45697056

ES

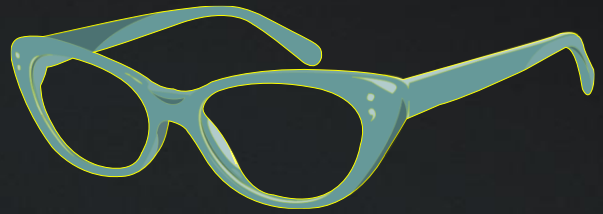
Front End Development

Prof. Alexandre Carlos profalexandre.jesus@fiap.com.br

Prof. Luís Carlos lsilva@fiap.com.br

45697056

SASS



CSS with superpowers

SASS

45697056

SASS = Syntactically Awesome StyleSheets

Sass é um préprocessador.

Mas o que é um **pré-processador CSS**?

Um **pré-processador CSS** é um programa que permite você gerar **CSS** a partir de uma sintaxe (en-US) única desse **pré-processador**.

Essas funcionalidades fazem a estrutura do **CSS** mais legível e fácil de manter, além de que permite que você use variáveis, regras aninhadas, mixins, importações e mais!

https://sass-lang.com/

SASS

45697056

Vamos instalar o SASS:

npm install –D sass

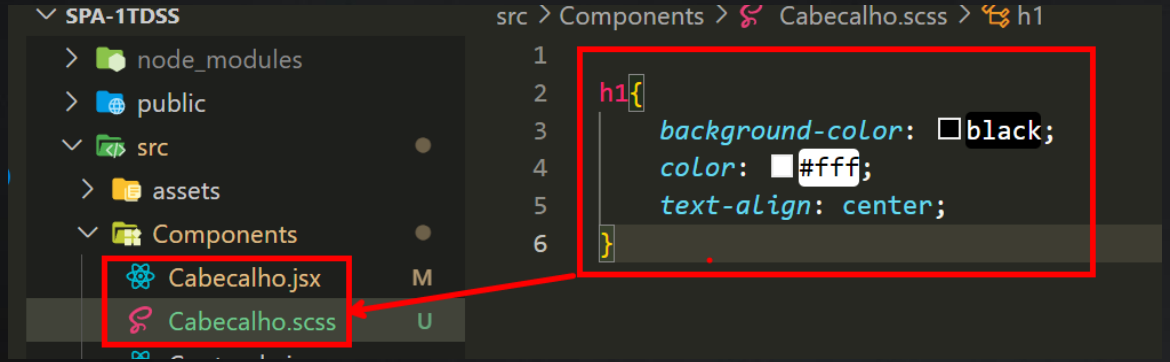
https://sass-lang.com/

SASS

45697056

Utilização:

Crie o arquivo scss e adicione o código conforme a contextualização:



https://sass-lang.com/

SASS

45697056

Utilização:

Depois basta importar o arquivo dentro do componente:

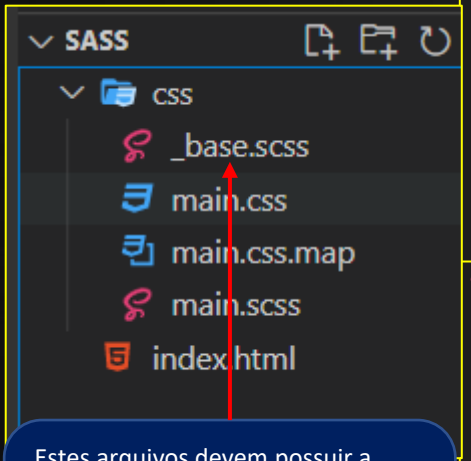
```
import { Link } from "react-router-dom";
import "./Cabecalho.scss";

export default function Cabecalho(){
      You, 3 weeks ago • Iniciamos com Componentização e Props
    return(
        <>
            <header>
```

https://sass-lang.com/

@IMPORT/PARTIALS

45697056

- Nós já vimos os a parte de organizar os arquivos em pastas, agora vamos ver como importar arquivos dentro do sass e também poder criar partials, uma espécie de arquivo que só é compilado se for importado.
- **Ex:**



Estes arquivos devem possuir a extensão **scss** e iniciar o nome com uma (_), sublinha ou underscore. Assim eles serão ignorados pelo compilador e poderão ser importados tranquilamente.

main.scss

```scss
$back-color: #e20775;

body {
    background-color: $back-color;
    @import "base";
}
```

Utilize aspas parao importe deste arquivo e não colo a extensão e nem a sublinha(underscore), apenas o nome. Nunca se esqueça do pont e virgula (;)

_base.scss

```scss
.container {
    $text-color: #00ff00;
    color: $text-color;
    p {
        color: #0000ff;
    }
}
```

Quando criamos arquivos partials a idéia principal é organizer todo o trabalho em componentes. Ou seja, criar CSS para partes do projeto.

https://sass-lang.com/

@IMPORT/PARTIALS

- A utilização dos import e partials é muito importante nos dias de hoje, pois com isso conseguimos tornar nosso código muito mais organizado.
- **Ex:**

main.css

css > main.css > body

```
1  body {
2    background-color: #e20775;
3  }
4
5  body .container {
6    color: #00ff00;
7  }
8
9  body .container p {
10   color: #0000ff;
11 }
12 /*# sourceMappingURL=main.css.map */
```

https://sass-lang.com/

@MIXINS / @INCLUDE

- Mixins são agrupamento de declarações que podem ser reusadas.
- Imagine que você criou um código CSS que se repete e se repete, porém tem algo que mude nesse código. Você pode utilizar o mixins com include para solucionar este problema.
- Criei um mixin de um box-shadow passando uma cor como parâmetro, vou utilizar dentro do arquivo importado base.
- **Ex:**

main.scss > ...

```scss
$back-color: #e20775;

@mixin box-shadow($color) {
    box-shadow: 5px 5px 5px 5px $color;
}

body {
    background-color: $back-color;
    @import "base";
}
```

https://sass-lang.com/

@MIXINS / @INCLUDE

- Utilizando o include posso chamar o mixin que foi criado no arquivo principal e passar o parâmetro solicitado.
- **Ex:**

```scss
// _base.scss > ...
.container {
    @include box-shadow(#0000ff);
    $text-color: #00ff00;
    color: $text-color;
    width: 200px;
    height: 200px;
    background-color: #fff;
    p {
        background-color: #ff0000;
        height: 100px;
        width: 100px;
        color: #0000ff;
        @include box-shadow(#0000ff);
    }
}
```

Podemos utilizar quantas vezes quisermos.


```html
<!-- index.html > ... -->
<!Doctype html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./css/main.css">
    <title>VIEW</title>
</head>
<body>

        <h1>INICIO COM SASS</h1>
            <div class="container">
                EXEMPLO ESCOPO
                <p>Somenthing</p>
            </div>

</body>
</html>
```

https://sass-lang.com/

@IF/@ELSE IF/@ELSE

- Como toda boa linguagem de programação que o sass não é ele também possui estruturas de decisão if/else if e else. Onde podemos realizar as mais varidas tarefas.

- **Ex:**

```scss
// main.scss > body
$back-color: #e20775;

@mixin box-shadow($color) {
    box-shadow: 5px 5px 5px 5px $color;
}

@mixin make-bold($bool) {
    @if $bool == true {
        font-weight: bold;
    }
}

@mixin text-effect($val) {
    @if $val == danger {
        color: #ff0000;
    } @else if $val == alert {
        color: #fff000;
    } @else {
        color: #000000;
    }
}

body {
    background-color: $back-color;
    @import "base";
}
```

```scss
// _base.scss > ...
.container {
    @include box-shadow(#0000ff);
    $text-color: #00ff00;
    color: $text-color;
    width: 200px;
    height: 200px;
    background-color: #fff;
    p {
        background-color: #1c1c1c;
        height: 100px;
        width: 100px;
        color: #0000ff;
        @include box-shadow(#0000ff);
        @include text-effect(danger);
    }
}
```

Realizamos um mixin recebendo uma variavel com um texto para ser avaliado.
Obs: Quando trabalhamos com texto no sass não utilizamos ASPAS.
O bloco padrão de validação por
If / else if /else

https://sass-lang.com/

%Placeholder

- Podemos reaproveitar blocos de códigos estáticos, onde não precisamos alterar valores
- **Ex:**

```scss
%center{
    display: flex;
    justify-content: center;
    align-items: center;
}

.div1{
    width: 500px;
    height: 250px;
    @extend %center;
}

.div2{
    width: 70%;
    height: 150px;
    @extend %center;
}
```

Repare que ele aplicou o center para as duas divs.

```css
.div2, .div1 {
  display: flex;
  justify-content: center;
  align-items: center;
}

.div1 {
  width: 500px;
  height: 250px;
}

.div2 {
  width: 70%;
  height: 150px;
}
```

https://sass-lang.com/

@FOR / [TO ,THROUGH]

- E não poderíamos deixar de ver os laços de repetição onde podemos aplicar dinamicidade em nossos códigos.
- **Ex:**

main.scss × main.css

css > main.scss > ...

```
27
28  @import "repete";
```

#{$x}

Esta interpolação deve ser usada para que se possa utilizar os valor do ponteiro do loop.

A única diferença que temos aqui entre os dois for é to não vai até o limite do contador, ele finaliza um antes e through realiza todos.

_repete.scss > ...

```
1  @for $x from 1 to 5 {
2      .text-#{$x} {
3          font-size: 16px * $x;
4      }
5  }
6
```

```
.text-1 {
  font-size: 16px;
}

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

.text-3 {
  font-size: 48px;
}

.text-4 {
  font-size: 64px;
}
```

```
.text-1 {
  font-size: 16px;
}

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

.text-3 {
  font-size: 48px;
}

.text-4 {
  font-size: 64px;
}

.text-5 {
  font-size: 80px;
}
```

_repete.scss > ...

```
1  @for $x from 1 through 5 {
2      .text-#{$x} {
3          font-size: 16px * $x;
4      }
5  }
```

https://sass-lang.com/

@FOREACH/@KEY

- Podemos criar listas aqui no sass o que pode nos auxiliar na hora de criar uma palleta de cores por exemplo.
- **Ex:**

```scss
$colors: (
    color1: blue,
    color2: red,
    color3: green,
    color4: yellow,
);

@each $key, $color in $colors {
    .#{$color}-text {
        color: $color;
    }
}
```

Aqui estamos utilizando chave para cria a lista

```scss
$colors: (blue, red, green, yellow);

@each $color in $colors {
    .#{$color}-text {
        color: $color;
    }
}
```

E aqui apenas as variáveis.

https://sass-lang.com/

@media

- Fica muito mais fácil trabalharmos com responsividade, pois podemos resolver os breakpoints dentro do próprio seletor do elemento.
- **Ex:**

```scss
div{
    width: 50%;
    min-height: 200px;
    background-color: lightblue;

    @media screen and (max-width: 768px) {
        width: 100%;
        background-color: lightcoral;
    }
}
```

No CSS ele sempre organiza o @media logo abaixo do selector

```css
div {
  width: 50%;
  min-height: 200px;
  background-color: lightblue;
}
@media screen and (max-width: 768px) {
  div {
    width: 100%;
    background-color: lightcoral;
  }
}
```

https://sass-lang.com/