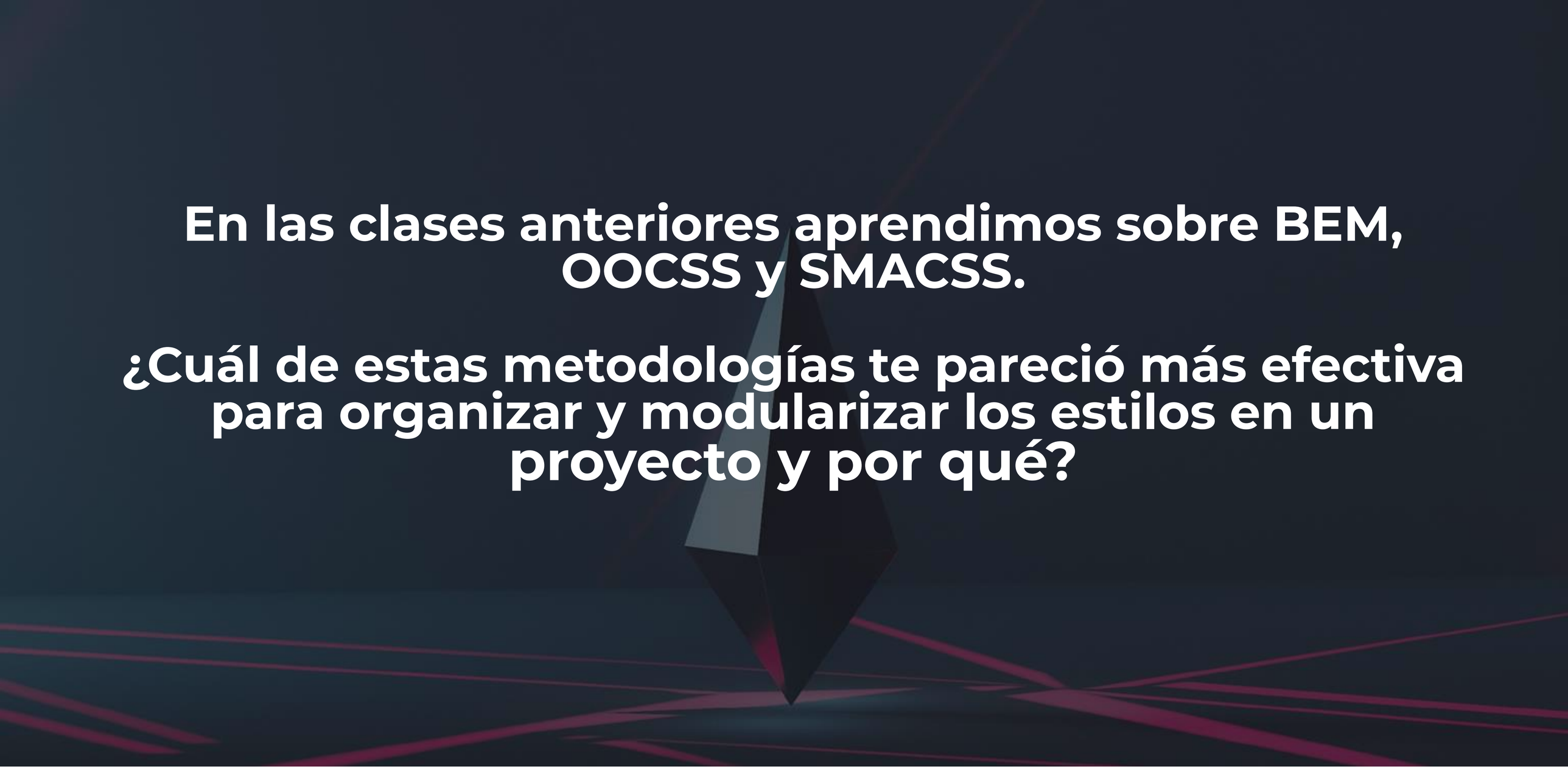
IT Academy

by KIBERNUM

Sass: Organización y Modularización de Estilos


En las clases anteriores aprendimos sobre BEM, OOCSS y SMACSS.
¿Cuál de estas metodologías te pareció más efectiva para organizar y modularizar los estilos en un proyecto y por qué?

Propósito de la clase

En esta sesión, exploraremos metodologías avanzadas para la organización y modularización de estilos en SASS. Aprenderás a estructurar y escribir código SASS de manera eficiente, escalable y mantenible, aprovechando técnicas como anidación, parciales, mixins, extends y el uso de variables. Al finalizar esta clase, serás capaz de:

✅ Comprender los conceptos fundamentales de SASS, incluyendo variables, anidación, mixins, extends y parciales.

✅ Diferenciar el uso de @extend, @mixin y @use, identificando cuándo y cómo aplicar cada uno en un proyecto real.

✅ Implementar un sistema de estilos escalable mediante SASS, organizando el código en parciales y estructurando los estilos de manera modular.

✅ Optimizar el código CSS reduciendo la redundancia mediante @extend, @mixin y @use, mejorando la mantenibilidad y la escalabilidad del proyecto.

Preguntas de Reflexión Iniciales:

Antes de entrar en materia, piensa en estas preguntas:

❓ ¿Has tenido que modificar código CSS sin saber si ese cambio afectará otros estilos en el proyecto?

❓ Si no has usado SASS, ¿cómo crees que el uso de variables y mixins podría ayudarte en tu código?

📌 Reflexiona sobre estos puntos, ya que aprenderemos a mejorar la organización del código CSS con SASS, evitando redundancias y asegurando un mantenimiento más eficiente. 🚀

¿Qué es Sass y por qué utilizarlo?

Sass (Syntactically Awesome Stylesheets) es un preprocesador CSS que extiende las capacidades de CSS, permitiendo escribir estilos de manera más modular, reutilizable y mantenible. Sass añade funcionalidades como variables, anidación, mixins, y más, facilitando la creación de estilos complejos de forma más sencilla y organizada.

Ventajas de usar Sass:

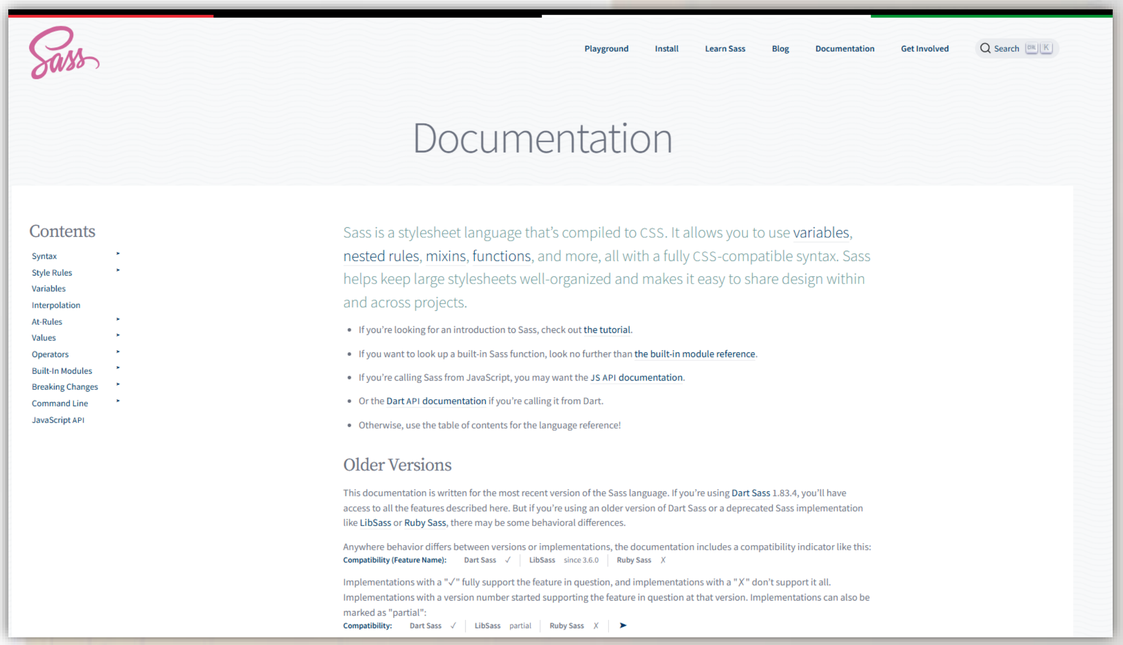
- **Modularización:** Divide tu CSS en archivos más pequeños y manejables.
- **Reutilización de código:** Utiliza variables, mixins y funciones para evitar la repetición.
- **Mantenibilidad:** Facilita la actualización y el mantenimiento de estilos.
- **Mejor organización:** Estructura tu CSS de manera lógica y coherente.
- **Funciones avanzadas:** Realiza cálculos, manipula colores y automatiza tareas.

Documentación de Sass

La documentación oficial de Sass está disponible en su sitio web: SASS. Allí encontrarás información detallada sobre las características de Sass, guías de inicio, referencias de funciones y módulos, así como tutoriales para aprovechar al máximo este preprocesador de CSS.

Consultar la documentación oficial de cualquier tecnología es fundamental para un desarrollador, ya que proporciona información precisa y actualizada directamente desde la fuente. Esto permite comprender a fondo las funcionalidades, evitar errores por información desactualizada y mejorar la eficiencia en el desarrollo al utilizar las mejores prácticas recomendadas por los propios creadores de la herramienta.

Además, la documentación oficial suele incluir ejemplos, explicaciones detalladas y casos de uso que facilitan la implementación adecuada en proyectos reales.



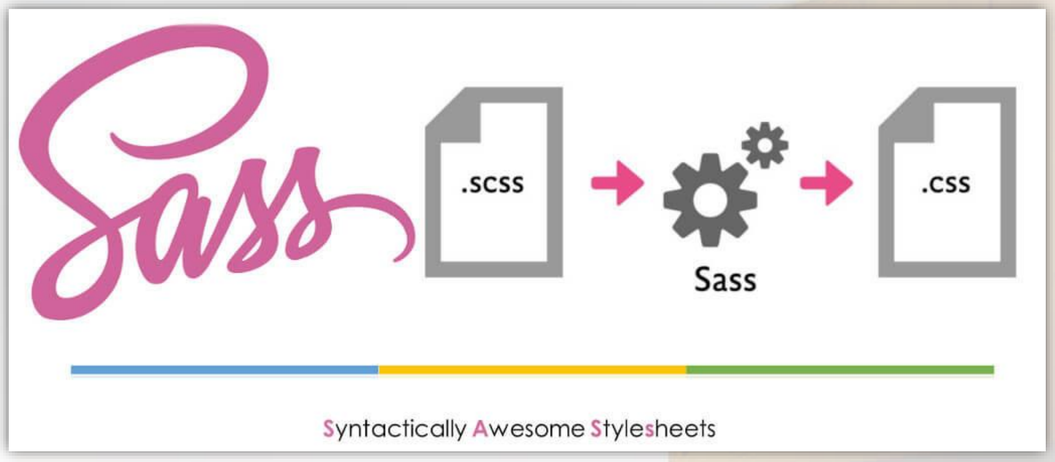
IT Academy
by KIBERNUM

¿Qué es un preprocesador y qué relación tiene con SASS? 🤔 💡

Un preprocesador CSS es una herramienta que extiende las capacidades del CSS tradicional, permitiéndonos escribir código más organizado, reutilizable y eficiente.

En lugar de escribir CSS directamente, usamos un preprocesador que luego convierte ese código en CSS estándar para que los navegadores lo entiendan.

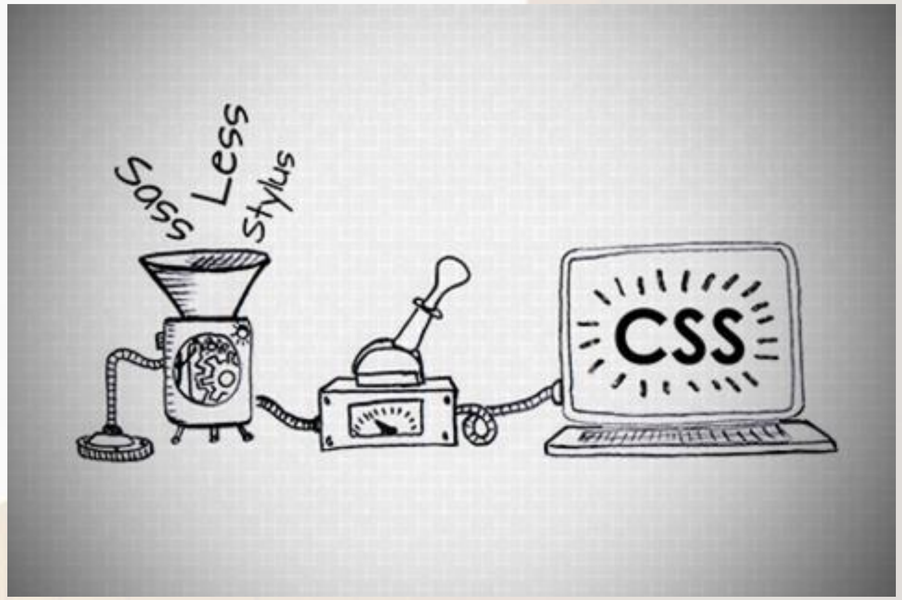
Uno de los preprocesadores más populares es SASS (Syntactically Awesome Stylesheets), que nos permite escribir estilos de manera más avanzada utilizando variables, anidación, mixins, herencia y más.



¿Cómo se relaciona SASS con los preprocesadores?

SASS es un preprocesador CSS porque nos permite:

- ✔ Usar variables para definir colores, tamaños y fuentes.
- ✔ Anidar estilos para mejorar la legibilidad.
- ✔ Reutilizar código con mixins y parciales.
- ✔ Evitar la repetición de código con herencia y extend.
- ✔ Optimizar el mantenimiento del CSS en proyectos grandes.



Fuente: https://www.acens.com/comunicacion/white-papers/preprocesadores-css-una-forma-diferente-de-crear-nuestras-hojas-de-estilos/

Sin un preprocesador como SASS, escribir CSS puede ser más repetitivo y difícil de escalar en proyectos grandes. Por eso, SASS simplifica y mejora la forma en que diseñamos sitios web. 🚀🎨


Creación de un Proyecto Sass

Instalación de Node y Sass

🔹 Paso 1: Instalar Node.js

✅ Si ya tienes Node.js instalado, sáltate este paso.

🛠️ Cómo instalar Node.js

- Ve a la página oficial de Node.js: 👉 https://nodejs.org/
- Descarga la versión LTS (recomendada).
- Instálalo como cualquier otro programa.
- Verifica la instalación abriendo una terminal y escribiendo:

```
node -v
```

Si te muestra un número de versión, ¡ya tienes Node.js instalado! 🎉

IT Academy
by KIBERNUM

Crear una carpeta para los proyectos

Para mantener el orden, crearemos una carpeta donde guardaremos todos los proyectos con SASS.

- Abre la terminal.
- Escribe este comando para crear la carpeta y entrar en ella:

```
mkdir proyectos-sass
```

```
cd proyectos-sass
```

(Puedes cambiar proyectos-sass por el nombre que prefieras)

- Ahora, cada vez que hagas un nuevo proyecto con SASS, créalo dentro de esta carpeta. 💾

Crear un proyecto nuevo

Dentro de la carpeta proyectos-sass, crea otra carpeta con el nombre de tu proyecto:

```
mkdir proyecto-1
```

```
cd proyecto-1
```

Inicializa un proyecto de Node.js con este comando:

```
npm init -y
```

Esto creará un archivo package.json, que es necesario para instalar dependencias.

🔷 Instalar SASS

Para poder usar SASS en nuestro proyecto, debemos instalarlo.

Ejecuta este comando en la terminal dentro de la carpeta del proyecto:

```
npm install sass -S
```

💡 Nota:

El -S significa que se guarda como una dependencia del proyecto (aunque en versiones nuevas de npm ya no es necesario).

🔷 Instalar PurgeCSS (Opcional)

PurgeCSS nos ayuda a eliminar CSS innecesario y reducir el tamaño del archivo final.

Para instalarlo, usa este comando:

```
npm install purgecss
```

💡 Nota:

Si no sabes si lo necesitas, puedes dejarlo para después. 🤓

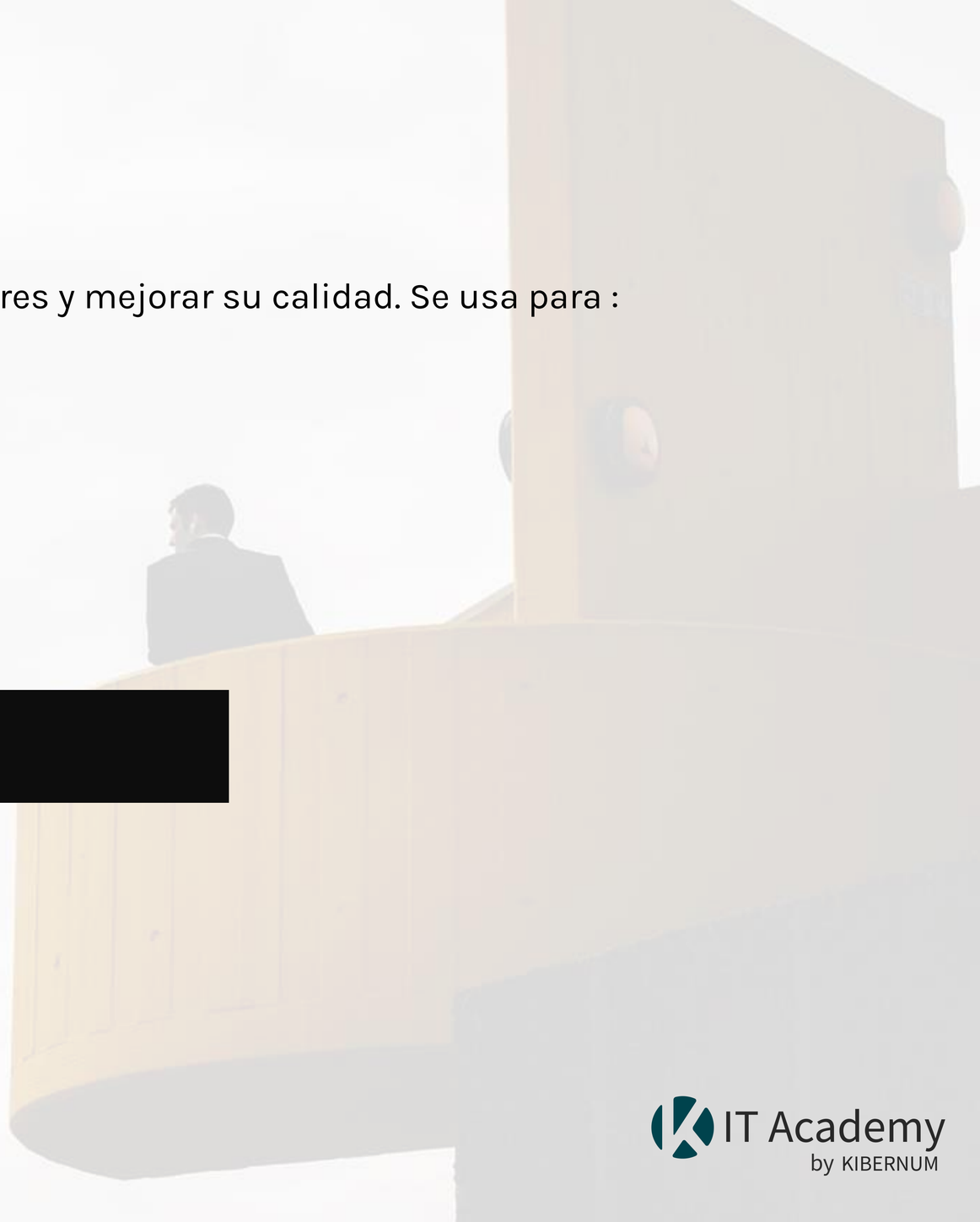
Instalar Linters

Un linter es una herramienta que analiza el código para detectar errores y mejorar su calidad. Se usa para :

- Prevenir errores comunes.
- Mantener un código limpio y ordenado.
- Aplicar buenas prácticas automáticamente.

Instalación de listers esenciales:
ESLint (JavaScript):

```
npm install -g eslint
npx eslint --init
```



Instalar Linters

Instalación de listers esenciales:

Stylelint (CSS y Sass):

```
npm install -g stylelint stylelint-config-standard
```

Prettier (formateo automático);

```
npm install -g prettier
```

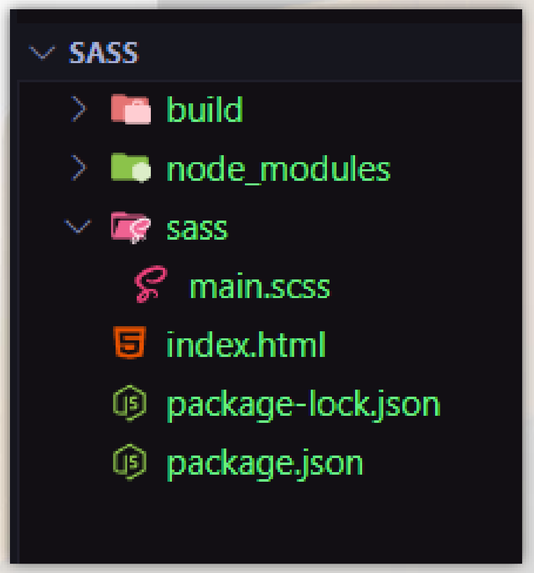
Para usarlos en VS Code, instala las extensiones ESLint, Stylelint y Prettier.

Estructura de carpetas

Aquí te muestro un ejemplo cómo debe quedar la estructura de archivos y carpetas un proyecto en sass:

```
proyecto-1
├── src
│   ├── scss
│   │   ├── main.scss
│   │   ├── _variables.scss
│   │   ├── _mixins.scss
│   │   ├── _base.scss
│   │   ├── _components.scss
│   │   ├── _layout.scss
│   │   └── _utilities.scss
├── build
│   ├── css
│   │   └── main.css
├── index.html
├── package.json
```

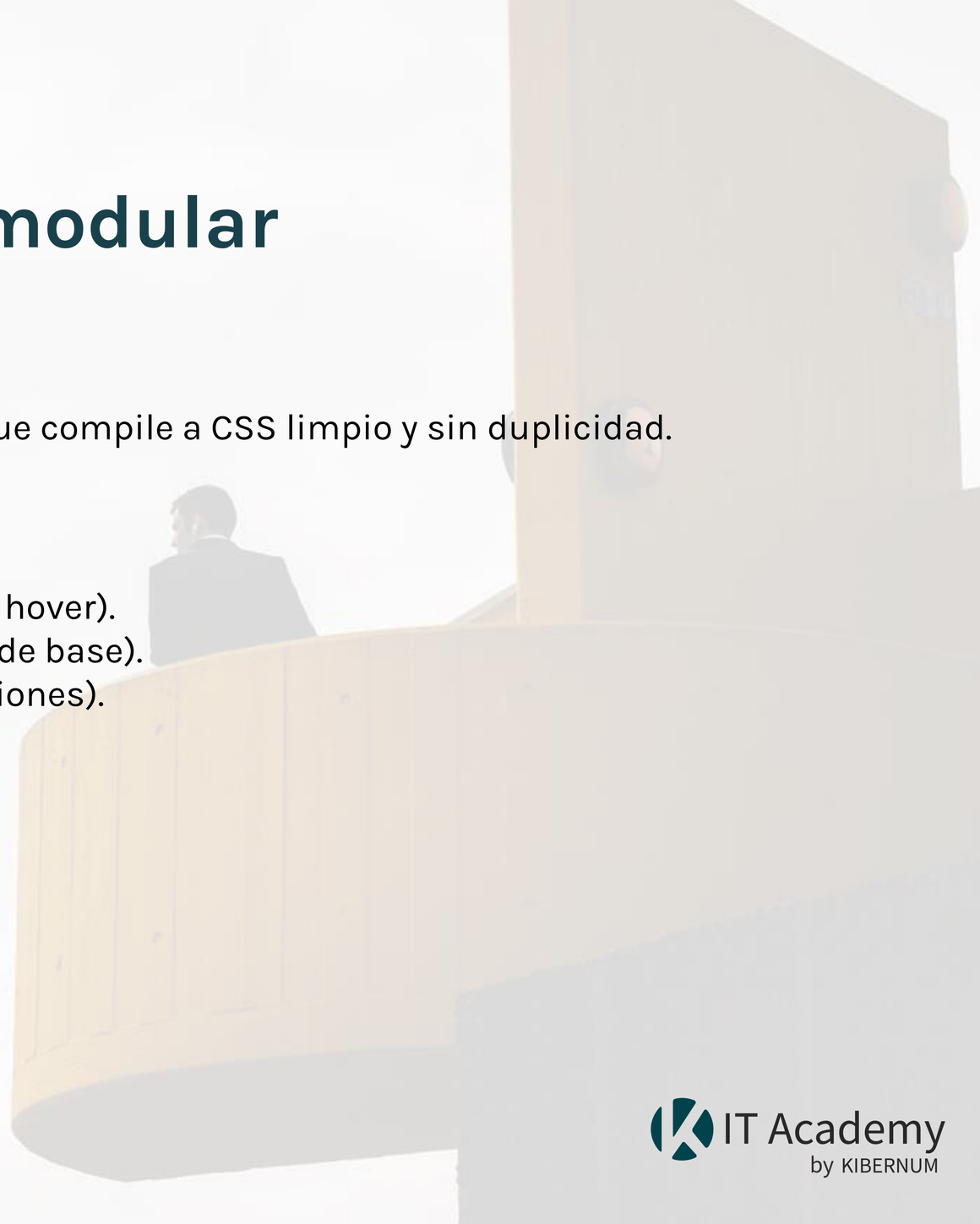
Comenzaremos desde lo más básico y progresaremos gradualmente hasta alcanzar un nivel más avanzado. Para ello, primero aprenderemos los fundamentos de Sass y su organización de archivos. Esta estructura inicial nos permitirá evolucionar de manera ordenada, incorporando progresivamente nuevos conceptos como componentes, funcionalidades y utilidades de Sass.



🔷 Reto: Aplica SASS de forma modular

Objetivo
Convertir estilos sueltos en un **paquete SASS organizado** que compile a CSS limpio y sin duplicidad.

Requisitos (checklist)
- Definir **variables** de colores y tipografías.
- Crear **1 @mixin** para botón (padding + tipografía + estado hover).
- Crear **1 @extend** para variantes (p. ej. .btn--primary extiende base).
- Usar **anidación** en un componente .card (título, texto, acciones).
- Separar en **parciales**: base/, components/, utilities/.
- Importar todo desde index.scss con **@use** (no @import).
- Compilar a dist/styles.css sin warnings.



IT Academy
by KIBERNUM

IT Academy

by KIBERNUM