BEM

Ahora que hemos terminado la base de nuestro proyecto es momento de comenzar a crear la estructura HTML de la maqueta, pero antes de hacerlo conoceremos un concepto qu nos ayudará a comprender a cómo nombrar nuestras clases de CSS de manera más clara y consistente.

¿Qué son las clases semánticas?

La tarea de crear clases es, aunque parezca aburrida, una práctica que debería ser tan importante como crear los estilos de la interfaz, ya que el hacerlo de una manera correcta nos ayudará a hacer más fácil la tarea de encontrar un elemento usando CSS o eventualmente el DOM en JavaScript.

Nicolás Gallagher, creador de Normalize.css menciona en un artículo acerca de este tema una interesante definición acerca de las clases semánticas:

> Se recomienda a los autores que utilicen valores de [atributo de clase] que describan la naturaleza del contenido, en lugar de valores que describan la presentación deseada del contenido.

En otras palabras, nosotros como maquetadores deberemos escribir clases que describan qué es objetivamente el contenido y no lo que pensamos qué es.

Otro punto interesante de este artículo de Gallagher las recomendaciones que menciona acerca de las clases.

A saber:

- **La semántica de los contenidos ya se encuentra dada por la naturaleza de los elementos HTML o por otro atributos**, por lo tanto podremos aprovecharnos de esos elementos para crear nuestras clases.
- **Las clases nos entregan información relevante para los desarrolladores y no así para las maquinas**, de modo que estas deberán ser legibles para humanos.
- **El proposito de las clases es ser el ancla de CSS y Javascript**. Crear clases las cuáles semánticas es clave para el correcto uso de estos selectores en CSS y JS.
- **Las clases comunican infomación útil a desarrolladores.** Escribir de buena manera clases hará que nuestros código sea escalable y fácil de entender por parte de otros desarrolladores.

Teniendo en cuenta los puntos vistos anteriormente, la forma en la que podremos llevar a la práctica la semántica de clases es creando nuestro propio sistema de clases o utilizar alguna metodología para nombrar clases.

La primera opción es algo útil si somos organizados y sabemos como presentar este sistema a otros desarrolladores, por que como sabemos nosotros no trabajaremos solos. Si por el contrarío deseamos usar un sistema probado y ampliamente utilizado nos podremos decantar por el uso de metodologías de arquitectura CSS.

¿Qué son las metodologías de Arquitectura CSS?

Estás metodologías son unas guías de estilo que nos ayudarán a organizar y estructurar nuestro CSS de manera que sea fácil escalarlos y mantenerlos por otros desarrolladores.

Las metodologías que podremos usar para esta tarea son:

- **OOCSS**
  - Esta es la primera metodología creada para este fin el cual separa la estructura y el contenido de la interfaz en objetos CSS que pueden ser modificados de manera independiente.
- **SMACSS**
  - Es una guía de estilos con la que podremos escribir código CSS basados en categorías.
- **Atomic Design**
  - Es una metodología que divide los elementos de una interfaz en atomos, que luego pueden ser unidos en moléculas y organismos.
- **SUITCSS**
  - Es una guía de estilo que presenta las mejores prácticas a usar en la creación de clases y estructura CSS.
- **BEM**
  - Es una metodología que basada en componentes que nos ayudará a separar los elementos que constituyen nuestra interfaz en bloques independientes.

Nosotros, dentro de esta gran cantidad de opciones usaremos BEM para construir nuestras clases en CSS.

¿Por qué elegir BEM?

Utilizaremos esta metodología por sobre otras por tres razones:

La primera tiene que ver con la propuesta que ofrece para escribir clases, que evitan la repetición de estilos y previenen conflictos asociados a la cascada CSS.

La segunda razón es ralacionada a su popularidad y aceptación por parte de los desarrolladores. El implementarlo en nuestros proyectos nos dará un marco de trabajo que será entendido no sólo por nosotros, sino que también por otros desarrolladores que revisen nuestro código.

La última tiene que ver con la facilidad de aprender esta metodología, de modo que la curva de aprendizaje que conlleva sus bases será baja.

Teniendo clara las razones de su uso es momento de conocer las bases de BEM.

¿Qué es BEM?

Bem significa bloque, elemento y modificador y como sabemos es una metodologia que nos ayudará a separar los elementos de nuestra interfaz en bloques.

Para entender como funciona BEM imaginemos que nuestra interfaz tiene un header que contiene en su interior una barra de navegación.

```
<div> <!-- Header -->

  <div><!-- Navegación -->
    <ul>
      <li>Inicio</li>
      <li>Acerca de</li>
      <li>Blog</li>
    </ul>
  </div><!-- fin navegación -->

</div> <!-- fin header -->
```

El `<div>` que representa a header cumple una función específica dentro de una interfaz la cual es contener todos los elementos de la cabecera del sitio. El `<div>` que contiene el listado desordenado cumple otra que es contener los elementos de la navegación.

Estos contenedores dentro de BEM son parte de una estructura conocida como bloques.

Conociendo BEM

Bloques:

Los bloques son entidades o funcionalidades independientes de un componente. Estos en HTML son representados con el atributo de clase.

Algo importante a saber sobre estos bloques es que ellos deben describir un propósito, o sea, identificar que es el elemento, asimismo este no debe ser confundido con un estado del elemento, o sea si este grande, pequeño, o tiene un color en especifico.

En el ejemplo anterior, la mejor forma de nombrar a estos bloques es identificando el tipo de elemento que son.

Como vimos el primer bloque es la cabecera del sitio y su objetivo es contener a los elementos que componen a un header, por lo tanto el nombre de clase más lógico para este bloque es `class="header"`.

El segundo bloque es la navegación y cumple con el objetivo de contener los elementos que componen a una barra de navegación. En este caso el nombre clase más adecuado para este bloque es `class="navigation"`.

Algo importante a saber sobre los bloques es que estos pueden ser anidados unos con otros, de modo que podremos integrarlos sin problemas la cantidad veces que creamos necesaria.

Como resultado, la cabecera y la navegación integrado con la nomenclatura de BEM se verá de la siguiente manera:

```
<div class="header">

  <div class="navigation">
    <ul>
      <li>Inicio</li>
      <li>Acerca de</li>
      <li>Blog</li>
    </ul>
  </div>

</div>
```

Elementos:

Ahora que ya identificamos los bloques que contienen a los elementos, veamos con mayor detalle el bloque de navegación.

El bloque de navegación contiene en su interior una lista desordenada, que a su vez tiene tres ítems que componen la estructura de una barra de navegación. Estos elementos que se encuentran al interior del bloque `.navigation` forman parte de los elementos del bloque.

Los elementos son entidades que están ligadas a un bloque y que usadas por si sola no tendrán una connotación semántica. Esto quiere decir que si usamos de forma independiente estos elementos perderán su contexto semántico y por ende no formarán parte del bloque.

Para reconocerlos dentro de nuestra interfaz deberemos identificar el objetivo del bloque y no su estado. Además, para identificarlos como elementos de un bloque deberemos nombrar a la clase con el `nombre-bloque__nombre-elemento` .

En el contexto de nuestro ejemplo, los elementos dentro del bloque tendrán la siguiente nomencaltura:

```
<div class="header">

  <div class="navigation">
    <ul class="navigation__list">
      <li class"navigation__item">Inicio</li>
      <li class"navigation__item">Acerca de</li>
      <li class"navigation__item">Blog</li>
    </ul>
  </div>

</div>
```

Algo importante a saber es que los elementos que encuentren anidados dentro de otros elementos deberán mantener la nomenclatura `nombre-bloque__nombre-elemento` y no `nombre-bloque__nombre-elemento__nombre-elemento` .

Modificador:

El último elemento a tener en cuenta sobre este BEM son lo modificadores los cuales cumplen una función importante ya que definen la aperiencia, estado o comportamiento de un elemento o bloque.

La sintaxis usada para estos elementos es:

Para bloques:

```
<div class"footer footer_dark-theme">
  <small class"footer__copyright">2018. Todos los derechos reservados</small>
</div>
```

Para elementos:

```
<form class="form">
  <div class="form__container">
    <input type="text" class="form__input form__input_focused">
  </div>
</form>
```

Revisemos un ejemplo:

Imaginemos que tenemos la siguiente estructura HTML con una implementación de BEM:

```
<form class="form">
  <div class="form__container">
    <input type="text" class="form__input">
    <input type="submit" value="Enviar" class="form__button">
  </div>
</form>
```

Si quisieramos cambiar el tamaño del botón por defecto por uno más pequeño deberiamos, en primer lugar, identificar donde se encuentra el elementos, que en este caso sería el `<input>` del tipo `type="submit"`. Aquí deberiamos agregar este modificador de la siguiente manera:

```
  <input type="submit" value="Enviar" class="form__button nombre-
bloque__nombre-elemento-clave_valor">
```

En este contexto, como deseamos que el tamaño del bloque sea de tamaño pequeño la implementación de este nuevo tamaño podría ser:

```
  <input type="submit" value="Enviar" class="form__button form__button_size-s">
```

Como vemos implementar BEM en nuestros proyectos es fácil y no requiere de mayor aprendizaje del que hemos visto hasta el momento.