



UNINASSAU
APRENDIZADO QUE FORMA LÍDERES

Minicurso de Sass/Scss ("Super" CSS)

Prof. Msc. Igor F. S. Revoredo



Agenda

- ❖ O que é SASS?
- ❖ Instalando SASS (no VSCode)
- ❖ Sintaxe!
- ❖ Primeiro contato
- ❖ Variáveis
- ❖ Nested Selectors
- ❖ Herança extend
- ❖ Import
- ❖ Mixin
- ❖ Melhorias
- ❖ Funções
- ❖ Minify
- ❖ Práticas





O que é SASS?

- ❖ Se você já tem algum conhecimento em web design provavelmente já ouviu sobre essa tecnologia,
- ❖ mas pode não conhecer a fundo sobre o que é exatamente o SASS é, o que faz e as maneiras como você deveria ou não utilizá-lo.





O que é SASS?

- ❖ O SASS é uma linguagem de extensão do CSS, a sigla significa “**Syntactically Awesome Style Sheets**” traduzindo ao pé da letra, **folhas de estilo com uma sintaxe incrível**.
- ❖ é uma linguagem de extensão SCSS que depois de **compilada** gera o CSS.
- ❖ Esse pré-processador traz inúmeros benefícios como a possibilidade de criação de variáveis, utilização de classes aninhadas, *mixins*, operações matemáticas, etc.
- ❖ O SASS tem como objetivo tornar o processo de desenvolvimento mais simples e eficiente.



Sintaxe!

- ❖ SASS possui duas opções de sintaxe:
 - ❖ SCSS: Utiliza a extensão .scss, e é totalmente compatível com a sintaxe padrão CSS. É obrigatório o uso de chaves e ponto e vírgula após cada instrução.
 - ❖ Indentada (SASS): Utiliza a extensão .sass, sem as chaves padrão do CSS, todo código gerado em SASS deve ser transformado em CSS, pois navegadores não leem arquivos .sass.

SCSS

```
$color: red;

@mixin my-border($color) {
    border: 1px solid $color;
}

body {
    background: $color;
    @include my-border(green);
}
```

Indentada (SASS)

```
$color: red
=my-border($color)
    border: 1px solid $color

body
    background: $color
    +my-border(green)
```



Documentação e Instalação

- ❖ Documentação do Sass/Scss: <https://sass-lang.com/>
- ❖ Instalação global do Sass/Sassc: `npm install -g sass`
- ❖ Para gerar automaticamente o código css (já indicando a pasta), execute o comando a seguir no Pronpt de Comando (DOS) na pasta do projeto.

```
sass --watch scss/styles.scss:css/styles.css
```

Arquivo e pasta de origem

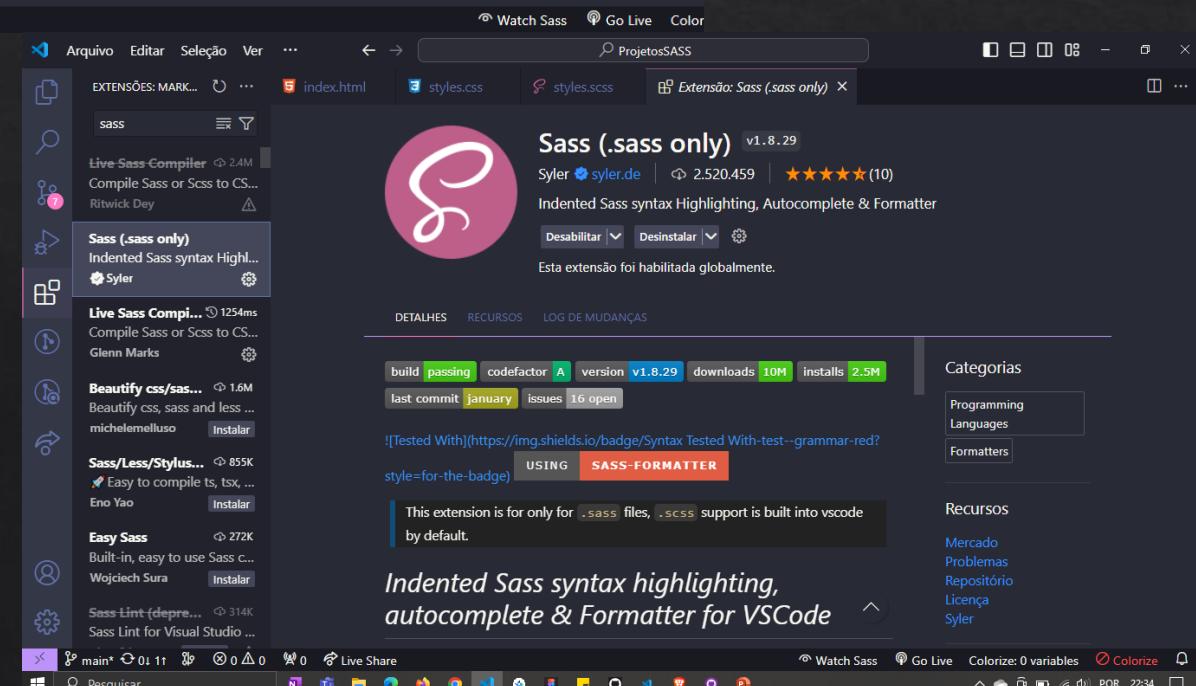
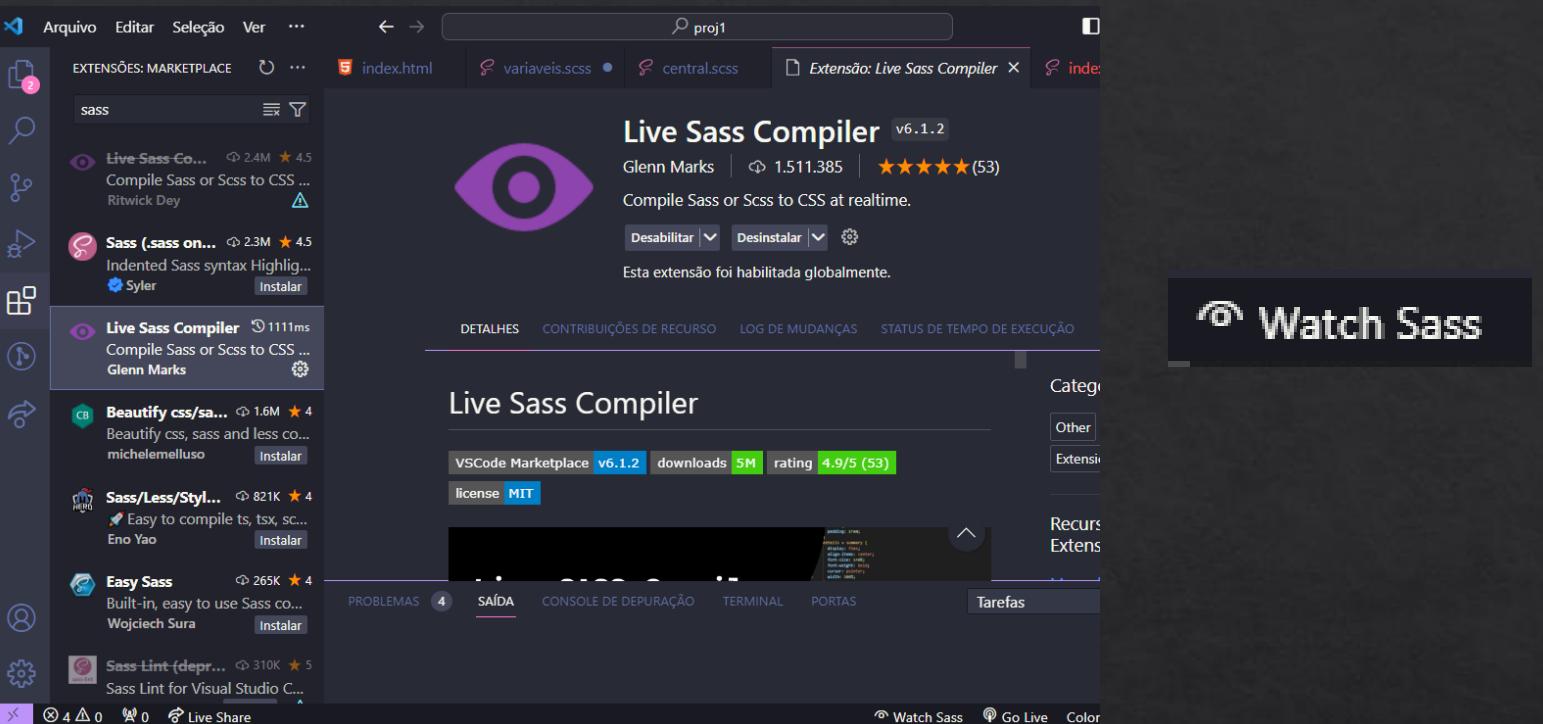


Arquivo e pasta de destino



Instalação no VSCode

- ◆ Em extensões, pesquise por “Live Sass Compiler”
- ◆ Ao instalar, clique em “Watch Sass” no rodapé do VSCode, após criar seu arquivo .scss
- ◆ Instale também a extensão “Sass (.sass only)” para reconhecer código .sass



Por que utilizar SASS?

- ❖ O SASS foi criado visando facilitar a criação de códigos CSS, por isso, nos traz várias vantagens que o CSS não possui. Algumas delas são:
- ❖ **Variáveis**
 - ❖ Podemos declarar uma variável utilizando o operador \$, então passamos a propriedade, podendo ela ser uma cor, fonte, tamanho da fonte, largura ou altura, etc.



Variáveis

- ◊ Podemos declarar uma variável utilizando o operador \$, então passamos a propriedade, podendo ela ser uma cor, fonte, tamanho da fonte, largura ou altura, etc.

```
$font-stack: Helvetica, sans-serif;  
$primary-color: #333;  
  
body {  
    font: 100% $font-stack;  
    color: $primary-color;  
}
```

```
$primaria: #E2CFEA;  
$secundaria: #A06CD5;  
$terciaria: #6247AA;
```

```
$textoClaro: #fff;  
$textoEscuro: #000;
```

```
$color-var: #157bad;  
$STRINGVAR: 'Texto';  
$numberVar: 30;  
$botaoAltura: 150px;
```



Escopo das variáveis em Scss

SCSS

```
// define the variable  
$purple: #6A67CE;  
  
.purple-btn  
{  
    text-align: center;  
    // re-defining the variable  
    $purple: red;  
    background-color: $purple;  
}  
  
.purple-text  
{  
    color: $purple;  
}
```

CSS
“resultado”

```
.purple-btn  
{  
    text-align: center;  
    background-color: red;  
}  
  
.purple-text  
{  
    color: #6A67CE;  
}
```



Por que utilizar SASS?

- ❖ **Nested Selectors (Seletores aninhados)**
 - ❖ *Em uma situação em que gostaríamos de acessar uma lista dentro de uma navbar. Seguindo a sintaxe padrão do CSS, iríamos utilizar o seguinte comando “navbar ul”. Agora, com o SASS, podemos apenas incluir a lista dentro da própria classe da navbar, facilitando o acesso, visualização e organização do código, que dessa maneira, flui de uma forma mais eficiente.*



Nested Selectors (Seletores aninhados)

SCSS

```
nav {  
  ul {  
    list-style: none;  
    margin: 0;  
    padding: 0;  
  }  
  
  li {  
    display: inline-block;  
  }  
  
  a {  
    display: block;  
    padding: 6px 12px;  
    text-decoration: none;  
  }  
}
```

CSS
“resultado”

```
nav ul {  
  list-style: none;  
  margin: 0;  
  padding: 0;  
}  
  
nav li {  
  display: inline-block;  
}  
  
nav a {  
  display: block;  
  padding: 6px 12px;  
  text-decoration: none;  
}
```



Utilizando Mixins (funções)

- ❖ Funções são uma excelente maneira de otimizar o código, impedir repetições e facilitar alterações.
- ❖ Exemplo:
 - ❖ Precisamos criar um texto de alerta, com a cor vermelha, em negrito e tamanho maior.
 - ❖ Ao invés de chamarmos essas propriedades cada vez que fosse necessário, podemos criar o mixin “texto-aviso”, incluindo-o dentro da nossa classe perigo, dessa forma, as propriedades incluídas nessa função serão passadas para essa classe.



Utilizando Mixins

- ❖ Algumas coisas em CSS são um pouco tediosas de escrever, especialmente com CSS3 e os muitos prefixos de fornecedores que existem. (ver comparação no slide a seguir)
- ❖ Um *mixin* permite criar grupos de declarações CSS que você deseja reutilizar em todo o seu site. Ajuda a manter seu Sass mais otimizado.
- ❖ Você pode até passar valores para tornar seu *mixin* mais flexível. Aqui está um exemplo para *theme*.

```
@mixin theme($theme: DarkGray) {  
    background: $theme;  
    box-shadow: 0 0 1px rgba($theme, .25);  
    color: #fff;  
}  
  
.info {  
    @include theme;  
}  
.alert {  
    @include theme($theme: DarkRed);  
}  
.success {  
    @include theme($theme: DarkGreen);  
}
```



Utilizando Mixins (funções)

SCSS

```
@mixin theme($theme: DarkGray) {  
  background: $theme;  
  box-shadow: 0 0 1px rgba($theme, .25);  
  color: #fff;  
}  
  
.info {  
  @include theme;  
}  
.alert {  
  @include theme($theme: DarkRed);  
}  
.success {  
  @include theme($theme: DarkGreen);  
}
```

CSS “resultado”

```
.info {  
  background: DarkGray;  
  box-shadow: 0 0 1px rgba(169, 169, 169, 0.25);  
  color: #fff;  
}  
  
.alert {  
  background: DarkRed;  
  box-shadow: 0 0 1px rgba(139, 0, 0, 0.25);  
  color: #fff;  
}  
  
.success {  
  background: DarkGreen;  
  box-shadow: 0 0 1px rgba(0, 100, 0, 0.25);  
  color: #fff;  
}
```

Interpolação SASS – usando #{}

- ❖ A interpolação é basicamente uma inserção. Ela nos permite interpolar expressões sass em um código SASS ou CSS simples. Significa que você pode definir (uma parte ou todo) o nome do seletor, o nome da propriedade, regras CSS, strings entre aspas, etc., como uma variável. A interpolação é um novo princípio e é amplamente utilizado no SASS.
- ❖ Para interpolar uma expressão, precisamos envolver a expressão usando #{}.



Interpolação SASS – usando #{}{}

```
/* syntax for interpolation */
#{}
/* anything inside will be evaluated */
```

SCSS

```
@mixin corner-icon($name) {
  /* using interpolation */
  .icon-#{$name} {
    background-image: url("/icons/#{$name}.svg");
  }
}

/* calling the above mixin */
@include corner-icon("mail");
```

CSS
“resultado”

```
.icon-mail {
  background-image: url("/icons/mail.svg");
}
```



Mais exemplos

SCSS

```
//Create the mixin for theme colors

@mixin theme($name, $color) {
    // Define colors in your theme
    $primary: $color;
    $secondary: lighten(adjust-hue($color, 20), 10%);
    // Add your classes with css colors added
    .#$name {
        .element {
            color: $primary;
        }
        .other-element {
            background: $secondary;
        }
    }
}
```

```
// Using the theme
@include theme(theme-banana, yellow);
```

CSS
“resultado”

```
// Output
.theme-banana .element {
    color: yellow;
}

.theme-banana .other-element {
    background: #bbff33;
}
```



Usando *Parent Selector* (Seletor de pai)

- ❖ O seletor de pai, &, é um seletor especial inventado pela Sass que é usado nos seletores encaixados para referir-se ao seletor externo.
- ❖ Ele torna possível reutilizar o seletor externo de maneiras mais complexas, como adicionar uma pseudo-classe ou adicionar um seletor antes do pai.

```
.alert {  
  // O seletor de pai pode ser usado para adicionar pseudo-classes ao  
  // seletor externo.  
  &:hover {  
    font-weight: bold;  
  }  
  
  // Ele também pode ser usado para estilizar o seletor externo num certo contexto,  
  // tal como um corpo definido para usar uma idioma de direita-para-esquerda.  
  [dir=rtl] & {  
    margin-left: 0;  
    margin-right: 10px;  
  }  
  
  // Tu podes ainda usá-lo como um argumento para os seletores de pseudo-classe.  
  :not(&) {  
    opacity: 0.8;  
  }  
}
```

Usando Parent Selector

SCSS

```
#studytonight {  
    /* styling for #studytonight div */  
    &-body {  
        /* styling for #studytonight-body div */  
    }  
}
```

HTML
“uso”

```
<div id="studytonight">  
    <div id="studytonight-body">  
        /* Some HTML code */  
    </div>  
</div>
```

SCSS

```
a {  
    text-decoration: none;  
    color:black;  
    // using parent selector  
    &:hover {  
        color:red;  
    }  
}
```

CSS
“resultado”

```
a{  
    text-decoration: none;  
    color:black;  
}  
a:hover  
{  
    color:red;  
}
```



Conclusão:

- ❖ O SASS nos fornece novas funcionalidades que agregam, facilitam e simplificam o processo de desenvolvimento.
- ❖ Para aqueles que estudam web design, o SASS acaba sendo uma ótima ferramenta, pois além dos recursos, possui uma sintaxe fácil de compreender.

