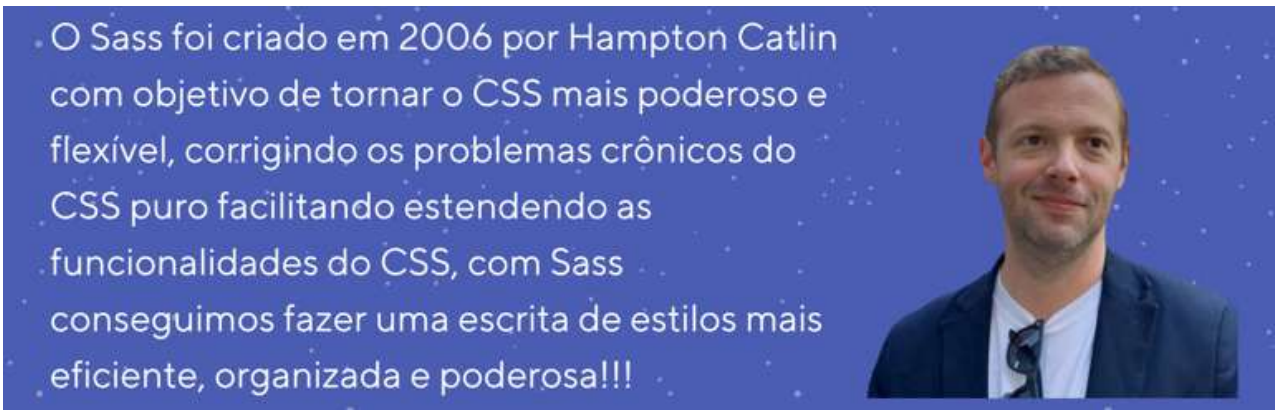
Introdução ao SaSS

Introdução ao SaSS

Quando criamos sites e aplicações web, usamos o **HTML** para estruturar o conteúdo, mas é com o **CSS** que deixamos tudo bonito e bem apresentado. No entanto, à medida que os projetos crescem, o CSS sozinho pode se tornar difícil de manter. É aí que entram ferramentas como o **SASS** e o **SCSS**, que ajudam a organizar e otimizar o nosso código de estilo.

Documentação do Sass ↗



Diferença entre CSS, SASS, SCSS

CSS (Cascading Style Sheets)

É a **linguagem padrão de estilos da web**, usada para estilizar páginas HTML. Com o CSS, conseguimos controlar:

- Cores
- Fontes
- Tamanhos
- Posições
- Layouts responsivos

Exemplo básico:

```
body {
  background-color: #f0f0f0;
  font-family: Arial, sans-serif;
}
```

Limitações do CSS puro:

- Não tem variáveis nativas (até recentemente) saiba mais: Utilizando propriedades CSS personalizadas (variáveis) ↗
- Não permite funções, loops ou reutilização avançada de código
- Pode se tornar repetitivo e difícil de manter em projetos grandes

SASS (Syntactically Awesome Stylesheets)

É um **pré-processador de CSS**, ou seja, uma ferramenta que adiciona recursos avançados ao CSS e depois **compila** o código em um CSS comum que os navegadores entendem.

O SASS nos permite usar:

- **Variáveis** (`$cor-primaria`)
- **Aninhamento de seletores**

- **Mixins** (blocos reutilizáveis)
- **Funções e loops**
- **Importações e organização por arquivos**

SCSS (Sassy CSS)

É uma **sintaxe alternativa do SASS**, mais parecida com o CSS tradicional.
Ou seja: **SCSS é SASS**, mas com uma sintaxe que usa chaves `{}` e ponto e vírgula `;` , como o CSS.

Diferença visual:

Sintaxe SASS (mais antiga):

```
$cor: blue
body
  background-color: $cor
```

Sintaxe SCSS (mais usada atualmente):

```
$cor: blue;

body {
  background-color: $cor;
}
```

Por que usar SASS/SCSS?

- Mais **organização**
- Mais **reutilização de código**
- Facilita a **manutenção em projetos grandes**
- Escrevemos menos e fazemos mais!

Qual dos dois utilizar?

- SASS: Sintaxe mais limpa e compacta, mas menos usada.
- SCSS: Sintaxe mais próxima do CSS, mais popular e compatível com arquivos CSS existentes.

Ambos, SASS e SCSS, fazem parte do mesmo pré-processador, mas oferecem uma escolha de sintaxe que pode se ajustar ao estilo de cada desenvolvedor.

Resumo das Diferenças

Linguagem	O que é?	Sintaxe	Usa chaves e ponto e vírgula?
CSS	Linguagem de estilo padrão	Tradicional	✅ Sim
SASS	Pré-processador do CSS	Mais enxuta, sem `{}` e `;`	❌ Não
SCSS	Versão do SASS parecida com CSS	Familiar para quem já usa CSS	✅ Sim

Instalação - Passo a Passo

Passo 1

Ao utilizar o Sass com HTML, devemos ter baixada a extensão Live Sass Compiler. Ela será responsável por processar o nosso código Sass para CSS puro, assim o navegador conseguirá entender o código transformado em CSS.

Passo 2

1. Comece criando os arquivos básicos para o seu projeto, um arquivo **.html** para o conteúdo e um arquivo **.scss** para os estilos.
2. **Abra o Visual Studio Code**, que é um editor de código popular e muito usado para desenvolvimento web.
3. No VS Code, **acesse a aba de extensões e busque por Live Sass Compiler**. Instale essa extensão, que facilita o processo de compilação de arquivos Sass para CSS.
4. No arquivo **.html**l, escreva o conteúdo da sua página, estruturando os elementos conforme necessário.

5. No arquivo **.scss**, escreva o código de estilo utilizando as funcionalidades do Sass, como aninhamento de regras.
6. Após instalar a extensão, você verá um botão na barra inferior do VS Code com a opção **"Watch Sass"**. Clique nele para ativar a compilação automática, o que fará com que o Sass seja transformado em CSS sempre que você salvar seu arquivo .scss.

Seletores

Assim como no CSS tradicional, o Sass (ou SCSS) também permite usar os seletores básicos para estilizar os elementos HTML:

O que é um seletor?

Um **seletor** é a **forma que usamos no CSS (ou no Sass)** para **"escolher" quais elementos HTML queremos estilizar**.

O **HTML cria** os elementos da página.
O **CSS escolhe (seleciona)** esses elementos para deixar mais bonitos, coloridos, com espaçamento, etc.

Seletor de Tag

Usa o nome da **tag HTML** diretamente, estiliza **todos os elementos** daquela tag na página.

```
body {
  background-color: #f9f9f9;
}

h1 {
  color: purple;
}
```

Seletor de Classe (class)

Usa o nome da classe com `.` (ponto), É o mais usado! Você pode aplicar a **mesma classe em vários elementos**.

```
.titulo {
  font-size: 24px;
  color: #333;
}
```

Exemplo no HTML:

```
<h1 class="titulo">Bem-vindo</h1>
<p class="titulo">Essa frase também usa a mesma classe.</p>
```

Seletor de ID

O ID **só deve ser usado uma vez por página**. Ele é **único**.
Exemplo no HTML:

Usa `#` (jogo da velha) seguido do nome do ID.

```
#principal {
  padding: 20px;
  background-color: lightgray;
}
```

Comparando:

Tipo	Símbolo	Exemplo Sass	Aplica onde?
Tag	*(sem símbolo)*	`p {}`	Em todas as tags `<p>` da página
Classe	`.`	`.box {}`	Em todos os elementos com `class="box"`
ID	`#`	`#menu {}`	Só no elemento com `id="menu"`

Variáveis

O que são variáveis no SCSS?

As **variáveis** são usadas no SCSS para **armazenar valores** que você pode reutilizar em várias partes do seu código. Isso ajuda a manter seu código mais limpo, organizado e fácil de manter.

Em vez de repetir a mesma cor, fonte ou tamanho várias vezes, você pode **guardar o valor em uma variável**.

Como declarar uma variável?

Essas variáveis podem ser usadas em qualquer lugar do seu arquivo `.scss`.

Exemplo prático

Vantagem: se quiser mudar a cor principal do seu site, você muda apenas **um valor**, e ele será atualizado em todos os lugares.

Dicas de boas práticas

- Nomeie as variáveis de forma clara e sem acentos: `$cor-titulo`, `$tamanho-botao`, etc.
- Guarde **cores, fontes, tamanhos, espaçamentos e sombras** em variáveis.

- Mude valores com segurança, sem quebrar o layout inteiro.

Aninhamento (nesting)

O que é o aninhamento no Sass?

Aninhamento (ou **nesting**) é a **possibilidade de escrever seletores "dentro" de outros seletores**, como se estivéssemos montando uma **árvore** de estilos, acompanhando a estrutura do HTML.

Isso **evita repetições** e deixa o código mais **organizado** e fácil de ler.

Exemplo simples (sem Sass):

```
.card {
  background-color: white;
}

.card h2 {
  color: purple;
}

.card p {
  font-size: 16px;
}
```

Veja como o nome `.card` precisa ser repetido várias vezes.

Agora com **Sass aninhado**:

```
.card {
  background-color: white;

  h2 {
    color: purple;
  }

  p {
    font-size: 16px;
  }
}
```

Muito mais organizado! E o resultado no CSS será o mesmo.

🧠 Dicas importantes

✅ Use aninhamento com **no máximo 2 ou 3 níveis**. Mais que isso pode deixar o código difícil de manter.
✅ Aninhar ajuda a **seguir a estrutura do HTML**, como se você estivesse "estilizando de dentro pra fora".

📌 Resumo

Vantagem do Sass (Nesting)	Por quê?
Organiza melhor o código	Acompanha a estrutura do HTML
Evita repetições	Não precisa repetir seletores longos
Melhora a leitura	Visualmente mais limpo e didático

Mixins

O que são Mixins?

Mixins são blocos de código reutilizáveis no SCSS.
Você pode pensar neles como "**funções de estilo**", que te ajudam a **evitar repetição** e **manter o código mais limpo**.

É muito útil quando você tem estilos repetidos em vários lugares, como botões, sombras, responsividade etc.

Como criar um Mixin?

Exemplo simples:

Para usar esse mixin em um elemento:

Mixins com parâmetros

Você pode deixar o mixin ainda mais **flexível** usando parâmetros.

Uso:

Por que usar Mixins?

- Reutilização de código
- Menos repetição
- Mais legível e organizado