuestro archivo sass, necesitamos que estén dentro de un elemento `.column` para que aplique. Por eso, en casos como este, lo mejor es declarar una jerarquía más flexible.
Por ejemplo:

HTML Sass

```
<main>
    <section>
        <div class="column">
            <div class="news"></div>
        </div>
    </section>

    <section class="secondary-news">
        <div class="news"></div>
        <div class="news"></div>
        <div class="news"></div>
    </section>
```

```
</main>
```

Si usamos el selector **&** anidado, lo que hacemos es referenciar al padre. Esto sirve para para utilizar en conjunto con pseudo-selectores, por ejemplo:

Sass CSS

```
h1{
    color: grey;

    &:hover{
        color: red;
    }
}
```

Media Queries

Podemos anidar **media queries** directamente dentro de selectores, sin necesidad de estar referenciándolos nuevamente.

Sass CSS

```
h1 {
    font-size: 32px;

    @media (max-width: 350px) {
        font-size: 28px;
    }
}
```

Archivos Parciales

Con **CSS** si queremos separar nuestro código en diferentes archivos para tenerlo mejor organizado y clasificado, tenemos que luego vincular cada uno de estos archivos en **HTML**, lo

cual no es muy recomendable, ya que aumenta los tiempos de carga de nuestra pagina. **Sass** soluciona ésto permitiéndonos crear **archivos parciales**.
Los archivos parciales **no** se transforman en archivos `.css` directamente, sino que primero deben ser importados en un archivo sass que no sea parcial, y compilar este último.
Al importar un archivo parcial, lo que se hace es inluir su código dentro del archivos al cual lo estamos importando.
Para declarar un archivo parcial, basta con anteponerle un guión bajo al nombre del archivo:

```
_variables.scss
```

Luego para importar este archivo e incluir su código, se utiliza la siguiente instrucción dentro del archivo al cual lo vamos a importar:

```
@import "variables";
```

- **Detalle importante**: cuando importamos un archivo parcial, colocamos solo el nombre, sin guión bajo ni extensión.

La instrucción `@import`, no toma el nombre de un archivo, sino la ruta relativa al mismo, por lo tanto, si el archivo a importar estuviera dentro de una carpeta `general`, la manera de hacerlo será la siguiente:

```
@import "general/variables";
```

Organización

La posibilidad de separar nuestro código en distintos archivos permite organizarlo y distribuirlo de forman que sea más ordenada y que facilite la tarea a la hora de encontrar y modificar estilos particulares. Una **posible** (no la única) manera de hacerlo es la siguiente:

```
📁mi proyecto
    📁fonts
    📁images
    📁html
        about.html
```

```
        contact.html
    styles
        general
            _fonts.scss
            _normalize.scss
            _variables.scss
        components
            _buttons.scss
            _inputs.scss
            _text.scss
        css
            main.css
        layout
            _aside.scss
            _footer.scss
            _header.scss
            _nav.scss
        pages
            _about.scss
            _contact.scss
            _home.scss
        main.scss
    index.html
```

La estructura anterior cuenta con un único archivo sass completo que será compilado a css; `main.scss`, el resto de los archivos son parciales. En este archivo `main.scss` no se escribe nada de código, solo se importan los demás archivos:

```
@import 'general/variables';
@import 'general/fonts';
@import 'general/normalize';

@import 'components/buttons';
...
```

- Dentro de `general` van las declaraciones más *generales* y que se utilizan en todos los estilos, por ejemplo, las variables y el archivo de normalización.
- En `componentes` van aquellos elementos que se repiten en distintas páginas y que comparten ciertos estilos (botones, inputs, cuadros de diálogo, etc). La idea es separar archivos parciales por tipo, y que solo refiramos a ellos a elementos o comjuntos de

elementos html muy chiquitos o indivisibles, es decir, que ya no pueden ser subdivididos y van siempre juntos.

- En `layout` van las secciones más generales de las páginas que se **repiten** en varias o todas ellas. Los ejemplos más usuales son el *header*, *footer*, *nav* y los incluimos dentro de un archivo parcial a cada uno.
- Dentro de `pages` va un archivo parcial por cada página, donde se incluyen los estilos **particulares** de cada una de ellas, es decir, aquellos estilos **específicos** que son propios de ellas. Por ejemplo, un botón puede repetir sus estilos en múltiples páginas, pero la ubicación de dicho botón es propio de cada página en particular (no está siempre ubicado en el mismo lugar).
- Finalmente, dentro `css` irá el archivo `main.css` que es la compilación de todos los archivos sass, y que será el que se vincula al archivo html.

Podemos aprovechar la posibilidad de anidar selectores en Sass para declarar los estilos que pertenecen a cada elemento, sección o página y que no afecten al resto de los elementos. Por ejemplo, dentro de `_footer.scss` incluiríamos algo así como:

```
footer{
    ul{
        li{

        }
    }
}
```

De modo tal que si tenemos algún `ul li` en alguna otra sección de nuestras páginas, no sean afectados por los estilos del footer. Es decir, incluimos todos los estilos del footer dentro de un selector, de modo que dichos estilos solo se apliquen a los que pertenecen a dicho selector.
Lo mismo podemos hacer con las páginas y layouts. Por ejemplo, suponiendo que los elementos `h2` de la página principal tienen un tamaño de letra diferente al de los demás elementos `h2` de las restantes páginas, entonces lo que podemos hacer es:

```
#home{
    h2{
        font-size: 24px;
    }
}
```

De esta forma estamos apuntando a los elementos `h2` dentro de un elemento `#home`, y el resto de los elementos `h2` no se verán afectados. Para que esto funcione, en html habrá que incluir a todos los elementos de la pagina *home* o *index* dentro de un contenedor con id *home*.

Edit this page

Previous
« Flexbox

Next
Variables y Tipos de Datos »

Linkedin

Instagram

Facebook