**Centro de Tecnología y Artes Visuales**

Diseño de Sitios Web II

Proyecto final

CSS Orientado a Objetos (OOCSS)

Aguilar José Andrés

Cruz Danya

Luna Jordan

Martínez María

Solís Marín María José

30 noviembre de 2016, Desamparados, San José

**Object Oriented CSS**

Es una metodología para el diseño de hojas de estilo creada por Nicole Sullivan. Esta metodología o framework consiste en organizar nuestro CSS para que sea más conciso, tenga un mejor rendimiento y sea accesible para otros desarrolladores.

Un ‘Objeto’ en OOCSS constituye, simplemente, un patrón visual de nuestra interfaz que se repite a lo largo de nuestro proyecto. Puede ser representado mediante una porción de código HTML independiente y, sobre todo, puede (y debe) ser reutilizado tantas veces como sea necesario. De este modo, llamaremos al objeto ‘Módulo’.

Estos ‘módulos’ pueden ser desde botones, contenedores genéricos, medias, widgets o incluso una página entera; hasta cualquier elemento de nuestra interfaz que pueda repetirse y ser considerado ‘objeto’. Para construirlo, deberemos tener en cuenta sus posibles variaciones de estilo, estado o incluso contenido así obtendremos un código limpio y reutilizable.

Se basa en dos principios básicos

**• Separar estructura de estilo visual**

Debemos definir la estructura de nuestro elemento. Las medidas width, paddings, displays, etc. Y separarlas de los estilos visuales como backgrounds, colores y tipografías.

Este primer principio abarca la división grafica entre la estructura y todo aquel atributo que le dé una visualización a un bloque de código, siendo este último un objeto grafico que puede ser utilizado en otros elementos, solo llamándolo con una clase dentro de la etiqueta de HTML.

**• Separar el contenedor del contenido**

OOCSS nos pide romper la dependencia entre contenedor y contenido. Para reducir aún más la posibilidad de código redundante, Sullivan nos anima a separar el contenedor y el contenido. Cuando nuestros estilos no están separados por contenedor y contenido no podemos aplicar fácilmente las mismas propiedades a otro elemento fuera de los contenedores especificados. Más específicamente, estas reglas sólo pueden aplicarse a sus respectivos elementos cuando están contenidas en un contenedor de una determinada clase. Debido a que deben estar en un determinado contenedor, los estilos no son tan reutilizables. Como diseñadores, debemos separar las propiedades que pertenecen al contenedor y las que pertenecen al contenido. Las clases sólo deben tener propiedades que pertenecen a uno u otro, pero no a ambos. La creación de esta separación reduce la probabilidad de tener que sobrescribir o repetir propiedades (rendimiento decreciente). Además, hace que nuestro CSS sea más manejable y más fácil de entender. Para maximizar la reutilización de nuestros estilos, debemos mantener una separación entre contenedor y contenido.

Cuando utilizamos selectores adyacentes los estilos aplicados a los elementos hijos dependen completamente de nuestro elemento padre.

.sidebar ul {

font-family: font-family:"Times New Roman",Georgia,Serif;

font-size: 12px;

line-height: 1;

color: red;

}

Aquí, las propiedades de estilo se aplican sólo a los elementos ul que son hijos de cualquier elemento con la clase .sidebar. Si queremos crear el mismo contenido en otro contenedor, fuera de .sidebar, tendríamos que especificar el estilo nuevamente con código redundante. Digamos, por ejemplo, que queríamos aplicar propiedades similares a una lista desordenada en un elemento con la clase .navbar

.sidebar ul {

font-family: font-family:"Times New Roman",Georgia,Serif;

font-size: 12px;

line-height: 1;

color: red;

}

.navbar ul {

font-family: font-family:"Times New Roman",Georgia,Serif;

font-size: 10px;

line-height: 1;

color: blue;

}

Los estilos anteriores no están separados por contenedor y contenido, y como tal, no podemos aplicar fácilmente las mismas propiedades a otro ul fuera de los contenedores especificados. Estas reglas sólo pueden aplicarse a sus respectivos elementos cuando están contenidas en un contenedor de una determinada clase (en este caso, .navbar o .sidebar). Debido a que deben estar en un determinado contenedor, los estilos no son tan reutilizables. Para maximizar la reutilización de nuestros estilos, debemos mantener una separación entre contenedor y contenido.

En el siguiente ejemplo podemos ver como separar el contenido del contenedor, obteniendo un código más semántico y reutilizable.

<div class="mod skin">

<div class="inner">

<h2 class="mod-heading"</h2>

<p class="mod-body"</p>

</div>

</div>

Para poder utilizar esta metodología de la mejor manera podemos adoptar las siguientes prácticas:

**• Crear una librería**

Nuestros componentes debe ser una pieza de lego es decir, combinables y reutilizables.

**• Usar estilos semánticos y consistentes**

El estilo de un nuevo elemento HTML debe ser predecible.

**• Diseño módulos transparentes**

El módulo es el contenedor que se podría utilizar con cualquier contenido.

**• Sea flexible**

Altura y anchura debe ser extensible y adaptarse sí mismos.

**• Aprende a amar los Grids**

Los grids permiten controlar el ancho. La altura se define generalmente por el contenido.

**• Minimizan los selectores**

Mantener una baja especificidad [0-0-1-0] para tener un mejor control sobre los selectores.

**• Separar la estructura de la piel**

Se debe de hacer la distinción, que es un principio fundamental de OOCSS. Crear objetos el esqueleto de la estructura de nuestros bloques y vestir a estos bloques con estilos visuales.

**• Separar contenedor del contenido**

Al hacer una distinción separamos el recipiente de su contenido. De esta forma nuestros objetos no dependen del selector padre.

**• Extender objetos con múltiples clases**

Los objetos son como Legos pones juntos para construir el resultado esperado.

**• YUI**

Podemos optar por utilizar la librería de Yahoo para complementar el uso de OOCSS.

**Media Object**

El objeto multimedia es una imagen a la izquierda, con contenido descriptivo a la derecha (posicionados a la inversa). El contenido puede ser cualquier otro objeto. Puede contener texto, listas, cuadrículas o incluso otros objetos multimedia dentro. Existen muchas versiones diferentes, el objeto multimedia es un módulo reutilizable que salvará a escribir un montón de código, ya que no lo vas a repetir, sin importar su tamaño con el contenido de su propiedad.

El Media Object es una imagen a la izquierda, con el contenido descriptivo de la derecha, como esta caja de comentario de Facebook por ejemplo:

El área de contenido de la derecha puede contener otros objetos. En este caso, contiene texto, pero que podría poner listas, rejillas, o incluso otros objetos multimedia dentro (imágenes, audio, videos).

La primera cosa que debemos hacer es ver qué componentes se pueden ser reusables y definir lo que sabemos y no acerca de estos.

*¿Qué sabemos?*

• Se puede anidar

• Debo poner Clearfix

*¿Qué hemos decidido “no” saber?* (Pensemos en flexibilidad)

• Ancho de las imágenes, los márgenes y la decoración son variantes.

• El contenido es desconocido.

• El ancho de este object es desconocido.

Una vez que se construye el Object Media podremos utilizarlo para crear muchos de los mismos objects básicos. En la siguiente imagen Nicole destaca todos los Object que se ven que incluso en aplicar esto ahorra mucho código que dejaríamos de estar repitiendo en el css.

(Object Media es el resaltado en rojo)

**Implementación de Media Object**

La parte difícil es tener seguro que la imagen que utilicemos puede tener un ancho variable de modo que el elemento es reusable. Esto significa que nuestro Object Media debe ser flexible para que pueda llenar todo el espacio que está disponible.

Ejemplo:

<div class = "media attribution">

<a href="http://twitter.com/stubbornella" class="img">

<img src = "http://stubbornella.com/img.jpg" alt = " me "/>

</a>

<div class =" bd ">

@Stubbornella 14 minutes ago

</ div>

</ div>

/\* ====== media ====== \*/

.media {

margin: 10px;

}

.media, .bd {

overflow: hidden;

overflow: visible;

zoom: 1;

}

.media .img {

float: left;

margin-right: 10px;

}

.media .img img {

display: block;

}

.media .imgExt {

float: right;

margin-left: 10px;

}

**Beneficios de OOCSS**

Algunos de los principales beneficios que obtenemos al utilizar OOCSS son los siguientes:

***Escalable***

Se aplica a proyectos de cualquier tamaño.

***Reutilizable***

Si aplicamos los dos principios principales de OOCSS podríamos aplicar estos estilos a los largo de la página en componentes que sean similares o con la necesidad de esos estilos.

***Hojas de estilos mantenibles y más pequeños***

En algunos casos cuando tenemos proyectos muy grandes nuestro css se vuelve una guerra de especificidad por reescribir los estilos de cada elemento en diferentes situaciones.

Con OOCSS la idea de reutilizar código nos permite hacer cambios a las diferentes secciones en donde fueron aplicados en un sólo lugar y así añadir el contenido o cambio deseado de manera más rápida y fácil.

***Sitios web más rápidos***

Cuando se tiene hojas de estilos más pequeñas y reutilizadas a lo largo de una página será más rápida su carga. Menor llamado del browser a los estilos y menor tiempo del browser renderizando los mismos.

***Freebies***

Algo a tener en cuenta es que si deseas añadir algún nuevo componente o reutilizas un objeto en la página y tienes todos estos estilos reutilizables solamente tendrían que ser aplicados, esto nos hace crear nuevos elementos con estilo, libres de nuevas líneas de código CSS. Algo que es sumamente apreciado en proyectos grandes y de gran ganancia.

Si bien es cierto que hará nuestro html un poco más largo, esto nos permitirá reutilizar estilos específicos a lo largo de nuestra página.

***Código más limpio***

Teniendo en cuenta todas las posibilidades al usar OOCSS, sabemos que nuestros CSS serán más pequeños entendibles y por lo tanto más limpios. Además nos hará evitar el caos de sobrescribir muchos elementos agregar ¡importants e ID para combatir la especificidad y darle a nuestro CSS una cascada más natural.

***Librerías***

Si pensamos en reutilizar código y hacerlo de una manera más fácil podríamos pensar en crear nuestras propias librerías y de esta manera aplicar este css en distintas páginas que estemos creando.

***Documentación***

Una de las mayores ventajas que podemos tener es la documentación directa de la principal exponente de OOCSS. Nicole Sullivan nos comparte vía Github un repo con toda la información necesaria y las dudas que podríamos tener acerca de OOCSS; además de que puedes bajar ese repo y aplicar todo lo que la documentación indica puesto que está creado para practicar y aprender.

**Recomendaciones**

Si queremos tener un mejor rendimiento al utilizar la metodología, podemos seguir las siguientes directrices:

***Evitar ID en los estilos***

Existen dos razones:

Estropean la especificidad porque son muy fuertes (principal razón).

Son identificadores únicos, que hace que los componentes construidos con ellos algo así como singletons, no reutilizables en la misma página.

Las clases nos permiten poder escribir código reutilizable, ya que podemos usarlas en diversos elementos de nuestro HTML y son más fáciles de acceder, mientras que los ID no, porque estos deben de ser únicos y en caso de querer reutilizar alguna serie de estilos aplicados a un ID no podríamos hacerlo y por lo tanto no estaríamos reutilizando nuestro código.

***Evitar selectores descendientes***

Por qué repetimos código y los estilos que se declaran usando el selector descendiente no son reutilizables, ya que dependen de un contenedor en particular.

***Evitar adjuntar clases a elementos***

Esto son estilos que afectan el contenido según la ubicación que tengan en la página. Para esto, ella recomienda agregar una clase base para el objeto HTML y agregar clases “hija” que son las que aplican los estilos según la ubicación en la página; con esto, este CSS puede ser fácilmente reutilizado en otro lugar del sitio.

***Usar CSS Lint para comprobar el css***

Es una herramienta de calidad de código para ayudar a los desarrolladores a detectar malas prácticas en su css, creada por Nicole Sullivan y Nicholas Zakas.

CSS Lint señala los problemas de tu código CSS. Hace un chequeo básico de tu sintaxis, además de aplicar una serie de ajustes al código que parece mostrar signos de ineficiente o de problemática. Las reglas son todas opcionales, así que puedes editar u omitir sencillamente las que no te gusten.

***Usar CSS Grids***

Sistema de cuadriculas en el que se implementa el uso de divisiones del layout, controlando así los widths. Se utilizan para dividir cualquier espacio en fracciones, pueden anidarse. Una unidad puede contener otra línea o puede contener otros objetos directamente.

***Esto va de acuerdo a estas propiedades:***

***line***

Agrupa unidades en una línea horizontal. Nota: las unidades de diseño móviles se pueden apilar para evitar desplazamientos horizontales.

***unid***

Clase base que divide una línea en secciones (columnas).

***sizeXofY***

Amplía la unidad. Indica el ancho fraccional de la unidad, por ejemplo el tamaño 3 de 4 ocuparía tres cuartos, o el 75%, del espacio horizontal. Se soportan las siguientes fracciones: 1 1/2 1/3 2/3 1/4 3/4 1/5 2/5 3/5 4/5.

***lastUnit***

Amplía la unidad. Aplicado al último hijo de cada línea.

***Ejemplo:***

<div class="line">

<div class="unit size1of2">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit ... </p>

</div>

<div class="unit size1of2 lastUnit">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit ... </p>

</div>

</div>

En conclusion la metología de CSS orientado a objetos facilitara la reutilizacion de código en nuestras hojas de estilos. Creando nuestras propias librerías, tendremos componentes los cuales podemos utilizar en todo nuestro proyecto, esto quiere decir que es escalable ya que lo podemos aplicar para proyectos de pequeña **y** grande escala. Ademas que obtendremos sitios web mas rápidos y la capacidad de dar a estilos a nuestros elementos sin lineas de código (freebies).