Esta clase va a ser

• grabada

Certificados oficialmente por

PedidosYa

CODERHOUSE

Clase 08. DESARROLLO WEB

Pseudoclases y BEM

Certificados oficialmente por

CODERHOUSE

CLASE N°07

Glosario

CSS Grid: es el sistema de maquetación más potente que hay disponible. Se trata de un sistema en 2D que permite definir filas y columnas (a diferencia de, por ejemplo, Flexbox, el cual funciona en una única dimensión).

Diseño responsive: se refiere a la idea de que un sitio web debería mostrarse igual de bien en todo tipo de dispositivo, desde monitores de pantalla panorámica hasta teléfonos móviles.
El diseño responsive se logra a través de "Media Queries" de CSS. Pensemos en las Media Queries como una forma de aplicar condicionales a las reglas de CSS.

CLASE N°07

Glosario

Mobile First: significa crear el código primero para los dispositivos más pequeños que los usuarios probablemente tengan, como teléfonos o tabletas. Implica trabajar en el dispositivo más pequeño, y luego acumular desde allí todo en el mismo código y el mismo proyecto, en lugar de hacer uno nuevo para cada tamaño de pantalla.

Meta viewport: una etiqueta <meta> viewport da al navegador las instrucciones sobre cómo controlar las dimensiones, y el ajuste a escala de la página.

Objetivos de la clase

- Identificar y aplicar pseudoclases
- Conocer y comenzar a utilizar la metodología BEM.

Temario

07

Grids II

- ✓ Grids Mobile First
- ✓ Media queries
- ✓ Grids+Flex+@Media

08

Pseudoclases y BEM

- ✓ Pseudoclases
- ✓ BEM

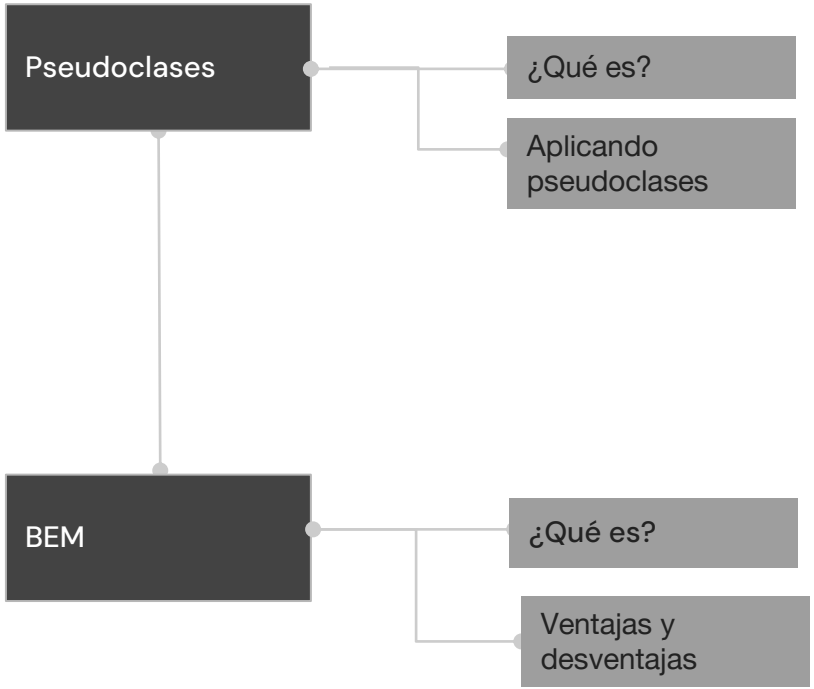
09

GIT

- ✓ Git
- ✓ Comandos básicos de la terminal
- ✓ Creando repositorios
- ✓ Ramas

CODERHOUSE

MAPA DE CONCEPTOS



Pseudoclases

¿Qué es una pseudo clase? 

Una pseudoclase CSS es una palabra clave que se añade a los selectores y que especifica un estado especial del elemento seleccionado.

Por ejemplo, ***:hover*** aplicará un estilo cuando el usuario haga hover sobre el elemento especificado por el selector.

selector:pseudoclase { propiedad: valor; }

¿Qué es una pseudoclase? 

Las pseudoclases, junto con los pseudoelementos, permiten aplicar un **estilo** a un elemento no sólo en relación con el contenido del árbol de documento, sino **también en relación a factores externos** como:

- ✓ El historial del navegador (***:visited***, por ejemplo),
- ✓ El estado de su contenido (como ***:checked*** en algunos elementos de formulario),
- ✓ La posición del ratón (como ***:hover*** que permite saber si el ratón está encima de un elemento o no).

Teniendo como referencia...

HTML

```
<nav>
  <a href="#uno">Link 1</a>
  <a href="#dos">Link 2</a>
  <a href="#tres">Link 3</a>
  <a href="#cuatro">Link 4</a>
  <a>Link 5</a>
</nav>
<div>
  <div class="otro">
    <div class="hijo">Soy un div</div>
    <div class="otro">Soy un div</div>
  </div>
  <div class="padre">
    <div class="otro">Soy un div</div>
    <div class="hijo">Soy un div</div>
        <div class="hijo">Soy un
div</div>
  </div>
</div>
```

CSS

```
/*Estilos base para los divs*/

div{
        border: solid grey 1px;
        padding:10px;
}

/*Estilos base para los enlaces*/

a{  /*Todos los elementos <a>*/

        display:inline-block;
        padding:10px;
        font-family:Verdana, Geneva, Tahoma, sans-serif;
        background-color: azure;
        color:rgb(10, 10, 60);

}
```

Apliquemos pseudoclases

```
a:link{ /*Todos los elementos <a> que tengan un atributo href definido y no hayan sido visitados*/
    background-color: aqua;
    color: darkslateblue;
}

a:hover { /*Todos los elementos <a> cuando el mouse esté posicionado sobre ellos*/
    font-weight: bold;
    padding:20px;
    background-color: rgb(10, 10, 60);
    color: azure;
}
```

Apliquemos pseudoclases

```
a:active {/*Todos los elementos <a> cuando se está haciendo click sobre ellos*/
    background-color: yellow;
    color: black;
}

a:visited {/*Todos los elementos <a> que tengan un atributo href definido y ya hayan sido
visitados*/
    background-color: darkslategrey;
    color: #999;
}
```

Apliquemos pseudoclases

Otra pseudo clase muy útil pero poco utilizada, aplica a todos los elementos A, excepto a los incluidos como elementos B. Es decir, podríamos aplicar unos estilos a todos los elementos “div” y evitar que éstos se apliquen a otras capas con un id o class determinado. En el siguiente ejemplo hacemos justo esto, evitando que se apliquen los estilos a las capas “padre“.

```
div:not(.padre){
    padding-left:40px;
}
```

Apliquemos pseudoclases

La solución a todos los problemas de maquetación de antaño con las filas y los bloques.

👉 Con :nth-child(N) podremos aplicar sus estilos a todos los elementos hijos cuya posición sea un número “N” respecto a un padre. Este número N no tiene que ser necesariamente un número entero para especificar una posición fija (la posición 2, por ejemplo), ya que también permite insertar fórmulas y palabras específicas.

```
.padre div:nth-child(2) {
    font-weight: bold;
    color: orange;
}
```

Break

¡10 minutos y volvemos!

BEM

BEM

Todos queremos hacer que **nuestro código sea más fácil de leer**. Esto nos ayuda a trabajar más rápidamente y de manera eficiente, y cuando otros trabajen con nosotros podremos mantener claridad y coherencia.

Hoy vamos a descubrir la metodología BEM, que nos ayudará a entender estructuras de CSS, y a mejorar las nuestras.

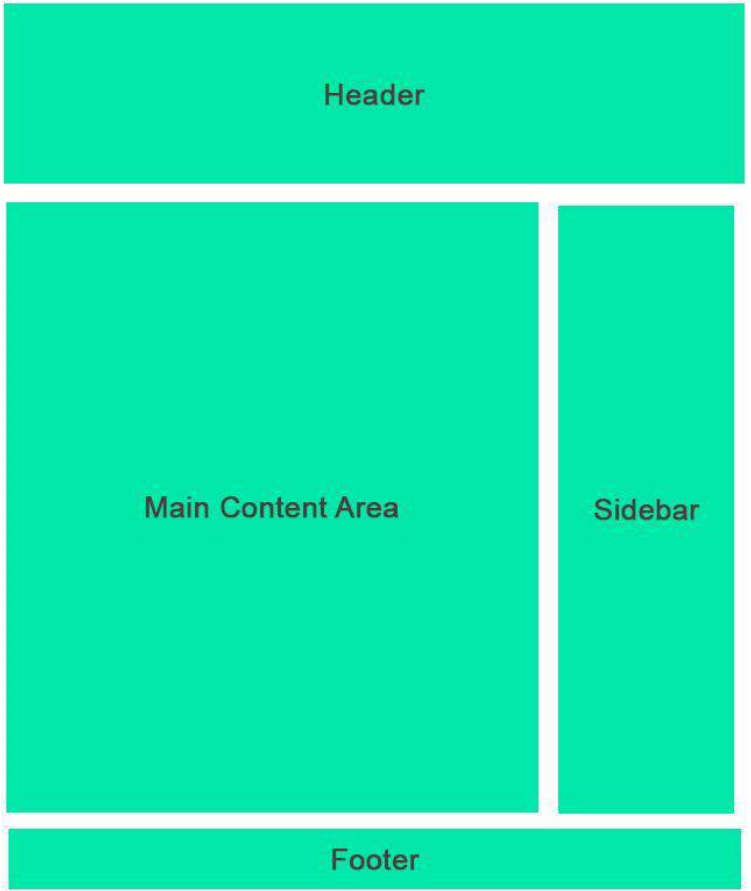
BEM

👉 **BEM** significa Modificador de Bloques de Elementos (Block Element Modifier) por sus siglas en inglés. Sugiere una manera estructurada de nombrar tus clases, basada en las propiedades del elemento en cuestión.

👉 **BEM** tiene como horizonte modularizar lo máximo posible cada uno de los bloques de código dispuesto. Se centra en tres parámetros o variables posibles: **bloques** (*div, section, article, ul, ol*, etc.), **elementos** (*a, button, li, span*, etc.) y **modificadores**. Estos últimos se definen de acuerdo a la posterior utilización que haga el desarrollador a cargo.

BLOQUE 1

El bloque es un **contenedor o contexto donde el elemento se encuentra presente**. Piensa como si fueran partes estructurales de código más grandes. Puede que tengas un encabezado, pie de página, una barra lateral y un área de contenido principal; cada uno de estos sería considerado como un bloque.



1 BLOQUE

El bloque de elementos forma la raíz de la clase y siempre irá primero. Solo debes saber que una vez que has definido tu bloque, estarás listo para comenzar a nombrar tus elementos.

```
.block__element {
    background-color: #FFFFFF;
}
```

BLOQUE 1

El doble guión bajo (*underscore*) te permite visualizar, navegar rápidamente y manipular tu código. Aquí hay algunos ejemplos de cómo funciona la metodología de elementos:

```
.header__logo {}
.header__tagline {}
.header__searchbar {}
.header__navigation {}
```

BLOQUE 1

El punto es mantener los nombres **simples, claros, y precisos.**
No lo pienses demasiado. 🚀

Actualizar el nombre de las clases no debería ser un problema cuando encuentras una mejor semántica que funcione (sólo debes tratar de ser consistente, y apegarte a ella).

BLOQUE 1

HTML

```
<section>
   <article class="noticia">
      <!-- Bloque contenedor -->
   </article>
</section>
```

CSS

```
.noticia {
   font-family: Georgia, 'Times New Roman', Times, serif;
   color: darkslategray;
   background: lightgray;
}
```

CODERHOUSE

ELEMENTOS 2

El elemento es una de las piezas que compondrán la estructura de un bloque. El bloque es el todo, y los elementos son las piezas de este bloque.

De acuerdo a la metodología BEM, cada elemento se escribe después del bloque padre, usando dos guiones bajos.

ELEMENTOS 2

HTML

```
<section>
  <article class="noticia">
    <h1 class="noticia__titulo">Título de la noticia</h1>
  </article>
</section>
```

CSS

```
.noticia__titulo {
  font-family: Verdana, Geneva, Tahoma, sans-serif;
  text-transform: capitalize;
}
```

3 MODIFICADORES

Cuando nombras una clase, la intención es ayudar a que ese elemento pueda ser repetido, para que no tengas que escribir nuevas clases en otras áreas del sitio si los elementos de estilo son los mismos.

Cuando necesitas modificar el estilo de un elemento específico, puedes usar un modificador. Para lograr esto, añade un doble guión -- luego del elemento (o bloque). Aquí tenemos un corto ejemplo:

```
.block--modifier {}
.block__element--modifier {}
```

MODIFICADORES

3

HTML

```
<section>
  <article class="noticia noticia--destacada">
    <h1 class="noticia__titulo--uppercase">Título de la noticia</h1>
    <p>Texto de la noticia</p>
  </article>
</section>
```

CSS

```
.noticia--destacada{
  background-color: cornsilk;
}

.noticia__titulo--uppercase{
  text-transform: uppercase;
}
```

Ejemplo en vivo

¡Vamos a practicar lo visto!

BEM - Ventajas

- Añade especificidad.
- Aumenta la independencia.
- Mejora la herencia múltiple.
- Permite la reutilización.
- Entrega simplicidad.

BEM - Desventajas

- Las convenciones pueden ser muy largas.
- A algunas personas les puede tomar tiempo aprender la metodología.
- El sistema de organización puede ser difícil de implementar en proyectos pequeños.

BEM: ¿Para qué lo usarías?

- Para simplificar nuestro CSS y conseguir un estilo consistente, por lo que nuestro código será mucho más legible y fácil de mantener.

- Si estamos usando un framework de desarrollo web y queremos modificar ciertas clases.

- Cuando trabajamos en equipo y cada miembro tiene una manera distinta de escribir CSS.

¿Preguntas?

Encuesta

Por encuestas de Zoom

¡Terminamos el módulo 4: animaciones!

Cuéntanos qué temas te resultaron más complejos de entender. **Puedes elegir más de uno**. Vamos a retomar aquellos temas que resultaron de mayor dificultad en el próximo AfterClass.

CODERHOUSE

MATERIAL AMPLIADO

Recursos multimedia

- ✓ Más información sobre BEM | *BEM 101*
- ✓ Más información sobre OOCSS | *Smashing Magazine*

Disponible en nuestro repositorio.

Resumen
de la clase hoy

- ✓ Concepto y aplicación de Pseudoclases
- ✓ BEM

CODERHOUSE

¿Sabías que premiamos a nuestros estudiantes por su dedicación?

Conoce los beneficios del **Top 10**

CODERHOUSE

Opina y valora esta clase

CODERHOUSE

Muchas gracias.

#DemocratizandoLaEducación

CODERHOUSE