MODULE 01-002 : SCSS. Who, What, When, Where, Why, and How

Welcome to the **first lesson** in the *Getting DRY with SCSS* course!
In this guide, we'll explore **the who, what, when, where, why, and how** of SCSS.
Let's dive in and uncover what makes SCSS a game-changer in modern web design. 🚀

👤 Who

SCSS was created by Hampton Catlin, a well-known figure in the Rails community.
Hampton also created:

- The **Hammell templating library** 🛠️
- The **Mobile Wikipedia version** 🌐

His work has likely influenced tools you already use without realizing it!

❓ What

SCSS stands for *Sassy Cascading Style Sheets*.

- **Age**: Over 10 years old 🎂
- **Evolution**: Has received powerful upgrades over the years.

📍 Where

You can use SCSS **anywhere**! 🌍
While initially built for Rails, SCSS is now:

- Compatible with **mobile frameworks** 📱
- Compatible with **JavaScript front ends** 💻
- Fully **stack-agnostic**—use it with any system!

In this course, we'll use SCSS with a **separate system** to show how flexible it is.

💡 Why

Why learn SCSS? Here's why it's worth your time:

1. **Streamlines your design process** 🎯

    - Reduces repetition by enabling shared code.
2. **Makes styles scalable** 📈
    - Tools like **mixins**, **variables**, and **conditionals** make your stylesheets efficient and dynamic.

✨ SCSS Benefits:

- **Mixins**: Bundle functionality into one element and reuse it across your system.
- **Conditionals**: Dynamically adapt styles based on conditions.
- **Variables**: Define colors, sizes, and more in one place for easy updates.
  Example: Change a color once, and it updates everywhere.

SCSS = DRY code + best practices. 🏆

⚙️ How

SCSS works as a **preprocessor**:

- It lets you write enhanced CSS with added functionality.
- Converts SCSS code into standard CSS.
- SCSS takes **mixins, variables, functions, and conditionals**, compiles them, and generates CSS ready for browsers.

Video Speech.

[ENG]
Welcome to the first lesson in this getting DRY with SCSS course.
In this guide what we're going to discuss are the who, what, when, where, why and how and of what SCSS is.
***
By getting some background and learning about who created the framework, that gives us a little bit of insight on the problem that they were trying to solve, if they were successful in doing it, and also if that's going to help you as you try to integrate and learn that system.

Who

SCSS was **created by Hampton Catlin**.
Hampton is somewhat of a celebrity in the Rails community because **he has created some incredibly popular libraries**. Things that you may have already been using and didn't even realize it.
In addition to SCSS, he also created **the Hammell templating library** and **Mobile Wikipedia version**.

What

Moving on to the what. SCSS is short for **Sassy Cascading Style Sheets** and it was **created over 10 years ago**.
It's gone through some incredible upgrades that have allowed us to use it in pretty powerful ways.

Where

That leads directly into the where. Where can we use SCSS? That's something that's really neat about how it was built, even though early on it was something that was considered to be **a Rails library**.
It now can be used in pretty much any kind of framework that you have.

Whether **it's a mobile framework to a javascript front end**, it's not connected or locked into working with one type of stack or another**.
And, in fact, when we're going through this course we're going to use a completely separate system in order to build our stylesheets just so you can see that it's completely agnostic to whatever stack you happen to be working on.

Why

So far we've covered the who, the what, and the where. And we're going to get to my favorite one which is the why.
Why is it important and why is it helpful to learn about SCSS?
That is something that I try to impress on any student when they're coming through the program and they want to get into more advanced topics when it's coming to front end design.

SCSS allows us to be able to streamline our entire design implementation process.
So if up to this point you've always written just in pure CSS that's perfectly fine.
It is a great place to start but what SCSS allows you to do is to streamline that process:

- **Everything that SCSS does revolves around being able to take out any kind of repetitive processes.**

So, if you find yourself writing the same type of code snippets or even very similar types of CSS styles on a regular basis, SCSS allows you to stop all that repetition and start using shared code.

Throughout this course, you're going to see how pretty much all of the **tools that are built into SCSS are giving you the ability to make all of your style sheets much more scalable.**

Everything from:
- **Being able to build mixins, where you can load a bunch of functionality inside of one type of element**
- **Then share it across your entire system**
- **To using conditionals, to be able to check if one style is what you want in one situation and it's something you wanted in a different situation** those styles can actually change dynamically.

Which is something that before this, we had to use things such as implementing code inside of our views or building view helpers in order to create that same type of feature.
Now that's all available directly in the SCSS file it works like a programming language.

In addition to tools like mixins, **SCSS also gives us the concept of variables**:
- If you've ever had a situation where you had a CSS style sheet and you had some color definition in 20 spots and then decided you maybe wanted to change the color just a little bit.
  You would need to change that color in every single location that you called it.

- With an SCSS variable, you can define all of your styles, colors, and sizes at the top and then reference those throughout the application. So when you want to make that change, you make it in one spot and it auto-populates every place else.
  Like I said everything in SCSS revolves around being able to create shared code that is much drier and follows best practices.

How

So how is all that possible? That leads directly to the last question we're going to answer in this guide, which is the how.

```
What SCSS does is it allows you to write CSS typestyles. It looks a lot like
CSS with some nice layers on top.
Technically SCSS is what's called a preprocessor. Which means that it's really
just a type of scripting language that the browser can't interpret.

As that preprocessor runs it takes all of the code that you wrote, all of those
SCSS styles, and converts it directly into CSS.
It takes all of the different efficiencies, the variables, the mixins, and the
functions, the conditionals, and then it builds your own CSS files directly for
you.

In the next guide, we're going to go through a screencast demo where you can
see exactly how that process unfolds.
```

[SPA]

Bienvenido a la primera lección de este curso de obtener drai con SCSS.
En esta guía vamos a discutir el quién, qué, cuándo, dónde, por qué y cómo y de qué es SCSS.

Al obtener algunos antecedentes y aprender sobre quién creó el marco, eso nos da un poco de información sobre el problema que estaban tratando de resolver, si tuvieron éxito al hacerlo y también si eso lo ayudará a medida que intenta integrar y aprender ese sistema.

¿Quién?

SCSS fue creado por Hampton Catlin.
Hampton es una especie de celebridad en la comunidad de Rails porque ha creado algunas bibliotecas increíblemente populares. Cosas que quizás ya hayas estado usando y ni siquiera te hayas dado cuenta.
Además de SCSS, también creó la biblioteca de plantillas Hammell y la versión móvil de Wikipedia.

¿Qué?

Pasando al qué. SCSS es la abreviatura de Sassy Cascading Style Sheets y fue creado hace más de 10 años.
Ha pasado por algunas actualizaciones increíbles que nos han permitido usarlo de maneras bastante poderosas.

¿Dónde?

Eso nos lleva directamente al dónde. ¿Dónde podemos usar SCSS? Eso es algo realmente interesante de cómo se construyó, aunque al principio se considerara una biblioteca de Rails.
Ahora se puede usar en prácticamente cualquier tipo de marco que tenga.

Ya sea un marco móvil para un front-end de JavaScript, no está conectado ni bloqueado para trabajar con un tipo de pila u otro.
Y, de hecho, cuando recorramos este curso, usaremos un sistema completamente separado para construir nuestras hojas de estilo solo para que pueda ver que es completamente independiente de cualquier pila con la que esté trabajando.

¿Por qué?

Hasta ahora hemos cubierto el quién, el qué y el dónde. Y vamos a llegar a mi favorito que es el por qué.
¿Por qué es importante y por qué es útil aprender sobre SCSS?
Eso es algo que trato de impresionar a cualquier estudiante cuando vienen a través del programa y quieren entrar en temas más avanzados cuando se trata de diseño de front-end.

SCSS nos permite optimizar todo nuestro proceso de implementación de diseño.
Entonces, si hasta este punto siempre has escrito solo en CSS puro, está perfectamente bien.

Es un excelente lugar para comenzar, pero lo que SCSS le permite hacer es optimizar ese proceso:

    Todo lo que hace SCSS gira en torno a poder eliminar cualquier tipo de proceso repetitivo.

Entonces, si te encuentras escribiendo los mismos tipos de fragmentos de código o incluso tipos muy similares de estilos CSS de forma regular, SCSS te permite detener toda esa repetición y comenzar a usar código compartido.

A lo largo de este curso, verá cómo casi todas las herramientas integradas en SCSS le brindan la capacidad de hacer que todas sus hojas de estilo sean mucho más escalables.

Todo desde:

    Poder crear mixins, donde puedes cargar un montón de funcionalidad dentro de un tipo de elemento
    Luego compártelo en todo tu sistema
    Usar condicionales, para poder verificar si un estilo es lo que desea en una situación y es algo que deseaba en una situación diferente, esos estilos pueden cambiar dinámicamente.

Lo cual es algo que antes de esto, teníamos que usar cosas como implementar código dentro de nuestras vistas o crear ayudantes de vista para crear ese mismo tipo de función.
Ahora que todo está disponible directamente en el archivo SCSS, funciona como un lenguaje de programación.

Además de herramientas como mixins, SCSS también nos brinda el concepto de variables:

    Si alguna vez ha tenido una situación en la que tenía una hoja de estilo CSS y tenía alguna definición de color en 20 lugares y luego decidió que tal vez quería cambiar el color solo un poco.
    Tendría que cambiar ese color en cada ubicación en la que lo llamó.

    Con una variable SCSS, puede definir todos sus estilos, colores y tamaños en la parte superior y luego hacer referencia a ellos en toda la aplicación. Entonces, cuando desee realizar ese cambio, lo realiza en un solo lugar y se rellena automáticamente en todos los demás lugares.
    Como dije, todo en SCSS gira en torno a poder crear código compartido que sea mucho más seco y siga las mejores prácticas.

¿Cómo?

Entonces, ¿cómo es posible todo eso? Eso nos lleva directamente a la última pregunta que vamos a responder en esta guía, que es el cómo.
Lo que hace SCSS es permitirle escribir estilos de CSS. Se parece mucho a CSS con algunas capas agradables en la parte superior.
Técnicamente, SCSS es lo que se llama un preprocesador. Lo que significa que es

realmente solo un tipo de lenguaje de script que el navegador no puede interpretar.

A medida que se ejecuta ese preprocesador, toma todo el código que escribió, todos esos estilos SCSS y lo convierte directamente en CSS.
Toma todas las diferentes eficiencias, las variables, los mixins y las funciones, las condicionales y luego crea sus propios archivos CSS directamente para usted.

En la próxima guía, veremos una demostración de pantalla donde podrá ver exactamente cómo se desarrolla ese proceso.