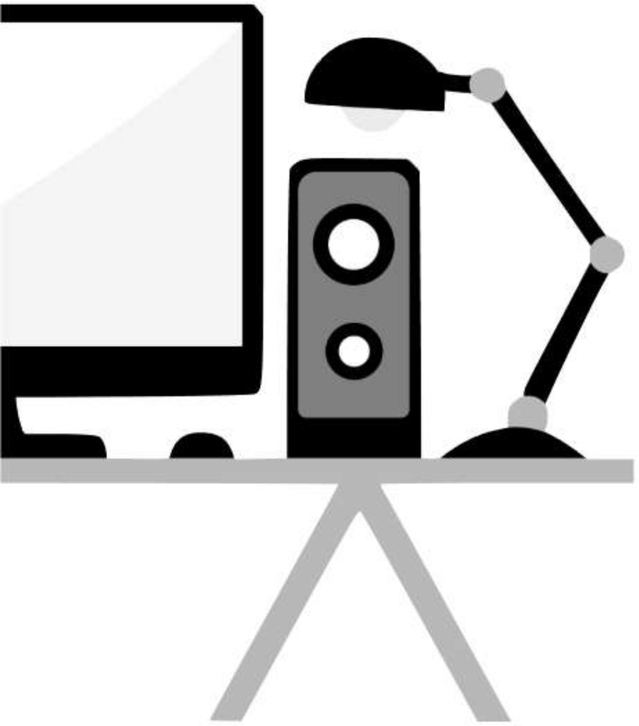
DESARROLLO DE LA INTERFAZ DE USUARIO WEB

Lección 02

sustantiva
CON•SENTIDO

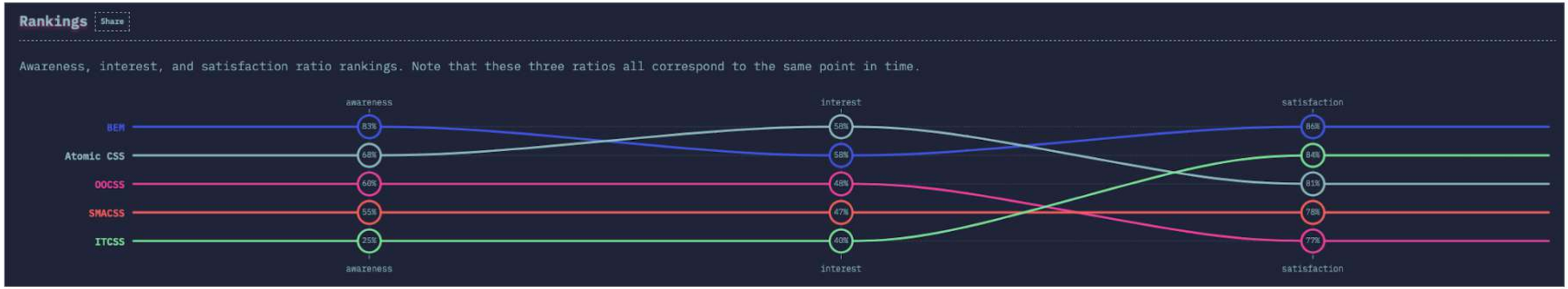
Contenidos



1. Metodologías de organización y modularización de estilos
2. Metodología BEM
3. En qué consiste esta metodología
4. Conceptos claves de BEM
5. Convención de nombres
6. Cómo estructurar un proyecto con BEM
7. Metodología OOCSS
8. En qué consiste esta metodología
9. Principios básicos de OOCSS
10. Metodología SMACCS
11. En qué consiste esta metodología
12. Categorías de elementos SMACCS

sustantiva
CON•SENTIDO

Metodologías de organización y modularización de estilos



En State of CSS 2019 ya se podia ver esta perspectiva de los usuarios respect a BEM, OOCSS, SMACSS y otras convenciones.

Awareness, es que tanto la conocen, interest indica que entre quienes lo conocen, si desean o no probarlo y satisfaction, que mide de entre quienes lo han probado, que tanto les ha gustado y si desean seguir usándolo.

sustantiva
CON•SENTIDO

Metodología BEM

Cómo funciona la Metodología BEM CSS

En BEM, definiremos todos nuestros estilos utilizando **solo clases** y pseudo-selectores. Queda **prohibido usar *#identificadores* o nombres de *etiquetas* HTML** en nuestro CSS.

Además, todos los nombres de las clases van a seguir una misma estructura:

.bloque__elemento—modificador

A continuación veremos qué significa eso de *bloque, elemento y modificador*, y cómo vamos a nombrar nuestras clases.

sustantiva
CON•SENTIDO

Metodología BEM

1.- Bloques: las partes independientes en BEM

En **BEM**, llamaremos bloque a un pedazo de nuestra página web que **tiene sentido por sí mismo**.

Para declarar los estilos de los bloques utilizaremos sencillamente una clase con el nombre del bloque, por ejemplo: *.cabecera*

A veces, los bloques pueden tener nombres compuestos por varias palabras. En este caso usaremos guiones en nuestra clase, por ejemplo: *.mi-bloque-compuesto*

Veamos algunos ejemplos típicos de bloques:

Metodología BEM

Un botón

Este es el ejemplo más típico de bloque.

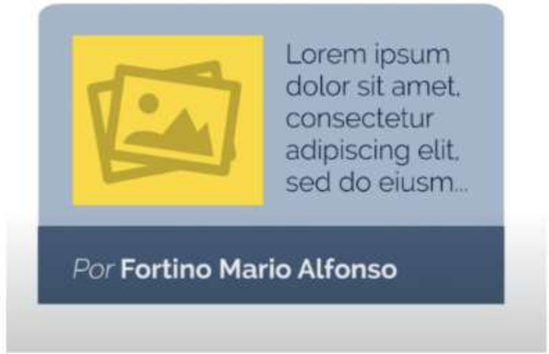
Un botón tiene sentido por sí mismo y podría reutilizarse en otras pantallas:

Con BEM, podríamos darle estilo en CSS con la clase *.button* y usarlo en HTML como:
<button class="button">Twitter</button>

Metodología BEM

Un post de una red social

Los bloques no tienen por qué ser siempre tan sencillos como un botón, también pueden existir bloque más complejos, como por ejemplo los posts (publicaciones) que aparecen en los feeds de la redes sociales.



Si te paras a pensarlo los posts son también trozos de página que se reutilizan y que tienen sentido por sí mismos.

Podríamos dar estilo a este bloque con la clase *.post* y usarlo en HTML como:

<article class="post"> … </article>

Metodología BEM

2.- Elementos BEM: las distintas partes de un bloque

Un elemento es una parte de un bloque. De esta manera un bloque podrá contener uno o varios elementos.

Para declarar los estilos de los elementos utilizaremos clases con esta estructura:
.bloque__elemento

Como ves el nombre de la clase combina el nombre del elemento con el nombre del bloque al que pertenece.

Veamos algunos ejemplos de elementos:

Metodología BEM

Las distintas opciones de un menú

Siguiendo con el ejemplo anterior, habíamos dicho que un menú de navegación podría ser un bloque, que tendría la clase .menu, pero, ¿qué clases usaríamos para referirnos a cada una de sus partes?

Como vemos contiene un logo y seis botones:
Siguiendo la estructura que hemos visto podríamos dar estilo al elemento logo usando la clase *.menu__logo* y a los elementos botones usando la clase *.menu__button*, de modo que el HTML quedaría parecido a esto:

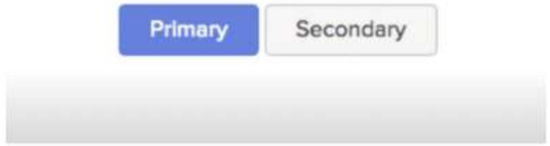
<nav class=”menu”> <svg class=”menu__logo”>…</svg> <button class=”menu__button”>Join</button> <button class=”menu__button”>Log in</button> <button class=”menu__button”>Pricing</button> <button class=”menu__button”>Features</button> <button class=”menu__button”>Inspiration</button> <button class=”menu__button”>Stock</button> </nav>

Metodología BEM

3.- Modificadores BEM: aliña tus bloques y componentes

Los modificadores son clases puedes **añadir** para cambiar la apariencia o el comportamiento de los *bloques* y *elementos*.
Los modificadores se utilizan normalmente para diferenciar las distintas variantes de un bloque o un elemento.
Para modificar un *bloque* usaremos clases con la estructura *.bloque–modificador y* para modificar un *elemento* usaremos *.bloque__elemento–modificador*
Por ejemplo, si tenemos dos botones, uno primario y otro secundario:

Metodología BEM



Usaremos modificadores para diferenciarlos:

```
<button class="button button--primary">
  Primary
</button>
<button class="button button--secondary">
  Secondary
</button>
```

Las clases *modificador* **siempre deben acompañar a una clase *bloque* o una clase *elemento***, no tiene sentido que aparezcan solas:

sustantiva
CON•SENTIDO

Como estructurar un proyecto con BEM

Por lo general un proyecto tiene al menos un nivel de redefiniciones.

No existe un número máximo de niveles.

```
project/
    common.blocks/
    library.blocks/
```

sustantiva
CON•SENTIDO

Como estructurar un proyecto con BEM

Las implementaciones de los bloques van en archivos css separados.

Por ejemplo, si creamos un input usando bem, el css va en un archivo input.css y el javascript en input.js

```
project
    common.blocks/
        input.css
        input.js
```

sustantiva
CON•SENTIDO

Como estructurar un proyecto con BEM

Podemos ver como se agrupan, según el significado, no el tipo y como conviven distintos bloques.

```
project
    common.blocks/
        input/
            input.css
            input.js
        popup/
            popup.css
            popup.js
```

sustantiva
CON•SENTIDO

Como estructurar un proyecto con BEM

Y acá podemos ver un ejemplo de distintos bloques, elementos y modificadores en un proyecto BEM.

```
project
    common.blocks/
        input/
            _type/
                input_type_search.css
            __clear/
                _visible/
                    input__clear_visible.css
                input__clear.css
                input__clear.js
        input.css
        input.js
```

Metodología OOCSS

Como vimos antes, OOCSS es bastante simple en su idea, con dos principios clave y básicos:

1.-SEPARACIÓN DE LA ESTRUCTURA DE LA PIEL O SKIN

2.- SEPARACIÓN DE CONTENEDORES Y CONTENIDO

AHORA, ¿COMO FUNCIONA ESTO EN UN EJEMPLO REAL APLICANDO OOCSS?

sustantiva
CON•SENTIDO

Metodología OOCSS

```css
.header-inside {
  width: 980px;
  height: 260px;
  padding: 20px;
  margin: 0 auto;
  position: relative;
  overflow: hidden;
}
```

Tenemos este código para la cabecera de un sitio. Podemos identificar rápido los elementos propios de la cabecera y a la vez, otros que podemos reutilizar, así que dividimos el código:

Metodología OOCSS

```css
.globalwidth {
  width: 980px;
  margin: 0 auto;
  position: relative;
  padding-left: 20px;
  padding-right: 20px;
  overflow: hidden;
}

.header-inside {
  padding-top: 20px;
  padding-bottom: 20px;
  height: 260px;
}
```

Ahora tenemos a globalwidth la cual tiene:

- Un ancho fijo
- Centrado usando margen: auto
- Posicionamiento relativo para crear un contexto de posicionamiento para los elementos secundarios
- Padding izquierdo y derecho de 20px
- Desbordamiento establecido en "oculto" para la corrección clara

sustantiva
CON•SENTIDO

Metodología OOCSS

Y finalmente implementamos globalwidth en los otros elementos del sitio que se benefician de esa clase:

```
<header>
  <div class="header-inside globalwidth">
  </div>
</header>

<div class="main globalwidth">
</div>

<footer>
  <div class="footer-inside globalwidth">
  </div>
</footer>
```

Metodología SMACSS

En palabras de sus autores, SMACSS se define a si mismo como: una guía de estilo, no un marco rígido. Aquí no hay ninguna biblioteca para que la descargues o instales. No hay ningún repositorio de git para que lo clones. SMACSS es una forma de examinar su proceso de diseño y como una forma de encajar esos marcos rígidos en un proceso de pensamiento flexible. Es un intento de documentar un enfoque coherente para el desarrollo del sitio cuando se utiliza CSS. Y en realidad, ¿quién no está construyendo un sitio con CSS en estos días?

Web para conocer SMACSS a fondo

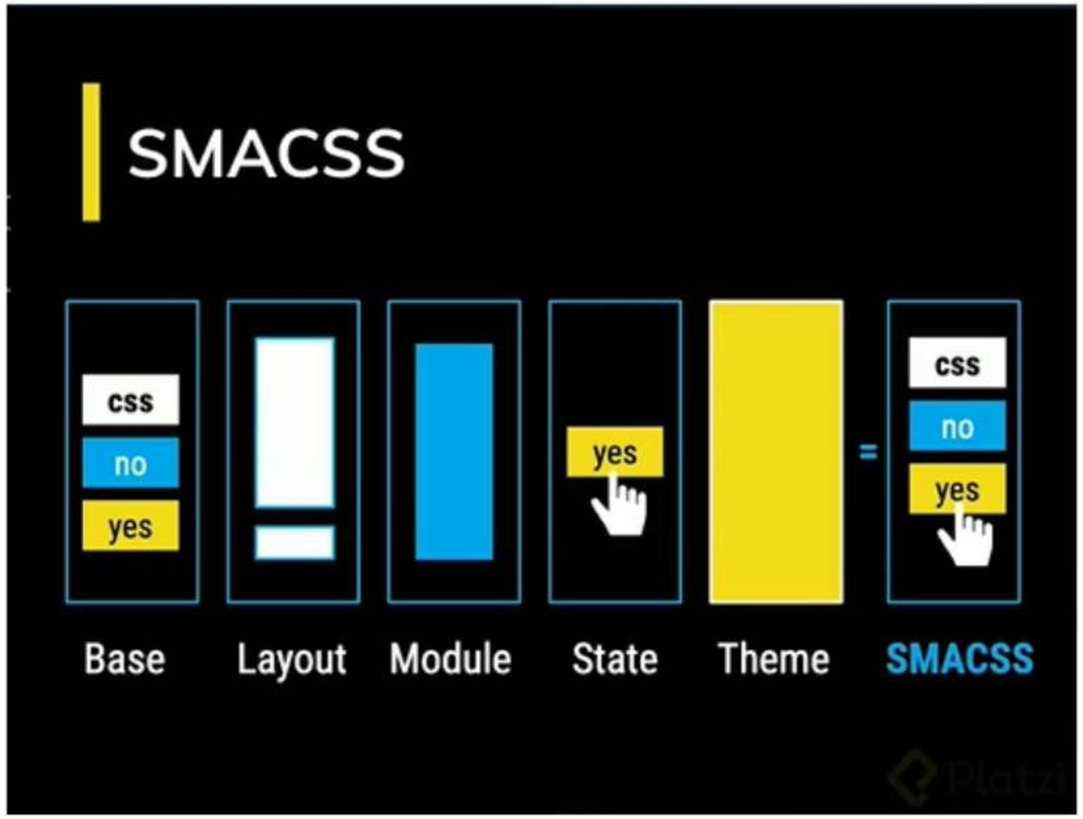
http://smacss.com

sustantiva
CON•SENTIDO

Categorías en SMACSS

La base de Smacss es separar las clases css por su funcionalidad según las siguientes categorías:

Base: Es un reset ampliado. Aquí se definen solo los tags, sin clases ni id´s.
Layout: Aquí va todo lo referente a la estructura de la pagina, las secciones, etc.
Module: Partes reutilizables de la web cuyo css debería ser escrito pensando en su independencia.
State: Todas las propiedades relativas a estados, desplegado, activo, apagado, colapsado, etc.
Theme: Unicamente el aspecto visual, fuentes, colores., etc.



sustantiva

CON•SENTIDO