

# Metodologías para la organización y el preprocesador Sass

Metodología OOCSS

***Implementar una interfaz  
de usuario web utilizando  
buenas prácticas  
en el manejo de estilos  
para brindar un aspecto  
visual e interacciones  
acordes a lo requerido***

- Unidad 1:  
Metodologías para la organización  
y el preprocesador Sass
- Unidad 2:  
El modelo de cajas y el Layout
- Unidad 3:  
Utilizando Bootstrap como  
Framework CSS



Te encuentras aquí



## ¿Qué aprenderás en esta sesión?

- *Describe las características principales de la metodología OOCSS para la organización y modularización de estilos.*

La metodología OOCSS  
se basa en 2 principios.

¿Cuáles son?



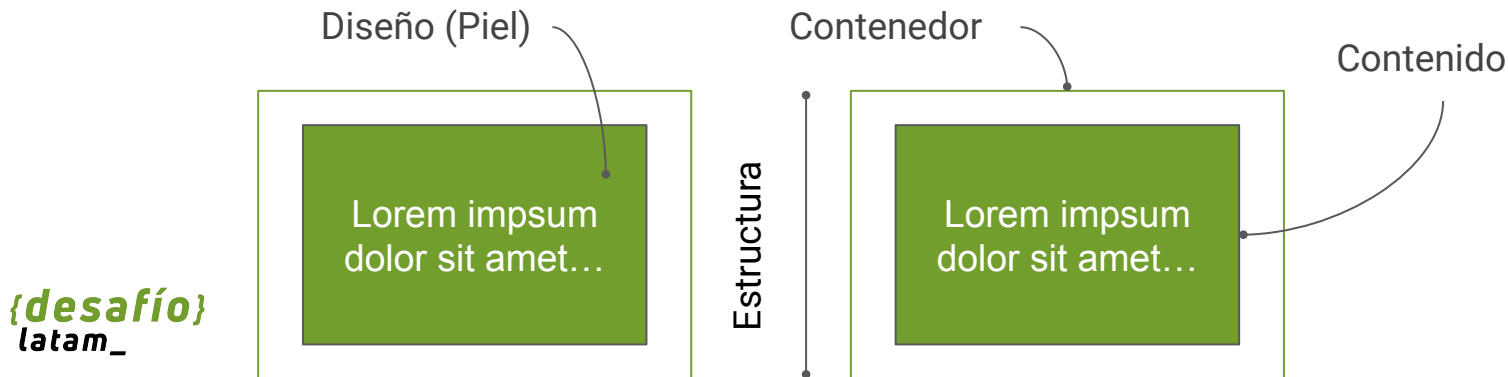
**/\* Metodología OOCSS \*/**

# En qué consiste esta metodología

El propósito de la metodología OOCSS es fomentar la reutilización de código, para así desarrollar hojas de estilos más rápidas y eficientes, las que resultan más fáciles de escalar y mantener.

Esta metodología se basa en 2 principios:

- Separar la **estructura** de la **piel** (Características visuales).
- Separación del **contenedor** de su **contenido**.



# Conceptos claves de OOCSS


## *CSS orientado a objetos*

La **programación orientada a objetos** es un paradigma de programación que se centra en crear objetos reutilizables y establecer relaciones entre ellos.

Un **objeto en CSS** se refiere a un elemento HTML o también puede referirse a una clase CSS o un método de JavaScript.

Un ejemplo común son los botones, los cuales tendrán la misma estructura (width, border, margin) etc.

Pero tendrán colores distintos de acuerdo a cada función.



Aceptar



Cancelar

# Principios básicos de OOCSS

*Separar la estructura de la piel*



La mayoría de los elementos que existen en una página web poseen diversas características visuales (**pieles**) que se repiten en diferentes contextos.

Por otra parte, tenemos otras características “invisibles” (estructura) que también se repiten.

Cuando logramos abstraer estas características en diferentes módulos, podemos reutilizarlos y aplicarlos en cualquier elemento.



# Principios básicos de OOCSS

*Separar la estructura de la piel*

En CSS las **propiedades de estructura** son invisibles para el usuario, pero modifican el tamaño y posición de un elemento, como por ejemplo:

- Height
- Width
- Margin
- Padding



# Principios básicos de OOCSS

*Separar la estructura de la piel*



También tenemos propiedades que **modifican el aspecto visual** de un elemento (propiedades de la **piel**), como por ejemplo:

- Color
- Font
- Shadow
- Gradient

# Principios básicos de OOCSS

## *Separar la estructura de la piel*

Para poder aplicar este principio de separar la estructura de la piel, debemos diferenciar las **propiedades que modifican la estructura** de las **propiedades que modifican la piel o diseño**.

Este es un ejemplo de estilos que mezclan las propiedades de estructura y de piel:



```
12 <button class="btn-aceptar">Aceptar</button>
13 <button class="btn-cancelar">Cancelar</button>
```



```
1  .btn-aceptar {
2    width: 100px;
3    height: 30px;
4    padding: 10px;
5    border: solid 1px #000;
6    background-color:#64CB29;
7  }
8
9  .btn-cancelar {
10   width: 100px;
11   height: 30px;
12   padding: 10px;
13   border: solid 1px #000;
14   background-color:#D33434;
15 }
```

# Principios básicos de OOCSS

## *Separar la estructura de la piel*

En el ejemplo anterior no estaba separada la estructura de la piel, ya que las propiedades de estructura estaban repetidas en ambas clases.

Si aplicamos este principio los estilos quedan de esta manera:



```
12 <button class="btn aceptar">Aceptar</button>
13 <button class="btn cancelar">Cancelar</button>
```



```
1  .btn {
2    width: 100px;
3    height: 30px;
4    padding: 10px;
5    border: solid 1px #000;
6  }
7
8  .aceptar {
9    background-color:#64CB29;
10 }
11
12 .cancelar {
13   background-color:#D33434;
14 }
```

# Principios básicos de OOCSS

## *Separación del contenedor de su contenido*

El segundo principio es la separación de los contenedores de su contenido, para ver su importancia veamos el siguiente ejemplo:

Estos estilos se aplicarán a cualquier h4 que sea hijo del elemento con id sidebar.

Pero, ¿cómo aplicamos los mismos estilos a los h4 de otra sección, pero con pequeñas variaciones, como un tamaño de fuente diferente o una sombra de texto modificada?

**{desafío}**  
latam\_



```
1 #sidebar h4 {  
2     font-family: Arial, Helvetica, sans-serif;  
3     font-size: .8em;  
4     line-height: 1;  
5     color: #777;  
6     text-shadow: rgba(0, 0, 0, .3) 3px 3px 6px;  
7 }
```

# Principios básicos de OOCSS

## *Separación del contenedor de su contenido*

Para aplicar el mismo estilo de los h4 definidos anteriormente deberíamos hacer algo como esto:

```
9  #sidebar h4, #section-post h4 {
10    font-family: Arial, Helvetica, sans-serif;
11    font-size: 2em;
12    line-height: 1;
13    color: #777;
14    text-shadow: rgba(0, 0, 0, .3) 3px 3px 6px;
15  }
16
17  #section-post h4 {
18    font-size: 1.5em;
19    text-shadow: rgba(0, 0, 0, .3) 2px 2px 4px;
20  }
```

Otra forma de hacerlo es de la siguiente manera:

```
9  #sidebar h4 {
10    font-family: Arial, Helvetica, sans-serif;
11    font-size: 2em;
12    line-height: 1;
13    color: #777;
14    text-shadow: rgba(0, 0, 0, .3) 3px 3px 6px;
15  }
16
17  #section-post h4 {
18    font-family: Arial, Helvetica, sans-serif;
19    font-size: 1.5em;
20    line-height: 1;
21    color: #777;
22    text-shadow: rgba(0, 0, 0, .3) 2px 2px 4px;
23  }
```

# Principios básicos de OOCSS

## *Separación del contenedor de su contenido*

En el ejemplo anterior **duplicamos estilos innecesariamente**, además los declaramos utilizando selectores descendientes:

- `#sidebar h4 { }`
- `#section-post h4 { }`

```
9  #sidebar h4 {  
10     font-family: Arial, Helvetica, sans-serif;  
11     font-size: 2em;  
12     line-height: 1;  
13     color: #fff;  
14     text-shadow: rgba(0, 0, 0, .3) 3px 3px 6px;  
15 }  
16  
17  #section-post h4 {  
18     font-family: Arial, Helvetica, sans-serif;  
19     font-size: 1.5em;  
20     line-height: 1;  
21     color: #fff;  
22     text-shadow: rgba(0, 0, 0, .3) 2px 2px 4px;  
23 }
```

# Principios básicos de OOCSS

## *Separación del contenedor de su contenido*

```
9 #sidebar h4 {
10   font-family: Arial, Helvetica, sans-serif;
11   font-size: 2em;
12   line-height: 1;
13   color: #fff;
14   text-shadow: rgba(0, 0, 0, .3) 3px 3px 6px;
15 }
16
17 #section-post h4 {
18   font-family: Arial, Helvetica, sans-serif;
19   font-size: 1.5em;
20   line-height: 1;
21   color: #fff;
22   text-shadow: rgba(0, 0, 0, .3) 2px 2px 4px;
23 }
```

Al usar estos selectores descendientes no podemos reutilizarlos, ya que siempre dependemos de un contenedor en particular, en este caso el sidebar o la sección de post.

Aplicando la metodología OOCSS nos aseguramos de que nuestros estilos no dependan de ningún elemento contenedor, permitiéndonos modularizar y reutilizar los estilos que sean necesarios.



# Principios básicos de OOCSS

## *Separación del contenedor de su contenido*

Dado lo anterior, la forma correcta de implementar estos estilos es la siguiente:

```
18 h4 {  
19   font-family: Arial, Helvetica, sans-serif;  
20   line-height: 1;  
21   color: #777;  
22 }  
23  
24 h4.sidebar-title {  
25   font-size: 2em;  
26   text-shadow: rgba(0, 0, 0, .3) 3px 3px 6px;  
27 }  
28  
29 h4.section-post {  
30   font-size: 1.5em;  
31   text-shadow: rgba(0, 0, 0, .3) 2px 2px 4px;  
32 }
```

De esta forma añadimos una clase a cada h4 que tenga variaciones pero siempre reutilizando el estilo base del encabezado.

# Ejercicio “Aplicando OOCSS”



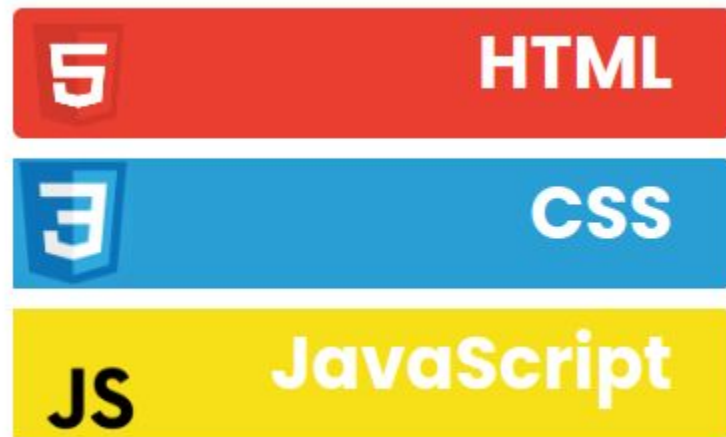
# Ejercicio

## Aplicando OOCSS

Un cliente nos pidió ordenar el código de un widget para su blog.

Al revisar el código, vemos que existen estilos y propiedades que se duplican entre los distintos elementos, debemos:

- Organizar el código HTML y CSS, aplicando los principios de la metodología. Para eso se deben crear las clases necesarias para poder así crear componentes reutilizables.



¿A qué nos referimos cuando  
hablamos de “CSS orientado  
a objetos”



¿Cuál es el propósito de la metodología OOCSS?



# Resumen

## Subtítulo

- Un **objeto en CSS** se refiere a un elemento HTML o también puede referirse a una clase CSS o un método de JavaScript.
- Esta metodología se basa en 2 principios:
  - Separar la estructura de la piel (Características visuales)
  - Separación del contenedor de su contenido.
- Las **propiedades de estructura** son invisibles para el usuario, pero modifican el tamaño y posición de un elemento.
- Las **propiedades de diseño (piel)** modifican el aspecto visual de los elementos.



## Próxima sesión...

- *Describe las características principales de la metodología SMACSS para la organización y modularización de estilos.*

**{desafío}**  
**latam\_**

*Academia de  
talentos digitales*

