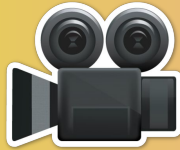


***RECUERDA PONER A GRABAR LA  
CLASE***





***¿DUDAS DEL ON-BOARDING?***

**MIRALO AQUI**



**Clase 13.** DESARROLLO WEB

***SASS /***



## ***OBJETIVOS DE LA CLASE***

- Conocer SASS.
- Aplicar SASS.

# ***GLOSARIO:***

## ***Clase 12***

**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:** 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 13

¡Para  
recordar!



# ***CRONOGRAMA DEL CURSO***

## Clase 12



### **Bootstrap + Themes**



PRÁCTICAS DE LO  
VISTO EN CLASE



SEGUNDA ENTREGA  
PROYECTO FINAL

## Clase 13



### **SASS I**



PRÁCTICAS DE LO  
VISTO EN CLASE



SASS I

## Clase 14



### **SASS II**



PRÁCTICAS DE LO  
VISTO EN CLASE





# ***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***





# ***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 así:

```
{ 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

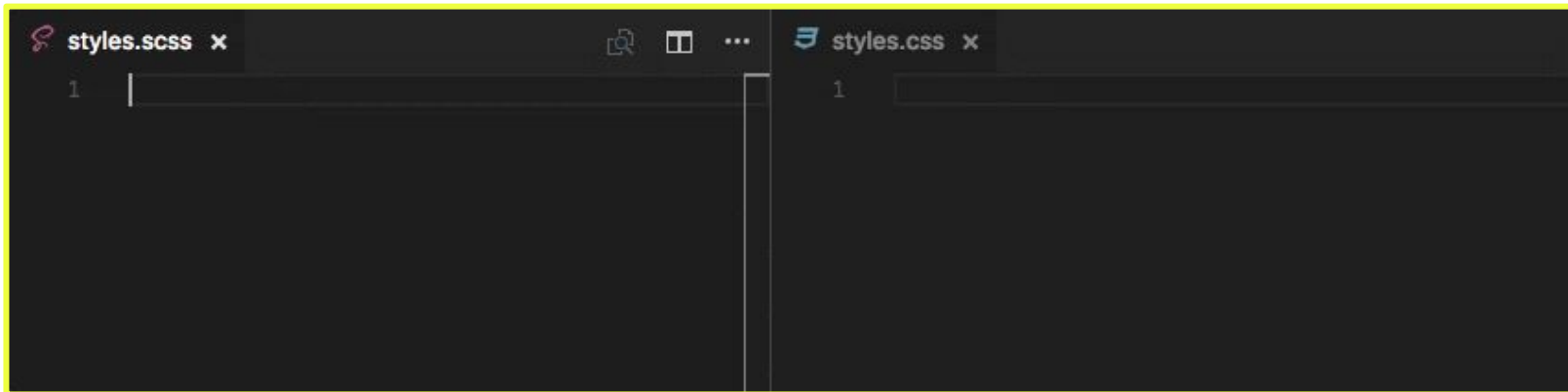
En la sección de scripts, añade un comando scss debajo del comando test, como se muestra abajo:

```
"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;  
}
```

# ***USO DEL & CON EL NESTING***

El **&**, es un selector especial inventado por Sass que se usa en selectores anidados para referirse al selector externo. Hace posible reutilizar el selector externo de formas más complejas, como agregar una pseudoclase o agregar un selector antes del padre.

# USO DEL & CON EL NESTING

**SCSS**

```
li {  
  color:blue;  
  &:hover{  
    color:red;  
  }  
}
```



**CSS**

```
li {  
  color:blue;  
}  
li:hover {  
  color:red;  
}
```

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***

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

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



***BREAK***

**¡5/10 MINUTOS Y VOLVEMOS!**



# ¿PREGUNTAS?

#CoderTip: Ingresa al [siguiente link](#) y revisa el material interactivo que preparamos sobre **Preguntas Frecuentes**, estamos seguros de que allí encontrarás algunas respuestas.

Ejemplo  
en vivo



***¡VAMOS A PRACTICAR LO VISTO!***



# ***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.

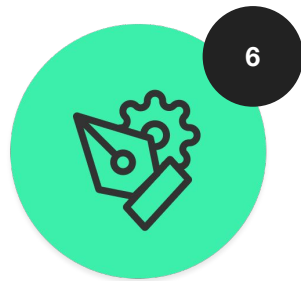
Ejemplo  
en vivo



***¡VAMOS A PRACTICAR LO VISTO!***

***CODER HOUSE***





## ***APLICANDO SASS***

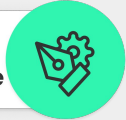
Configurar nuestro proyecto para que soporte SASS. Aplicar SASS en una página a elección. Cargar todo en github.

# 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:** Configurar nuestro proyecto para que soporte SASS. Aplicar los conceptos vistos durante la clase de SASS I en una página a elección (podes modificar la sintaxis de tu código CSS hecho previamente y adaptarlo a las reglas de SASS). Cargar todo en github.

**>>Aspectos a incluir en el entregable:**

- Instala SASS y crea el archivo SCSS para compilarlo en CSS. Envía ambos archivos como parte del desafío.
- Aplicar algún concepto que incorpora SASS: generar los partials a partir de la idea de componentes, usar variables para aplicarla como valor a una propiedad y/o usar el selector & dentro de un nesting.

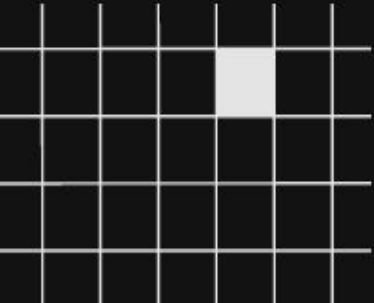
***¿PREGUNTAS?***





***¡MUCHAS GRACIAS!***

Resumen de lo visto en clase hoy:

- Aplicación de SASS.
  - Conocer BEM.
- 



***OPINA Y VALORA ESTA CLASE***

***#DEMOCRATIZANDO LA EDUCACIÓN***

***CODER HOUSE***