Sass

@Import

*Para importamos arquivos em css, usamos o atributo `@import "nomeDoarquivo.css"`, também podemos usar o @import em Sass para importamos todos arquivos scss dentro do **style.scss**, não havendo necessidade de importamos no arquivo css, aumentando o processamento de nosso código pelo browser.*

Ao importamos todos os arquivos scss para dentro do **Style.scss**, o arquivo final gerado pelo pré-processador utilizado será um css com todos esses arquivos incluídos automaticamente.

Informações necessárias

Não precisamos definir a extensão .scss ao importarmos usando Sass

```
1 @import "nomedoarquivo"
```

Nota: Arquivos que devem ser processados numa ordem predefinida devem ser importados na ordem desejada, pois serão processados da forma como foram importados.

_Partials

*Ao usarmos um pré-processador como (Scout, ou Gulp), estes identificarão todos os arquivos .scss daquele diretório, e farão a concatenação para gerar um único arquivos css. Podemos impedir que tal ação seja impedida usando um **Partials** "_" na frente do nome do arquivo*

nomeDoArquivo.scss

Variáveis

Assim como em linguagens de programação, as variáveis são responsáveis por armazenar um valor que poderá ser utilizado mais tarde. Em Sass não é diferente, podemos armazenar valores de fontes, cores, tamanhos e etc.

Informações necessárias

Para definirmos uma variável em Sass, usamos o símbolo de cifrão atrelado ao nome da variável.

```
$nomeVariavel: blue;
```

Exemplo de Código

```
$cor-primaria: #252525;
$fonte-primaria: Arial;

h1 {
    color: $cor-primaria;
    font-family: $fonte-primaria;
}
```

Aninhamento / Nesting

Um aninhamento em Sass é quando inserimos um seletor dentro de outro, organizando essa inserção em camadas. Ao processarmos Sass para Css, o arquivo final terá todos os seletores organizados da forma padrão de Css.

Informações necessárias

O uso de aninhamento Sass em código, pode trazer algumas dores de cabeça quando tentamos identificar a ordem em que se encaixam, por este motivo o recomendado é que o uso de aninhamentos, sejam de até 3 seletores.

Tipos de Aninhamento

- **& (Ampersand)** - Usado para conectar um determinado seletor a outro: `a:hover{}`, estando dentro do seletor **a** , seria `&:hover{}`.
    - Também podemos utilizar o **&** na frente do seletor, fazendo com que o seletor selecionado seja enviado para o primeiro lugar do código: `.active &`, como exemplo ficaria `.active ul li{}`.
- **Nesting de Propriedades*** - Alinha todas as propriedades de um determinado seletor:

```
p {
    font: {
        size: 16px;
        family: Arial;
        weight: bold;
        style: italic;
    }
}
```

Exemplo de Código

```
.aninhamento ul {
    text-align: center;

    li {
        display: inline-block;
        list-style: none;

        a{
            display: block;
            padding: 20px;
            background: $cor-secundaria;
            text-decoration: none;
            color: $cor-terciaria;
            &:hover{
                background: $cor-bg;
            }
        }
    }
}
```

@Mixins

Um **@mixin** funciona como uma função, onde podemos definir dentro do mesmo vários elementos de um seletor, e chamar essa função através do comando **@include** em outro seletor usando os recursos definidos anteriormente.

Informações necessárias

- **Mixin dentro de Mixin** - Podemos incluir um @mixin dentro de outro.
- **Parâmetros** - Podemos definir um ou mais argumentos à um @mixin, e ao chamarmos na função via @include, inserimos o valor de argumento necessitado.

```
@mixin nomeMixin ($parametro1, $parametro2) {
    //código
    }
}
```

  - **Parâmetro com valor padrão** - Também podemos definir um valor padrão ao @mixin, podendo este valor ser sobrescrito quando na chamada da função

@include.

- **Argumentos** - Elementos como **box-shadow e text-shadow**, podem receber outros valores após a vírgula, e através da inserção de 3 pontos no parâmetro, estamos insinuando ao @mixin que tal parâmetro pode ter mais de um valor: `@mixin box-shadow($shadow...)`.
- **@content** - Permite que sejam adicionadas novas linhas de código ao bloco, na chamada do @include:

```scss
@mixin mobile {
    @media (max-width: 600px) {
        @content;
    }
}

//Chamada da função

.aninhamento h2 {
    @include titulo-grande;

    @include mobile {
        text-align: left;
        font-size: 1.5em;
    }
}
```

Exemplo de Código

```scss
@mixin separador($color, $largura, $altura: 4px) {
    &::after {
        content: '';
        display: block;
        width: $largura;
        height: $altura;
        background: $color;
        margin: 0 auto;
    }
}

.aninhamento h2 {
    @include titulo-grande;
    @include separador($cor-secundaria, 100px);
}
```

Operadores Matemáticos

Como em quaisquer linguagens de programação, podemos usar operadores matemáticos como (soma, subtração, multiplicação e divisão).

Informações necessárias

- **Somando cores** - Podemos usar operadores matemáticos para somar elementos como cores
    - `background: #333 + #777;`

Exemplo de Código

```
$gutter: 20px;

div {
    width: 400px - $gutter;
    padding: $gutter;
}
```

Condicionais

Como em programação, podemos usar as cláusulas If/Else para realizar determinadas ações no css.

Informações necessárias

- *Sinais de comparação* < = > - Podemos usar os sinais (maior >), (menor <), ou (igual =), para compararmos expressões em css.

Exemplo de Código

```
@mixin tipografia-1($tamanho) {
    font-family: monospace;

    @if $tamanho==16 {
        font-size: 1em;
        font-weight: normal;
    }

    @else if $tamanho==18 {
        font-size: 1.125em;
        font-weight: normal;
    }

    @else if $size=24 {
        font-size: 1.5em;
        font-weight: boldF;
    }
}

//Chamando o mixin

p {
    @include tipogrfia-1(16);
}
```

Funções

Documentação Sass Built-in Modules

O Sass disponibiliza algumas funções pré-estabelecidas, que podem ser usadas para; escurecer determinado tom de cor com a função **Darken**, ou clarear um tom de cor com a função **Lighten**, abaixo mais exemplos:

Informações necessárias

- **Darken (cor, porcentagem)** - Escurece determinado tom de cor, com base no valor de porcentagem passado.
- **Lighten (cor, porcentagem)** - Clareia determinado tom de cor, com base no valor em porcentagem passado.
- **Transparentize (cor, valor decimal)** - Transforma uma cor em rgba(), com o tom de transparência informado.
- **Round (expressão matemática)** - Arredonda o valor da expressão matemática passado em parâmetro.
- **@function** - Também podemos criar nossas próprias funções, como em uma linguagem de programação, normalmente.

Exemplos de Código

Darken

```
a:hover {
    background-color: darken($rosa, 20%);
}
```

Lighten

```
a:hover {
    background-color: lighten($rosa, 20%);
}
```

Transparentize

```
$rosa: pink;

a:hover {
    background-color: transparentize($rosa, 0.5);
}
```

Round

```
$width: 100%;

div {
    width:  round($width / 3);
}
```

@function

Calculando **em**

```
@function em($pixels, $contexto: 16) {
@return ($pixels / $contexto) * 1em;
}
```

A função acima recebe um valor de pixel e retorna a o valor da divisão de pixel por contexto em **em**.

Calculando **Grid**

```
@function grid($colunas, $total: 12) {
    @return ($colunas / $total) * 100%;
}
```

Loop

O Sass possui os tipos **@for**, **@while** e **@each**

Exemplos de Código

@for

```
//Grid em Porcentagem

$container: 100;
$gutter: 20;
$colunas: 12;

@for $i from 1 through $colunas {

    $width: $container / $colunas *$i + %;

    .grid-#{$i} {
        width: $width;
    }
}

//Css

```

```
.grid-1 {
  width: 8.33333%;
}

.grid-2 {
  width: 16.66667%;
}

//etc..

```

@while

```
$i: 1;

@while $i <= 6 {
    .type-#{$i} {
    font-size: 16 * $i + px;
    }

    $i: $i + 1;
}

//Css

.type-1 {
  font-size: 16px;
}

.type-2 {
  font-size: 32px;
}

//etc..

```

@each

```
$lista: facebook twitter instagram youtube;

@each $item in $lista {
    .rede-#{$item} {
        background-image: url('img/#{$item}.png');
    }
}

//Css

.rede-facebook {
  background-image: url("img/facebook.png");
}

```

```
.rede-twitter {
  background-image: url("img/twitter.png");
}

//Etc..

```