

***RECUERDA PONER A GRABAR LA
CLASE***





¿DUDAS DEL ON-BOARDING?

MIRALO AQUI



Clase 11. DESARROLLO WEB

SASS /



OBJETIVOS DE LA CLASE

- Comprender SASS.
- Conocer BLEM.

GLOSARIO:

Clase 10

JavaScript: es un lenguaje con muchas posibilidades. Se utiliza para crear pequeños programas que luego son insertados en una página web, y en programas más grandes, orientados a objetos mucho más complejos. Con Javascript podemos crear diferentes efectos e interactuar con nuestros usuarios.

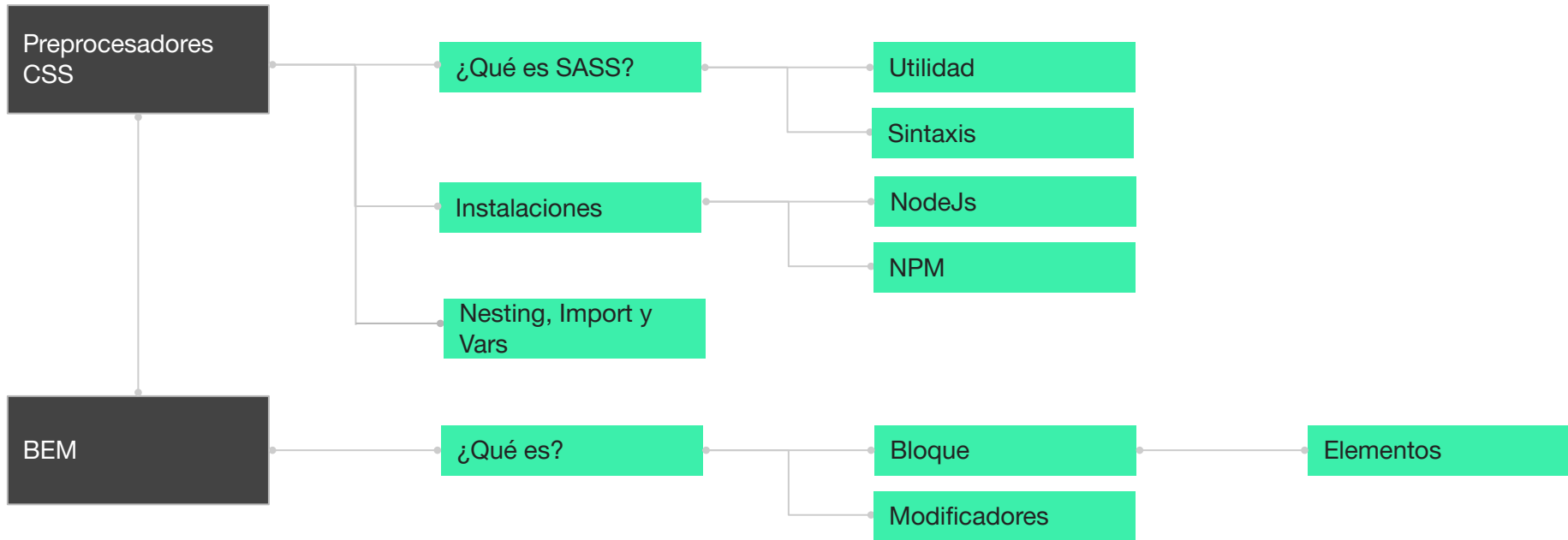
Bootstrap themes: son marcos contruidos por expertos, que permiten tener un diseño base como una extensión de Bootstrap, especialmente para un conjunto específico de problemas.

Página responsive (repaso): el sistema detecta automáticamente el ancho de la pantalla y, a partir del mismo, adapta todos los elementos de la página, desde el tamaño de letra hasta las imágenes y los menús, ofreciendo al usuario la mejor experiencia posible.

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS CLASE 11

¡Para
recordar!



CRONOGRAMA DEL CURSO

Clase 10



Bootstrap + Themes



PRÁCTICAS DE LO
VISTO EN CLASE



BOOTSTRAP



BOOTSTRAP CON JS

Clase 11



SASS I



PRÁCTICAS DE LO
VISTO EN CLASE



APLICANDO SASS

Clase 12



SASS II



PRÁCTICAS DE LO
VISTO EN CLASE



APLICANDO SASS -
OPERACIONES



TERCERA ENTREGA
DEL PROYECTO FINAL



GUIÓN DE LA CLASE

Accede al material complementario [aquí](#).

PREPROCESADORES CSS

¿QUÉ ES SASS?

Es un preprocesador de CSS que te permite escribir un código, el cual luego se transforma (compila) en un archivo de CSS puro.

Esto genera un código más limpio y sencillo de mantener y editar, a través de una estructura ordenada, usando un lenguaje de estilos.

SASS

Sass significa “**Syntactically Awesome Stylesheets**”. Permite crear hojas de estilos estructuradas, limpias y fáciles de mantener.

Con SASS podrás escribir hojas de estilo que te ayudarán a generar ficheros **CSS más optimizados**, incorporando mayor contenido semántico.

Esto permite utilizar funcionalidades que normalmente encontrarías en lenguajes de programación tradicionales, como el uso de variables, creación de funciones, etcétera.

SASS: ¿POR QUÉ ES ÚTIL?

Normalmente, crear una hoja de estilos es relativamente sencillo. Lo malo es cuando el proyecto va creciendo en tamaño: su CSS puede terminar siendo muy extenso.

SASS permite una sintaxis más simple y elegante, implementando además bastantes características extra, para hacer más manejable tu hoja de estilos.

SASS: SINTAXIS

En Sass cuentas con dos diferentes tipos de sintaxis: SCSS y SASS. La primera y más popular, es conocida como SCSS (Sassy CSS). Es muy similar a la sintaxis nativa de CSS, tanto así que te permite importar hojas de estilos CSS (copiar y pegar) directamente en un archivo SCSS, y obtener un resultado válido.

Para utilizarla, sólo debes crear un archivo con terminación .scss de la siguiente manera: *archivo.scss*

SASS: SINTAXIS

¿Crees que es válido el siguiente CSS dentro de un SCSS?

```
div {  
  width: 100px;  
  height: 100px;  
  background-color: red;  
  padding: 15px;  
}
```

```
div p {  
  font-size: 20px;  
  color: white;  
  font-family: Arial, sans-serif;  
}
```

SASS: SINTAXIS



SI

SASS: SINTAXIS

Entonces, ¿cómo se escribe el SCSS?
¿igual que el CSS?

Si bien es válido el CSS tal como lo
escribimos, podemos ir de a poco
agregando la sintaxis SCSS.

Siguiendo el ejemplo anterior, podría quedar
de la siguiente forma.

¿Notas la diferencia sutil?

```
div {  
  width: 100px;  
  height: 100px;  
  background-color: red;  
  padding: 15px;  
  p {  
    font-size: 20px;  
    color: white;  
    font-family: Arial, sans-serif;  
  }  
}
```

Ejemplo
en vivo



INSTALACIÓN DEL NODEJS Y EL NPM

REPASANDO LA INSTALACIÓN DEL PROCESADOR

1

Instala **nodejs**.

2

Instala **npm**.

3

Ingresa al
directorio del
repositorio.

4

Inicia el npm,
con **npm
init**.

REPASANDO LA INSTALACIÓN DEL PROCESADOR

4

Instala el nodemon: **npm install -D node-sass nodemon.**

5

Crea la carpeta SCSS y CSS y sus archivos respectivos.

6

Edita el package.json e inserta las líneas.

"build-css":

- "node-sass --include-path scss scss/prueba.scss css/pruebacss.css",
- "watch-css": "nodemon -e scss -x \"npm run build-css\""

7

Compila con npm: **run watch-css.**

COMANDAR PARA COMPILAR

Todo está listo para escribir un pequeño script para compilar Sass. Abre el archivo *package.json* en un editor de código. Verás algo: como

```
{} package.json ●
1  {
2    "name": "clasesass",
3    "version": "1.0.0",
4    "main": "index.js",
5    "scripts": {
6      "test": "echo \"Error: no test specified\" && exit 1"
7    },
8    "author": "Sebastian",
9    "license": "ISC",
10   "devDependencies": {
11     "node-sass": "^4.7.2",
12     "nodemon": "^1.14.11"
13   },
14   "description": ""
15 }
16
```

COMANDAR PARA COMPILAR

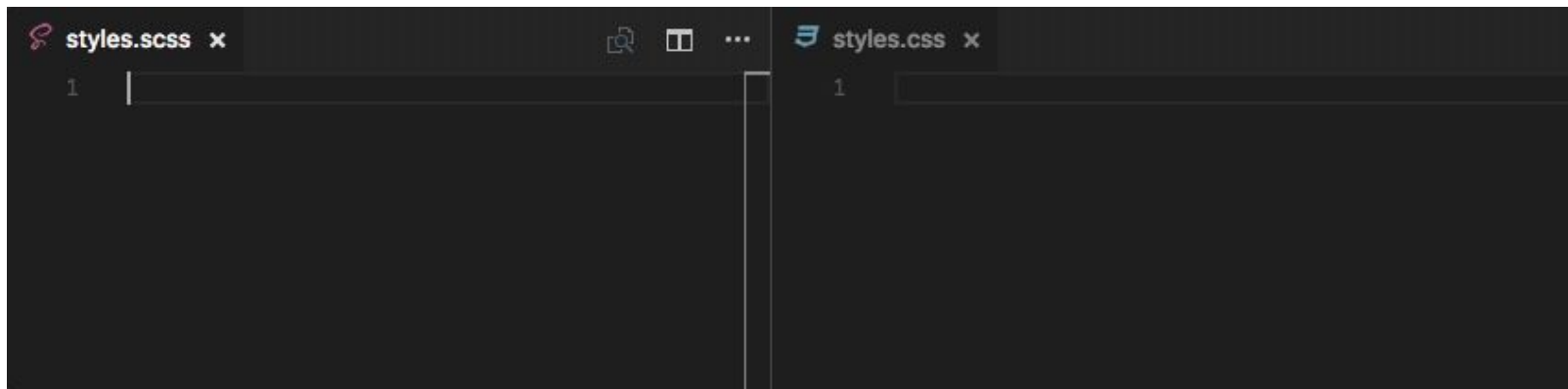
En la sección de scripts, añade un comando scss debajo del comando test, como se muestra abajo:

```
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1",  
  "build-css": "node-sass --include-path scss scss/style.scss css/main.css",  
  "watch-css": "nodemon -e scss -x \"npm run build-css\""
```

```
"build-css": "node-sass --include-path scss scss/style.scss css/main.css",  
"watch-css": "nodemon -e scss -x \"npm run build-css\""
```

COMANDAR PARA COMPILAR

Para ejecutar nuestro script de una línea, necesitamos ejecutar el siguiente comando en la terminal: **\$ npm run watch-css**



Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE

NESTING, IMPORT Y VARS



NESTING O ANIDACIÓN

HTML sigue una estricta estructura de anidación, mientras que CSS, por lo general, es un caos total.

Con la anidación de SASS, puedes organizar tu hoja de estilo de una manera que se asemeja a la de HTML, lo que reduce la posibilidad de conflictos en el CSS.

NESTING O ANIDACIÓN

SCSS

```
ul {  
  list-style: none;  
  li {  
    padding: 15px;  
    display: inline-block;  
    a {  
      text-decoration: none;  
      font-size: 16px;  
      color: #444;  
    }  
  }  
}
```



CSS

```
ul {  
  list-style: none;  
}  
ul li {  
  padding: 15px;  
  display: inline-block;  
}  
ul li a {  
  text-decoration: none;  
  font-size: 16px;  
  color: #444;  
}
```

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE

IMPORT

Una de las características más útiles de SASS es la posibilidad de separar tus hojas de estilo en archivos separados. A continuación, puedes usar *@import* para incluir la fuente de tus archivos individuales en una hoja de estilo maestra.

IMPORT

```
@import "estructura";  
@import "colores";  
@import "tipografia";  
@import "grilla";
```

Ejemplo: quieres tener por separado los estilos donde nos enfocamos en la estructura, colores, tipografía y grilla.

¡importante! el archivo debe tener “_” (guión bajo) al principio del nombre. Ej:
_colores.scss.

¿CÓMO ESTRUCTURAR LOS PROYECTOS SASS?

Si deseas llevar un orden tu proyecto, puedes seguir esta [estructura](#).

¿Hay una forma estándar de separar tus archivos CSS?

No, dependerá de los frameworks que uses.

VARs (VARIABLES)

Las variables son una manera de **guardar información que necesites reutilizar en tus hojas de estilos**: colores, dimensiones, fuentes o cualquier otro valor. SASS utiliza el símbolo dólar (\$) al principio de la palabra clave para crear una variable.

Estas variables se comportan como atributos CSS, y su valor puede ser cualquiera que pudiera adquirir un atributo CSS.


```
/* Variables */  
$title-font: normal 24px/1.5 'Open Sans', sans-serif;  
$cool-red: #F44336;  
$box-shadow-bottom-only: 0 2px 1px 0 rgba(0, 0, 0,  
0.2);  
  
/* SCSS */  
h1.title {  
    font: $title-font; /* Uso la variable */  
    color: $cool-red;  
}  
  
div.container {  
    color: $cool-red;  
    background: #fff;  
    width: 100%;  
    box-shadow: $box-shadow-bottom-only;  
}
```

SCSS

```
h1.title {  
    font: normal 24px/1.5 "Open Sans",  
    sans-serif;  
    color: #F44336;  
}  
  
div.container {  
    color: #F44336;  
    background: #fff;  
    width: 100%;  
    box-shadow: 0 2px 1px 0 rgba(0, 0, 0, 0.2);  
}
```

CSS

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE

VARs (VARIABLES)

Una variable se podrá definir fuera o dentro de algún selector.

- Si se define por fuera, dicha variable será global.
- Si se define por dentro de un selector, será local.

Una buena práctica común consiste en definir todas las variables globales al principio del fichero, para que puedan localizarse rápidamente.

USO DE !DEFAULT EN LAS VARIABLES

Si haces esto, el color que tomará será #000000:

```
$color: #FF0000;  
$color: #000000;
```

Pero si agregamos el **!default**, se tomará el #333333. El **!default** indica que si dicha propiedad no fue asignada, tome el #000000 por defecto:

```
$color: #333333;  
$color: #000000 !default;
```

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE

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 cubrir la **metodología BEM**, que nos ayudará a entender estructuras de CSS, y a mejorar las nuestras.

BEM

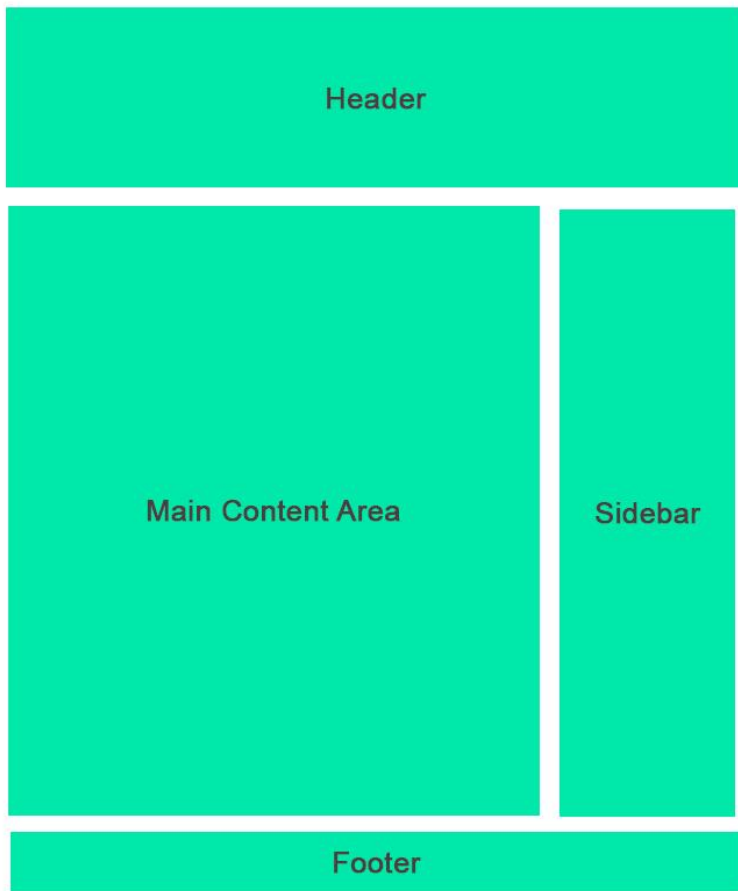
BEM significa *Modificador de Bloques de Elementos* (*Block Element Modifier*) por sus siglas en inglés. Sugiere una manera estructurada de nombrar tus clases, basada en las propiedades del elemento en cuestión.

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

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. Miremos la imagen a continuación:

BLOQUE



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

La doble barra baja 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

HTML

```
11 <section>
12 |   <article class="noticia">
13 |     <!--Bloque contenedor-->
14 |   </article>
15 </section>
```

CSS

```
1 .noticia{
2 |   background:  lightgray;
3 }
```

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).

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 {}
```

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE

ELEMENTOS

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

CSS

```
5  .noticia__titulo{  
6      font-family: 'Times New Roman', serif;  
7  }
```

HTML

```
11  <section>  
12      <article class="noticia">  
13          <h1 class="noticia__titulo">Título de la noticia</h1>  
14      </article>  
15  </section>
```



¡NUNCA USES SÓLO MODIFICADORES!

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

Esto está mal 👎

```
<button class="button-primary"></button>
```

```
<button class="menu__button-primary"></button>
```

Esto está bien 👍

```
<button class="button button-primary"></button>
```

```
<button class="menu__button menu__button-primary"></button>
```

MODIFICADORES

CSS

```
9  .noticia--destacada{
10  |     background: dimgray;
11  | }
12  .noticia__titulo--uppercase{
13  |     text-transform: uppercase;
14  | }
```

HTML

```
11  <section>
12  |     <article class="noticia--destacada">
13  |         <h1 class="noticia__titulo--uppercase">Título de la noticia</h1>
14  |     </article>
15  </section>
```

Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

CODER HOUSE

BEM



Ventajas

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



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 Bootstrap y queremos modificar ciertas clases.
- Cuando trabajamos en equipo y cada miembro tiene una manera distinta de escribir CSS



APLICANDO SASS

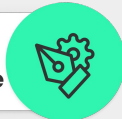
Crea el archivo SCSS de sus proyectos y compilarlos.

APLICANDO SASS

Formato: archivo HTML, CSS y SCSS. Debe tener el nombre “Idea+Apellido”.

Sugerencia: carpeta en formato zip o rar, con el/los archivos HTML, CSS y SCSS.

Desafío
entregable



>> Consigna: crea el archivo SCSS de sus proyectos y compilarlos. Además, deberás crear al menos cuatro (4) bloques con la metodología de BEM.

>>Aspectos a incluir en el entregable:

- Instala SASS y crea el archivo SCSS para compilarlo en CSS. Envía ambos archivos como parte del desafío.
- Crear al menos cuatro (4) bloques con la metodología de BEM.

>>Ejemplo:

[Carpeta comprimida con BEM](#) / [Carpeta comprimida con SASS](#)



APLICANDO SASS I

APLICANDO SASS I

Formato: Archivo html y css

Sugerencia: carpeta en formato zip o rar con el/los archivos html y CSS.

Desafío
Complementario



>> Consigna:

1. Crear el archivo SCSS con un estructura de archivos adaptada a sus proyectos y compilarlos.
2. Crear al menos ocho (8) bloques con la metodología de BEM.

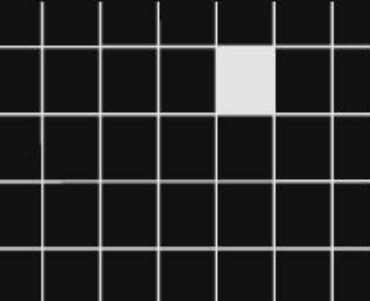
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Aplicación de SASS.
 - Conocer BEM.
- 



OPINA Y VALORA ESTA CLASE