Sass: Guía de Dictado de Clases

Documentos provistos por Digital House para el armado del temario:

[Organización de Temas](https://docs.google.com/spreadsheets/d/1h-Y5ef48OejX1mTAbysS67fuRNP9RY1fIWxVdCOM618/edit?ts=5734eef5#gid=762641498)

[Draft General de Sass](https://docs.google.com/spreadsheets/d/1PHi9vOxzjoc5_AX_Xul1Mj1gYT4vvlvS2A8cvrdSjR0/edit?ts=5734eeef#gid=387944668)

# Clase 1 de 2:

[PPT 1a](https://drive.google.com/open?id=1surHDMoQtiEKWLpJ6QvQHFHKJ2uPqyAYyocA74WTRsY)

## Introducción a los Precompiladores

Definición de “Precompilador”: En Informática, un preprocesador o precompilador (o preprocessor en inglés) es un programa que procesa datos a la entrada y produce datos en la salida que serán usados como entrada de datos para otro programa. ([Wikipedia](https://en.wikipedia.org/wiki/Preprocessor): a **preprocessor** is a [program](https://en.wikipedia.org/wiki/Computer_program) that processes its input data to produce output that is used as input to another program.)

No es mucho más que un software que actúa como intermediario. Su función es la de procesar unos datos de entrada y lanzarlos de manera que puedan ser entendibles desde otro programa.

Al hablar de precompiladores CSS, nos referimos a programas que utilizan lenguajes de programación complejos ofreciendo la posibilidad de realizar código CSS con muchas más posibilidades. Algunas muestras de porque permiten realizar un mejor código CSS, son:

* Permite herencia y POO.
* Permiten el uso de variables.
* Permiten realizar operaciones.
* Se realizan comentarios de código automáticos.
* Se organiza de forma estructurada y automática el código.
* Permiten el uso de funciones, estructuras de control, bucles…
* Facilitan el concepto de desarrollo DRY

No persiguen otro objetivo que no sea el de optimizar el lenguaje y su tiempo de desarrollo.

No lo confundan con los Transpiladores. Sin ahondar en este tema, sepan que un transpilador es una forma de compilar.

**Compilar** es a nivel general, tomar un código fuente y transformarlo en otro.

**Transpilar** es más específico en su definición, y se refiere al proceso de tomar un código fuente escrito en una lenguaje y lo transforma en otro lenguaje que tiene un nivel similar de abstracción. a specific term for taking source code written in one language and transforming into another language that has a similar level of abstraction.

***NOTA:*** *En el caso de SASS, es un preprocesador para CSS. Pero antes de seguir adentrándonos, está bueno mostrar cómo este concepto aplica a HTML, JS o PHP, y sugerir a los alumnos que piensen cómo llevarían ese mismo concepto a CSS. Una vez presentado este concepto aplicado a otros lenguajes, la home de* [*Stylus*](http://stylus-lang.com/) *tiene un excelente formato de presentación de los features que puede ayudarnos a entender mejor qué es lo que proponemos con esto de los compiladores para CSS-*

Veamos cómo este concepto aplica a HTML. Uno de los preprocesadores más conocidos es [Haml](http://haml.info/).

Source Code: SASS Language -> Preprocesador -> CSS

Veamos las diferencias de SCSS y SASS. Básicamente se diferencian en la sintaxis con la que se escriben. El más popular es Sassy, dado que no tiene mucho sentido tener restricciones de tabulación. Tampoco se optimiza el tiempo de trabajo por evitar escribir corchetes o puntos y comas, sobretodo porque hoy los IDEs son muy completos y ágiles en este sentido, y porque no es crítico impedir la compilación de los archivos por temas relacionados a la tabulación. Sin embargo, no le restemos importancia a este tema en particular: puede ser útil en procesos en los que un build puede romperse por errores de tipeo. Podemos explicar un poco sobre más sobre los problemas que se pueden tener en este aspecto en aplicaciones grandes como ML, y cómo podemos solucionarlos en Sassy empleando comprobaciones en otras partes del proceso usando Task Runners como Gulp.

Como primer ejercicio, vamos a ver un poco lo que pasa con el código en SCSS y en SASS. Podemos pedirle a los alumnos que agreguen algún elemento más en la vista y que le apliquen estilos escribiendo en Sass y compilándolo y viendo el resultado.

[Ejercicio de Sassy vs. SCSS](https://github.com/bastiantowers/sass-dh/tree/master/Clase%201/ejercicios/sass_sassy)

***NOTA:*** *En nuestra primera ejercitación, estaría bueno tomar algo en CSS que los chicos hayan hecho, que sea cortito y simple, y que podamos reescribirlo en SCSS.*

## Introducción a la sintaxis del código

### Variables

[PPT 1b](https://docs.google.com/presentation/d/1AA7tHBMsPHE9MbMv6vUg1R6D6qUfMXp6e8thfOF3bzY/edit#slide=id.p5)

Piensen en las variables como en cualquier otro código: en este caso, lo usamos para guardar valores que podremos reutilizar en todas nuestras hojas de estilos. Están pensados para guardar datos como colores, tamaños de fuentes, o cualquier otro valor que piensen que van a reutilizar, y que pueda ser instanciado desde nuestras hojas de estilos. Se usa el símbolo $ para declarar variables. Las variables pueden declararse sin inicializar valores.

### Scope de las Variables

Existen las variables globales y las variables locales. Todas las variables que se definen por fuera de una clase o función, se consideran variables globales y son accesibles desde cualquier parte del código.

Link copado para entender mejor las funciones que trabajan con colores:

<http://jackiebalzer.com/color>

# Clase 2 de 2

En general el material está bastante bien guionado en las PPTs. Básicamente, se tratan los temas más internos del lenguaje, como los mixins, los condicionales, etc.

Es importante que no olvidemos transmitir nuestra experiencia sobre estos temas. Introduzcamos las metodologías [BEM](https://en.bem.info/) o similares, para que los alumnos tengan en consideración el uso racional y medido de estas potencialidades de SASS. Mencionemos también que es mejor añadir autoprefixers desde un task runner, en lugar de hacerlo con mixins.

***Importante:*** *según cómo vengamos de tiempo, el PPT “Mixins Útiles” podríamos obviarlo. Quizás una introducción suficiente a este tema sea mencionar las librerías conocidas de Bourbon o Andy, o comentar que Bootstrap incluye su propio set de mixins, y sugerir que investiguen este tema aparte. Los mixins pueden ser muy generales, como muy personales o específicos, por lo que podemos invertir el tiempo en clase en algo más antes que en esto.*