Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (front 1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

CSS

Syntactically Awesome StyleSheet

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas

O que é SASS?

Acrônimo para ***Syntactically Awesome Stylesheet***.

Traduzido para o português como **Folhas de Estilo Sintaticamente Incríveis**.

- É uma **extensão** do CSS

  Objetivo é adicionar recursos especiais como variáveis, mixins, funções, operações, etc.

  A ideia é manter a mesma lógica do CSS (seletores, regras...), mas de uma maneira mais organizada, intuitiva e com trechos de **código facilmente reutilizáveis.**

- SASS é um **pré-processador** CSS

  Ou seja, é um programa que tem uma entrada, um processamento e uma saída.

- É **totalmente compatível** com todas as versões de CSS.
  - Reduz a repetição de CSS e, portanto, **economiza tempo.**
- Foi projetado por **Hampton Catlin** e desenvolvido por **Natalie Weizenbaum** em 2006.
  - SASS é **gratuito** para baixar e usar.

Porque SASS?

As CSS estão ficando cada vez **maiores**, **complexas** e **difíceis** de manter.

Neste cenário, um **pré-processador CSS** pode ajudar.

SASS permite que você use recursos que não existem no CSS, como **variáveis, regras aninhadas, mixins, importações, herança, funções integradas e outras coisas**.

Exemplo

#a2b9bc

#b2ad7f

#878f99

```
/* definindo variáveis as principais cores de web site*/
$cor_1: #a2b9bc;
$cor_2: #b2ad7f;
$cor_3: #878f99;
```

arquivo.scss

```
/* usando as variáveis*/
.principal {
  background-color: $cor_1;
}

.menu-esquerdo {
  background-color: $cor_2;
}
```

Como SASS funciona?

Um navegador não entende o código **SASS**.

Se faz necessário um pré-processador para **converter o código SASS em CSS padrão**.

Este processo é denominado transpilar.

Transpilar (ou *transpile*) refere-se ao processo de pegar código escrito em uma linguagem de programação de **alto nível** e convertê-lo para outra linguagem ou para uma versão diferente da mesma linguagem, **mantendo-se no nível de código fonte**.

Outros exemplos de transpilação no desenvolvimento web, além de SASS >> CSS:

- **TypeScript para JavaScript**: TypeScript é uma linguagem que se baseia em JavaScript. O código TypeScript é transpilado para JavaScript **para ser executado em navegadores ou ambientes que suportam JavaScript**.
- **ES6/ES2015+ para ES5**: As versões mais recentes do JavaScript (ES6/ES2015 e posteriores) introduziram novos recursos que nem todos os navegadores suportam nativamente. Ferramentas de transpilação, como Babel, convertem esse código para uma versão mais antiga do JavaScript (ES5) **para garantir compatibilidade**.

Transpilar x Compilar

Aspecto	Transpilar	Compilar
Definição	Tradução de uma linguagem de programação de alto nível para outra linguagem de alto nível ou para uma versão diferente da mesma linguagem.	Transformação de código de uma linguagem de alto nível para código de máquina (executável) ou bytecode.
Nível de Abstração	Mantém o código em um nível de alto nível, geralmente legível e compreensível para desenvolvedores.	Converte para um nível de baixo nível, como código de máquina ou bytecode, geralmente não legível para humanos.
Objetivo	Facilitar o desenvolvimento usando sintaxes ou funcionalidades avançadas, garantindo compatibilidade ou padronização.	Produzir código executável pelo hardware ou por uma máquina virtual, com foco em desempenho e eficiência.
Exemplos	TypeScript para JavaScript, ES6 para ES5, SASS para CSS.	C++ para executável nativo, Java para bytecode (JVM).

Sobre o SASS

Tipo de arquivo: Arquivos SASS têm a extensão de arquivo "`.scss`".

Comentários:

SASS oferece suporte para comentários CSS padrão:

```
/* comentário */
```

Também suporta comentários *inline*

```
// comentário de linha
```

```
/* comentário padrão */
$cor_1: #a2b9bc;
$cor_2: #b2ad7f;

.principal {
  background-color: $cor_1; // comentário inline
}
```

.sass x .scss

.sass

Suportada, mas **não é mais a padrão**.

arquivo.sass

```
$cor:blue
h1
  color:$cor
```

Usa **sintaxe baseada em indentação** (semelhante ao Python)

- Não tem ';' marcando o final de instrução.
- Nova linha indica que é uma nova instrução.

.scss (sassy css)

Aceita código CSS no mesmo arquivo SASS.

arquivo.scss

```
$cor:blue;
h1{
    color: $cor;
}
```

Transpilação

arquivo.css

```
h1{
    color: blue;
}
```

.sass x .scss

Característica	.sass	.scss
Sintaxe	Sintaxe indentada (sem chaves ou ponto e vírgula).	Sintaxe mais semelhante ao CSS, com chaves {} e ponto e vírgula ;.
Uso de Pontuação	Não utiliza chaves {} nem ponto e vírgula ;.	Utiliza chaves {} para definir blocos e ponto e vírgula ; para separar declarações.
Facilidade de Leitura	Requer um estilo de código mais limpo, dependendo da indentação para determinar blocos de código.	Mais próxima do CSS tradicional, facilitando a transição de desenvolvedores que já estão familiarizados com CSS.
Compatibilidade	Suportada pela maioria das ferramentas e compiladores SASS.	Também amplamente suportada e, muitas vezes, preferida devido à sua semelhança com o CSS.
Adoção	Menos utilizada em novos projetos, mas ainda suportada.	Mais popular em projetos modernos e considerada a sintaxe padrão para novos desenvolvimentos.
Comentários	Utiliza // para comentários de uma linha e /* ... */ para múltiplas linhas.	Igual ao CSS e SASS, utilizando // para comentários de uma linha e /* ... */ para múltiplas linhas.

Instalação do SASS

SASS

Sistema Operacional: **independente de plataforma**

Suporte de navegador: **funciona no Edge/IE (a partir do IE 8), Firefox, Chrome, Safari, Opera**

Linguagem de programação: **Sass é baseada em Ruby**

https://sass-lang.com/

Instalando o SASS (Windows)

>>cmd (prompt de comandos)

>>npm --version

Caso não tenha o npm (*Node Package Manager*), instale-o junto com ambiente do Node.JS
https://docs.npmjs.com/downloading-and-installing-node-js-and-npm

>>npm install -g sass

Comando para instalar o SASS

>>sass --version

sass-lang.com/install

Outras ferramentas

Koala

É um aplicativo GUI para compilação Less, Sass, Compass e CoffeeScript, para ajudar os desenvolvedores da Web a usá-los com mais eficiência. Koala pode ser executado em Windows, Linux e Mac.

http://koala-app.com/

Live Sass Compiler

Uma extensão VSCode que ajuda o processo de compilar/transpilar arquivos SASS/SCSS para arquivos CSS em tempo real com recarregamento do navegador ao vivo.

https://marketplace.visualstudio.com/items?itemName=ritwickdey.live-sass

Variáveis

SASS

Variáveis SASS

São uma forma de armazenar informações que podem ser reutilizadas posteriormente.

SASS permite armazenar informações em variáveis, como:

1. Strings
2. Numbers
3. Colors
4. Booleanos
5. Listas
6. nulls

Sintaxe

```
$nomeDaVariavel: valor;
```

estilo.scss

```
$minhaFonte: Helvetica, sans-serif;
$cor1:rgb(34, 104, 34);
$tamanhoFontePrincipal: 18px;
$largura: 680px;

body {
    font-family: $minhaFonte;
    font-size: $tamanhoFontePrincipal;
    color: $cor1;
}

#container {
    width: $largura;
}
```

estilo.css

```
body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: #226822;
}

#container {
  width: 680px;
}
```

Exemplo_01

Escopo das variáveis SASS

Estão disponíveis apenas no **nível de aninhamento** em que foram definidas.

estilo.scss

```
$minhaCor: red;

h1 {
  $minhaCor: green;
  color: $minhaCor;
}


p {
  color: $minhaCor;
}
```

estilo.css

```
h1 {
  color: green;
}


p {
  color: red;
}
```

Esse é o comportamento padrão para o escopo da variável.

SASS !global

O **comportamento padrão** do escopo da variável pode ser substituído usando a opção `!global`.

estilo.scss

```
$minhaCor: red;

h1 {
  $minhaCor: green !global;
  color: $minhaCor;
}


p {
  color: $minhaCor;
}
```

`!global` indica que uma variável é global, ou seja, está acessível em todos os níveis.

estilo.css

```
h1 {
  color: green;
}


p {
  color: green;
}
```

Dica: Variáveis globais devem ser definidas fora de quaisquer regras.

É sensato definir as variáveis globais em seu próprio arquivo, "**_globals.scss**", e incluí-lo com a palavra-chave **@include**.

Regras aninhadas

SASS

Regras aninhadas SASS

SASS permite **aninhar seletores** CSS da **mesma forma que o HTML**.

arquivo.sass

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li {
    display: inline-block;
  }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

No Sass, os seletores `ul`, `li` e `a` estão aninhados dentro do seletor `nav`.

Essa funcionalidade permite escrever menos código de forma mais legível, tornando sua manutenção mais fácil.

arquivo.css

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}

nav li {
  display: inline-block;
}

nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

Enquanto em CSS, as regras são definidas uma a uma (não aninhadas).

Exemplo_02

Propriedades aninhadas SASS

Muitas propriedades CSS têm o **mesmo prefixo**:

font-family, **font**-size e **font**-weight

text-align, **text**-transform e **text**-overflow.

Estas propriedades podem ser escritas de forma aninhada no SASS.

Exemplo_03

```
p {
    font: {
       family: Lucida Sans Unicode;
       size:20px;
       weight: bold;
       variant: small-caps;
    }

    text: {
        align: center;
        transform: lowercase;
        overflow: hidden;
      }
 }
```

estilo.scss

O transpilador SASS converte o código para CSS normal

```
p {
  font-family: Lucida Sans Unicode;
  font-size: 20px;
  font-weight: bold;
  font-variant: small-caps;
  text-align: center;
  text-transform: lowercase;
  text-overflow: hidden;
}
```

estilo.css

Diretiva SASS

@import

SASS @import

Diretiva que permite **incluir** o conteúdo de um arquivo em outro.

```
@import nomeDoArquivo;
```

Exemplo_04

reset.scss

```scss
html, body, ul, ol {
    margin: 0;
    padding: 0;
}
```

estilo.scss

```scss
@import "reset";

body {
    font-family: Helvetica, sans-serif;
    font-size: 18px;
    color: rgb(24, 92, 24);
}
```

Não é preciso especificar uma extensão de arquivo, Sass automaticamente assume que é um arquivo .sass ou .scss.

estilo.css

```css
html, body, ul, ol {
  margin: 0;
  padding: 0;
}

body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: #185c18;
}
```

SASS

partials

SASS partials

Por padrão, o **Sass transpila todos os arquivos** .scss diretamente.

No entanto, quando se deseja importar um arquivo, **não é necessário que o ele seja transpilado diretamente**.

partials é o mecanismo que faz com o arquivo **não seja transpilado imediatamente**.

_nomeDoArquivo.scss;

Exemplo_05

partial

_cores.scss

```
$rosa: #ee82ee;
$azul: #4169e1;
$verde: #8fbc8f;
```

Partials são arquivos scss iniciados com "_"

estilo.scss

```
@import "cores";
body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: $azul;
}
```

estilo.css

```
body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: #4169e1;
}
```

Ao importar o arquivo parcial, omita o sublinhado. **O Sass entende que deve importar o arquivo "_colors.scss"**

Diretivas SASS

@mixin e @include

@mixin

Diretiva que permite a criação de código CSS que pode ser **reutilizado** em todo o site.

Sintaxe

```
@mixin name {
  property: value;
  property: value;
  ...
}
```

```
@mixin texto-importante {
  color: red;
  font-size: 25px;
  font-weight: bold;
  border: 1px solid blue;
}
```

```
.atencao {
  @include texto-importante;
  background-color: green;
}
```

@include

Diretiva que possibilita a **inclusão do @mixin**.

Sintaxe

```
selector {
  @include mixin-name;
}
```

```
.atencao {
  color: red;
  font-size: 25px;
  font-weight: bold;
  border: 1px solid blue;
  background-color: green;
}
```

Dica: Hifens e sublinhados são considerados iguais.

@mixin importante-texto { } e **@mixin importante_texto { }**

São considerados como o mesmo **mixin**!

Exemplo_06

Passando variáveis para um @mixin

mixins aceitam argumentos, assim **é possível passar variáveis para eles**.

Parâmetros de entrada

```
@mixin borda($cor, $largura) {
  border: $largura solid $cor;
}
```

Argumentos

```
.artigo {
  @include borda(purple, 10px); // chamada a mixin
}

.notas {
  @include borda(green, 4px); // chamada a mixin
}
```

```
.artigo {
  border: 10px solid purple;
}

.notas {
  border: 4px solid green;
}
```

Exemplo_07

Valores padrão em um @mixin

É possível definir **valores *default*** para os **mixins**.

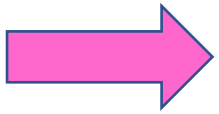
```
@mixin borda($cor: blue, $largura: 1px) {
  border: $largura solid $cor;
}

.artigo {
  @include borda($largura: 50px);
}

.notas {
  @include borda(green, 4px);}

.rodape {
  @include borda;
}
```

Então, só é preciso especificar os valores que mudam quando o **mixin** é incluído.



```
.artigo {
  border: 50px solid blue;
}

.notas {
  border: 4px solid green;
}

.rodape {
  border: 1px solid blue;
}
```

Exemplo_07_b

Diretiva SASS

@extend e herança

Sass @extend (herança)

Permite **compartilhar um conjunto de propriedades** CSS de um seletor para outro.

Útil se temos elementos de estilo quase idênticos que **diferem apenas em pequenos detalhes**.

```
.botao-basico {
  border: none;
  padding: 15px 30px;
  text-align: center;
  font-size: 16px;
  cursor: pointer;
}

.botao-report {
  @extend .botao-basico;
  background-color: red;
}

.botao-submit {
  @extend .botao-basico;
  background-color: green;
  color: white;
}
```

```
.botao-basic, .botao-report, .botao-submit {
  border: none;
  padding: 15px 30px;
  text-align: center;
  font-size: 16px;
  cursor: pointer;
}

.botao-report {
  background-color: red;
}

.botao-submit {
  background-color: green;
  color: white;
}
```

Exemplo_08

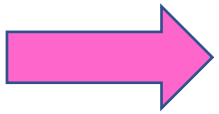
Herança com múltiplas diretivas

Uma classe herda características duas ou mais classes.

```
.classe1 {
    color: red;
}

.classe2 {
    background-color: black;
}


.vermelho {
    @extend .classe1;
    @extend .classe2;
}
```



```
.classe1, .vermelho {
  color: red;
}


.classe2, .vermelho {
  background-color: black;
}
```

Exemplo_17

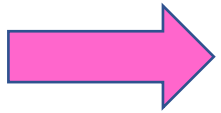
Herança com encadeamento

Uma classe **herda características de uma classe, que já herdou características de outra antes**.

```
.pai {
    color: red;
}

.filho {
    @extend .pai;
    background-color: blue;
}


.neto {
    @extend.filho;
    padding: 15px;
}
```



```
.pai, .filho, .neto {
  color: red;
}

.filho, .neto {
  background-color: blue;
}

.neto {
  padding: 15px;
}
```

Exemplo_18

Placeholders (%)

Recurso que permite o **aproveitamento da relação de herança**, sem que isso de fato gere um CSS.

Suponha que não desejemos criar as classes **pai** e **filho** no código CSS. **Só queremos aproveitar a relação de herança, sem que isso de fato gere uma codificação CSS.**

```
%pai {
    color: red;
}

%filho {
    @extend %pai;
    background-color: blue;
}

.neto {
    @extend %filho;
    padding: 15px;
}
```

```
.neto {
  color: red;
}

.neto {
  background-color: blue;
}

.neto {
  padding: 15px;
}
```

Exemplo 19

Diretivas para controle de fluxo

Diretivas de controle para laços de repetições

for... through

```
@for $i from 1 through 3 {
    .item-#{$i} {
        background: green;
        margin-bottom: 2px;
    }
}
```

```
.item-1 {
  background: green;
  margin-bottom: 2px;
}

.item-2 {
  background: green;
  margin-bottom: 2px;
}

.item-3 {
  background: green;
  margin-bottom: 2px;
}
```

Exemplo_10

se **for... through** é usado,
o último valor **é incluído**.

```
<div class="item-1">Item 1</div>
<div class="item-2">Item 2</div>
<div class="item-3">Item 3</div>
```

Diretivas de controle para laços de repetições

for... to

```
@for $i from 1 to 3 {
    .item-#{$i} {
        background: green;
        margin-bottom: 2px;
    }
}
```

```
.item-1 {
  background: green;
  margin-bottom: 2px;
}

.item-2 {
  background: green;
  margin-bottom: 2px;
}
```

se **for... to** é usado, o último valor **não é incluído**.

```
<div class="item-1">Item 1</div>
<div class="item-2">Item 2</div>
<div class="item-3">Item 3</div>
```

Diretivas de controle para laços de repetições

while

```
$contador: 1;

@while $contador<5 {
    .item-#{$contador} {
        background: orange;
        margin-bottom: 2px;
    }
    $contador: $contador + 1;
}
```

```
.item-1 {
  background: orange;
  margin-bottom: 2px;
}

.item-2 {
  background: orange;
  margin-bottom: 2px;
}

.item-3 {
  background: orange;
  margin-bottom: 2px;
}

.item-4 {
  background: orange;
  margin-bottom: 2px;
}
```

Exemplo_12

```
<div class="item-1">Item 1</div>
<div class="item-2">Item 2</div>
<div class="item-3">Item 3</div>
<div class="item-4">Item 4</div>
…
```

Diretivas de controle para laços de repetições

each in

```
$lista: blue, yellow, red;
$cont: 1;

@each $cor in $lista {
    .item-#{$cont} {
        background: $cor;
        margin-bottom: 2px;
    }

    $cont: $cont + 1;
}
```

```
.item-1 {
  background: blue;
  margin-bottom: 2px;
}

.item-2 {
  background: yellow;
  margin-bottom: 2px;
}

.item-3 {
  background: red;
  margin-bottom: 2px;
}
```

Exemplo_11

```
<div class="item-1">Item 1</div>
<div class="item-2">Item 2</div>
<div class="item-3">Item 3</div>
<div class="item-4">Item 4</div>
```

Diretivas de controle condicionais

```
$cor: yellow;

@if ($cor == green) {
    p {
        color: $cor;
    }
} @else if($cor== red) {
    p {
        color: $cor;
    }
} @else {
    p {
        color: black;
    }
}
```

```
p {
  color: black;
}
```

Exemplo 20

Diretivas de controle com @mixin

Imagine uma situação em que o seu site, tem **temas que variam** de acordo com alguma época do ano.

```
@mixin tema($tema: padrao){
    @if($tema == natal){
        background: red;
        color: white;
    } @else if($tema == blackfriday){
        background: black;
        color:white;
    } @else{
        background: white;
        color: black;
    }
}
```

```
body {
  background: black;
  color: white;
}
```

```
body{
    @include tema(blackfriday);
}
```

Exemplo_21

Funções

Funções

Tal como em linguagem de programação, **funções de Sass** (diretiva `@function`) permitem fazer operações complexas e **retornar valores**.

A diferença para `mixins` é que a função retorna somente um valor, tendo, inclusive, uma diretiva especial para informar isso, a `@return`.

```
@function potencia($base, $expoente) {
    $resultado: 1;
    @for $_ from 1 through $expoente {
      $resultado: $resultado * $base;
    }
    @return $resultado;
  }

  .sidebar {
    float: left;
    margin-left: potencia(4, 3) * 1px;
  }
```

```
.sidebar {
  float: left;
  margin-left: 64px;
}
```

Exemplo

Funções

Exemplo 13

É possível utilizar uma função para calcular a porcentagem e desenvolver dinamicamente um sistema de **grid**, semelhante ao trabalhado em aulas anteriores.

1) Estabelecemos o número total de colunas do grid (12).

2) Definimos uma função para calcular a largura q devem ser ocupadas em cada um dos casos

```
$total:12;


@function largura-coluna($coluna) {
    @return $coluna/$total * 100;
}


@for $i from 1 through 12{
    $resultado: largura-coluna($i);
    /* #{$resultado} */
}
```

3) Ao passarmos um número indicando o número de colunas, a função retorna o espaço que ela ocupara.

```
/* Funções */
/* 8.33333 */
/* 16.66667 */
/* 25 */
/* 33.33333 */
/* 41.66667 */
/* 50 */
/* 58.33333 */
/* 66.66667 */
/* 75 */
/* 83.33333 */
/* 91.66667 */
/* 100 */
/*
```

4) É possível usar funções nativas, como a **percentage**, que já traz o resultado percentualmente.

```
$total:12;

@function largura-coluna($coluna) {
    @return percentage($coluna/$total);
}
```

```
/* 8.3333333333% */
/* 16.6666666667% */
/* 25% */
/* 33.3333333333% */
/* 41.6666666667% */
/* 50% */
/* 58.3333333333% */
/* 66.6666666667% */
/* 75% */
/* 83.3333333333% */
/* 91.6666666667% */
/* 100% */
```

Criando um grid com funções


```scss
$total:12;

@function largura-coluna($coluna) {
    @return percentage($coluna/$total);
}



@for $i from 1 through $total {
    .col-#{$i} {
        width: largura-coluna($i);
        background: black;
    }
}
```

```css
/* Criando um grid */
.col-1 {
  width: 8.33333%;
  background: black;
}

.col-2 {
  width: 16.66667%;
  background: black;
}

.col-3 {
  width: 25%;
  background: black;
}

.col-4 {
  width: 33.33333%;
  background: black;
}

.col-5 {
  width: 41.66667%;
  background: black;
}

.col-6 {
  width: 50%;
  background: black;
}

.col-7 {
  width: 58.33333%;
  background: black;
}

.col-8 {
  width: 66.66667%;
  background: black;
}

.col-9 {
  width: 75%;
  background: black;
}

.col-10 {
  width: 83.33333%;
  background: black;
}

.col-11 {
  width: 91.66667%;
  background: black;
}

.col-12 {
  width: 100%;
  background: black;
}
```

Exemplo_14

Outras funções nativas


```
.misturar{
    background: mix(green, orange, 70%);
    color: yellow;
}
```

Mistura as cores, baseado na porcentagem.

```
.escurecer{
    background: darken(blue, 50%);
    color: yellow;
}
```

Escurece a cor seguindo determinada porcentagem.

```
.clarear{
    background: lighten(black, 70%);
    color: yellow;
}
```

Clareia a cor, baseado na porcentagem passada.

Exemplo_15

Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (front 1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

CSS

Syntactically Awesome StyleSheet

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas