Parte superior do formulário

Parte inferior do formulário

**Sass**

**O CSS com superpoderes**

**Faça esse curso de HTML e CSS e:**

* Descubra as vantagens de utilizar o pré-processador mais conhecido no mundo
* Use variáveis em seu código CSS
* Aprenda a estender seu CSS e evitar repetições
* Facilite a manutenção da folha de estilo
* Implemente funções e lógica na estilização do seu site
* Torne o site responsivo de maneira simples e reutilizável

**Aulas**

[Primeiros passos](https://cursos.alura.com.br/course/sass-css-superpoderes/section/11900/tasks)

[0 / 1042min](https://cursos.alura.com.br/course/sass-css-superpoderes/section/11900/tasks)

* + Apresentação
  + Preparando o ambiente
  + O que é Sass?
  + Sobre o Sass
  + Para saber mais: Pré-processadores
  + Estruturando o index.html
  + Faça como eu fiz: Implementando
  + Estilizando e criando variáveis
  + Comentários e sintaxe
  + O que aprendemos?

[Regras de estilo e sintaxe](https://cursos.alura.com.br/course/sass-css-superpoderes/section/11901/tasks)

[0 / 954min](https://cursos.alura.com.br/course/sass-css-superpoderes/section/11901/tasks)

* + Projeto da aula anterior
  + Nesting (Agrupamento) e Parent Selector (&)
  + Para saber mais: BEM
  + Estilizando article da home
  + Parent selector e nesting
  + Interpolation
  + Faça como eu fiz: Página “Sobre Nós”
  + Estilizando componentes da página
  + O que aprendemos?

[At Rules](https://cursos.alura.com.br/course/sass-css-superpoderes/section/11902/tasks)

[0 / 1351min](https://cursos.alura.com.br/course/sass-css-superpoderes/section/11902/tasks)

* + Projeto da aula anterior
  + Placeholder Selector (%) e Extend
  + Faça como eu fiz: Alterar imagens e títulos da sidebar
  + Mixins e Include
  + Faça como eu fiz: Criar página article.html
  + Media queries e mixins
  + Mixins
  + Faça como eu fiz: Adicionando Media Query nas páginas
  + Function
  + Faça como eu fiz: Function
  + Import
  + Para saber mais: At Rules
  + O que aprendemos?

[Controle de fluxo e laços de repetição](https://cursos.alura.com.br/course/sass-css-superpoderes/section/11903/tasks)

[0 / 1037min](https://cursos.alura.com.br/course/sass-css-superpoderes/section/11903/tasks)

* + Projeto da aula anterior
  + Iteração com @each
  + Faça como eu fiz: Substituir propriedades de margin pelos novos placeholder selectors
  + If/else + Values
  + Faça como eu fiz: Página “Collection”
  + Condicionais
  + While
  + Faça como eu fiz: Página “Contato”
  + O que aprendemos?
  + Conclusão

# 01Apresentação

SEM TRANSCRIÇÃO

# 02Preparando o ambiente

Para este curso, precisaremos instalar algumas ferramentas — caso você ainda não tenha feito isso anteriormente:

**NodeJS**: se você já fez outros cursos dessa formação, o NodeJS já deve estar instalado. você pode conferir com o comando node --version ou node -v no terminal. Caso retorne um número de versão, como por exemplo, v10.16.0, o NodeJS já está instalado. Caso precise instalar, os links estão na [página oficial](https://nodejs.org/en/download/) do NodeJS.

**Visual Studio Code**: caso precise instalar, os links estão na [página do Visual Studio](https://code.visualstudio.com/download).

Os recursos e assets do projeto estão no [GitHub](https://github.com/alura-cursos/projeto-sass) do curso.

Faremos o restante das instalações durante o desenvolvimento do projeto.

# 03O que é Sass?

SEM TRANSCRIÇÃO

# 04Sobre o Sass

Nessa primeira aula vimos sobre o Sass, o que ele é, por quê usá-lo e como fazer a instalação no nosso projeto. Assinale as alternativas corretas:



# 05Para saber mais: Pré-processadores

Pré-processadores são programas que permitem gerar CSS a partir de determinada sintaxe. Existem diversos pré-processadores, sendo que a maioria deles permite adicionar funções que não são possíveis em CSS puro, como mixins, nesting, seletores, herança, etc. É possível usar a própria sintaxe CSS mas também podemos escrever a folha de estilo com algumas particularidades, como não usar chaves e colocar elementos dentro um do outro, criando o nesting. Essas funções facilitam a manutenção e aumentam a legibilidade do CSS.

Os principais pré-processadores são **Sass**, **Less** e **Stylus**, mas existem outros menores também. Todos eles fazem coisas similares, com algumas particularidades e uma sintaxe própria. Neste curso estamos usando o Sass, um dos mais populares e também o mais antigo, criado em 2006. Sass significa Syntactically Awesome Style Sheets, e foi escrito em Ruby. O segundo pré-processador mencionado é o LESS, que foi criado 3 anos depois do Sass, em 2009, inspirado pelo mesmo, e foi escrito em JavaScript. E o último pré-processador mencionado é o Stylus, que foi lançado logo após o Less, em 2010, influenciado pelos dois anteriores, foi escrito em Node.js e combina características do Sass e do Less.

# 06Estruturando o index.html

SEM TRANSCRIÇÃO

# 07Faça como eu fiz: Implementando

Implemente o que aprendemos até agora sobre variáveis, se ainda não tiver feito isso, coloque os nomes que preferir e faça testes à vontade!

É importante colocar em prática todos os conceitos vistos até aqui, e treinar bastante!

# 08Estilizando e criando variáveis

SEM TRANSCRIÇÃO

# 09Comentários e sintaxe

SEM TRANSCRIÇÃO

# 10O que aprendemos?

Nesta aula, aprendemos:

* O que é Sass e porque usá-lo;
  + Sass é um pré-processador CSS, que permite adicionar funções que não são possíveis no CSS puro. As principais vantagens de usar o Sass são a facilidade na escrita e sintaxe, e a possibilidade de adicionar funções, mixins, nestings, seletores e herança.
* Como instalar o Sass no seu projeto;
  + Para instalar o Sass é necessário possuir o node na sua máquina, e também um editor de código. Depois, basta digitar o comando "npm install -g sass”.
* Como criar e usar variáveis no Sass;
  + Para criarmos variáveis usamos o símbolo $, junto dele o nome da variável, e depois dos dois pontos o valor que queremos. E para usá-las simplesmente chamamos o seu nome. Por exemplo: $branco: #fff; para criar e, para usá-la, chamaremos ela dentro da propriedade css, como por exemplo: color: $branco;
* Como comentar o código no Sass;
  + Como em outras linguagens de programação, também é possível fazer comentários no Sass. Existem dois tipos: com duas barras (//) ou entre barras e asteriscos (/\*\*/), a diferença entre eles é que no primeiro caso o comentário é por linha, ou seja, caso o enter seja pressionado, o comentário não será continuado, já o segundo permite comentário em múltiplas linhas.

# 01Projeto da aula anterior

Aqui você pode visualizar e baixar o projeto inicial do curso.

Visualize os arquivos: [reset.css](https://github.com/alura-cursos/projeto-sass/blob/main/css/reset.css), [index.html](https://github.com/alura-cursos/projeto-sass/blob/main/index.html) e [style.scss](https://github.com/alura-cursos/projeto-sass/blob/main/scss/style.scss" \t "_blank), no Github!

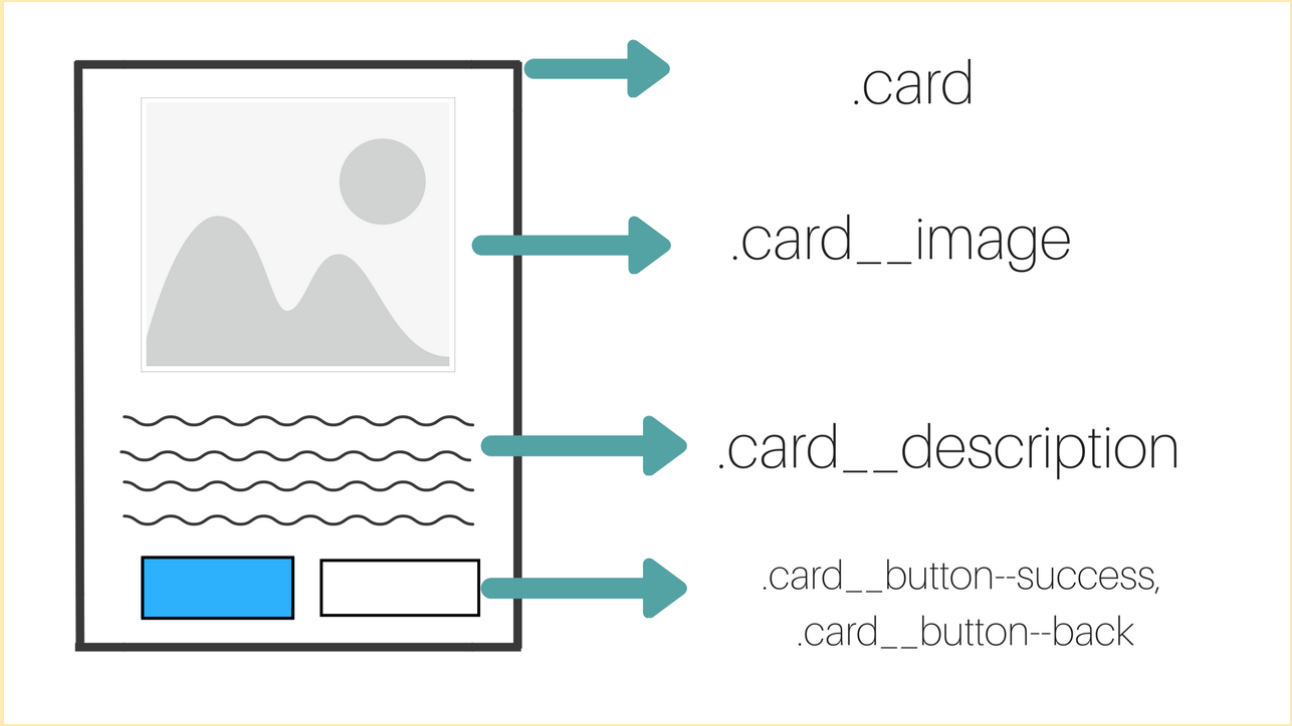
# 02Nesting (Agrupamento) e Parent Selector (&)

SEM TRANSCRIÇÃO

# 03Para saber mais: BEM

CSS é simples de se aprender e complexo de ser mantido. Conforme um projeto cresce, a manutenção do CSS vai ficando cada vez mais difícil, e para isso foram surgindo soluções, e o **BEM** é uma delas. BEM é um padrão de nomenclatura para manter o projeto simples e organizado. Ele foi criado pela Yandex e tenta resolver o problema de nomenclatura e estrutura do CSS permitindo facilitar e escalar o código, de forma que qualquer desenvolvedor tenha autonomia para mexer em qualquer projeto. A sigla BEM significa **Block Element Modifier**, em português, Bloco, Elemento, Modificador. Esses três pilares são a base dessa metodologia, e também as categorias em que os elementos são divididos.

Vamos supor que você queira construir um componente de card. Nesse caso, o bloco seria o .card. E qualquer outra seção dentro do card seriam elementos. No exemplo, temos uma imagem, uma descrição e um botão. A convenção da nomenclatura do BEM conecta o bloco ao elemento com dois underlines, por exemplo .card\_\_image. Ainda na nossa imagem, nós temos dois tipos de botões, eles seriam os modificadores, e nesse caso para a nomenclatura nós conectamos ao elemento com dois hífens, por exemplo .card\_\_button--success e .card\_\_button--back.



Mas como saber o que é um elemento e o que é um modificador?

Um elemento sempre será uma parte da estrutura do bloco, já os modificadores são estados que os nossos blocos ou elementos poderão ter, como botões com diferentes aparências ou uma situação de destaque.

**Prós e contras**:

BEM deixa o nosso CSS com uma estrutura modular. Por conta dos nomes únicos, não acontecerão conflitos entre as classes, o que facilita a manutenção. Porém, às vezes, os nomes das classes ficam muito compridos ao se usar a metodologia, mas com o uso de um pré-processador (como estamos usando no curso), isso não será um grande problema (pois usaremos o nesting).

Referências: [“What is BEM? And why you should use it in your project”, texto em inglês por Danny Huang](https://medium.com/@dannyhuang_75970/what-is-bem-and-why-you-should-use-it-in-your-project-ab37c6d10b79) e [“BEM em 5min”, em português, por Julio Lozovei](https://medium.com/trainingcenter/bem-em-5min-f5c80fd23439).

# 04Estilizando article da home

SEM TRANSCRIÇÃO

# 05Parent selector e nesting

Luiza está desenvolvendo o seu primeiro site, depois de ter feito algumas páginas ela instalou o Sass no seu projeto e vai precisar refatorar o CSS para que fique de acordo com a metodologia.

.component {

width: 100%;

padding: 20px 0;

}

.component\_\_logo {

max-width: 100px;

}

.component\_\_txt {

margin: auto;

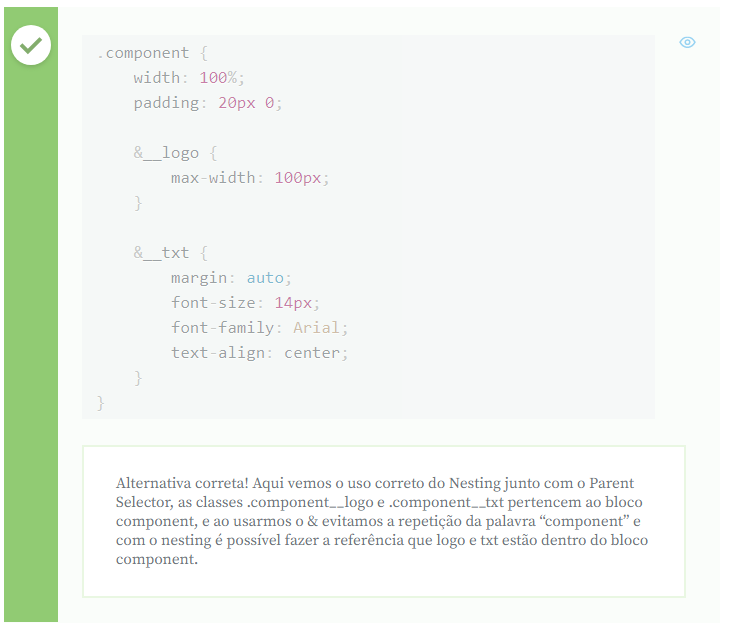
font-size: 14px;

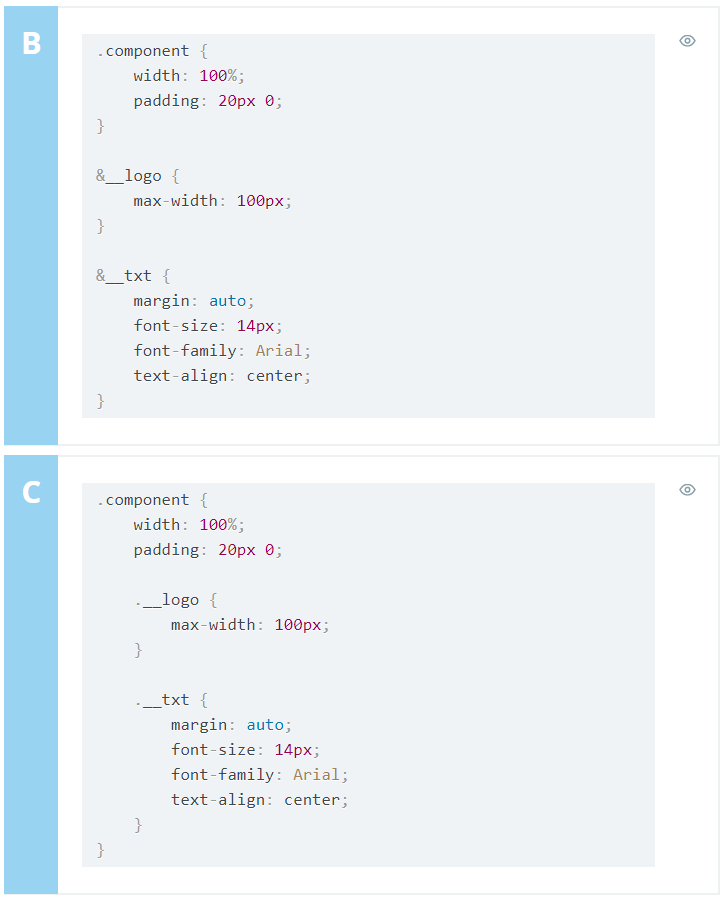
font-family: Arial;

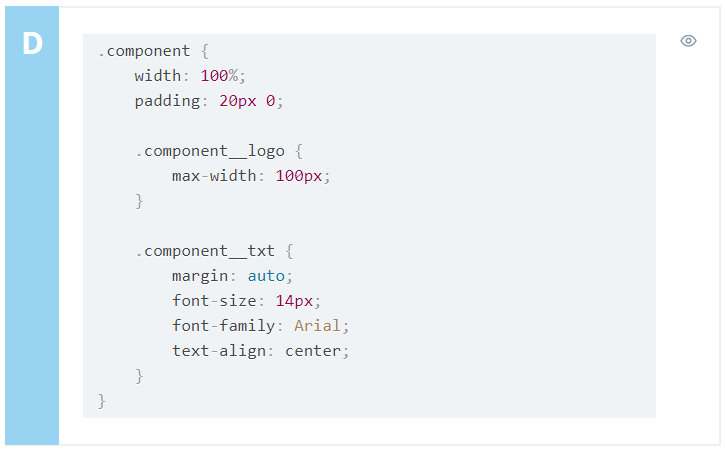
text-align: center;

}COPIAR CÓDIGO

Dado o seguinte trecho de código acima, como ficará a refatoração correta do código usando Nesting e Parent selector corretamente?







# 06Interpolation

SEM TRANSCRIÇÃO

# 07Faça como eu fiz: Página “Sobre Nós”

Vamos utilizar as Regras de estilo e Sintaxe para criar uma página Sobre Nós no nosso site. A primeira coisa que vamos fazer é criar o arquivo para a página e linkar o nosso menu a este arquivo. No nosso VS Code vamos criar um arquivo chamado “about.html” e dentro dele vamos copiar a estrutura base do html de todo nosso site, todo o nosso index html, e colar no arquivo about, e antes de salvar vamos retirar toda a tag <main>. Depois, vamos para o index.html e alterar o href do link Sobre nós, adicionando o about.html, e então fazer a mesma coisa no nosso novo arquivo. Ao salvarmos e irmos para o browser, podemos clicar no link Sobre nós, e ele vai redirecionar para nossa nova página ainda vazia, apenas com o header e o footer. Vamos criar o conteúdo dela. Voltando para o vscode, embaixo da tag header vamos criar uma tag article com a classe “about”. Dentro do article teremos a div “about\_\_title” com um h1 Sobre Nós e mais uma div com o breadcrumb, que nada mais é que uma lista de links que redirecionam o usuário para as páginas navegadas anteriormente.

Primeiro temos um <a> que vai redirecionar de volta para a home, depois uma imagem que será o ícone de uma seta e mais um <a> que é a própria página Sobre Nós. Abaixo dessa div do título, teremos a div do conteúdo, primeiro com uma imagem principal e um texto de introdução. E depois mais uma div, chamada about\_\_split, onde vamos dividir o conteúdo ao meio, tendo uma imagem menor do lado esquerdo e um texto do lado direito com 3 parágrafos. Ao irmos para o navegador e recarregarmos a página, todos os componentes estarão lá, porém, nada organizados. Por isso precisamos do nosso css, ou no nosso caso, Sass. Vamos no nosso arquivo style.scss e adicionar o seletor da classe about, dentro dele teremos um width de 80%, o font family Arial e margin: 0 auto 40px. Para o título, vamos colocar um background cinza claro, alinhar o texto no centro e incluir um padding de 20px. Para o H1 teremos um font size de 22px, a cor será o cinza escuro, padrão dos nossos títulos, o espaçamento entre as letras será de 1px, margin top de 5px, e margin bottom de 10px.

Nosso próximo passo é estilizar o breadcrumb, para isso vamos chamar a classe, definir um display flex e alinhar ao centro. Para o link, colocaremos um font size de 12px, tiraremos o estilo padrão do link, e a cor também será o cinza escuro. Vamos adicionar um hover roxo para o usuário saber que é um link ao passar o mouse em cima dos textos, e definir um tamanho fixo para o ícone da setinha, de 10px, e margin de 8px para o espaçamento entre os textos e a imagem. Fora do breadcrumb, a imagem principal da página terá width de 100%, um margin bottom de 20px e o texto de introdução terá fonte de 16px, cor cinza escuro, peso da fonte de 100, largura de 70%, sendo menor que a largura da imagem, alinhamento do texto no centro e margin top e bottom de 30px.

Vamos salvar e atualizar o navegador com tudo que fizemos até agora. Para definirmos a divisão da nossa div que possui imagem e texto, vamos usar o display flex, flex wrap e alinhamento ao centro na div principal. Antes de dividirmos a imagem e o texto, vamos setar o estilo dos parágrafos com padding 10px e 20px e tamanho da fonte de 14px. Agora sim, para a divisão da nossa div nós vamos usar a interpolação, mas antes de tudo vamos ver o que ela é. Interpolação é usada para injetar valores em expressões como variáveis e funções nos nossos seletores. Isso pode ser muito útil ao se passar parâmetros que podem ser alterados.

No caso da nossa divisão nós vamos usar um cálculo do css para a largura da div do nosso texto, vamos fazer sem a interpolação primeiro, definindo a largura da imagem 45%. A largura do texto será o resultado da largura total menos a largura da imagem, ficando “width: calc(100% - 45%);”, porém, como já vimos, é muito mais interessante usarmos variáveis em valores que podem ser alterados caso haja alguma necessidade, então vamos transformar os 45% em uma variável “$img-width: 45%;”, substituir no estilo da imagem e usar a interpolação para chamá-la na div do texto, ficando: “width: calc(100% - #{$img-width});”. Ao recarregarmos o navegador podemos ver que o cálculo funcionou e conseguimos dividir a imagem de um lado e o texto do outro.

Abaixo temos o resultado final do código da página Sobre nós.

## HTML

<!DOCTYPE html>

<html>

<head>

<title>Gatito PetShop</title>

<meta name="viewport" content="width=device-width, initial-scale=1" />

<link rel="stylesheet" href="css/style.css">

<link rel="stylesheet" href="css/reset.css">

</head>

<body>

<header class="header">

<figure class="header\_\_logo">

<img src="assets/img/logo.png" alt="Logo" class="header\_\_img" />

</figure>

<nav class="header\_\_menu menu">

<ul class="menu\_\_list">

<li class="menu\_\_links"><a href="index.html" class="menu\_\_link">Página inicial</a></li>

<li class="menu\_\_links"><a href="about.html" class="menu\_\_link">Sobre nós</a></li>

<li class="menu\_\_links"><a href="collection.html" class="menu\_\_link">Cães</a></li>

<li class="menu\_\_links"><a href="collection.html" class="menu\_\_link">Gatos</a></li>

<li class="menu\_\_links"><a href="collection.html" class="menu\_\_link">Pássaros</a></li>

<li class="menu\_\_links"><a href="contact.html" class="menu\_\_link">Contato</a></li>

</ul>

</nav>

</header>

<article class="about">

<div class="about\_\_title">

<h1 class="about\_\_h1">Sobre nós</h1>

<div class="about\_\_breadcrumb breadcrumb">

<a href="index.html" class="breadcrumb\_\_link">Home</a>

<img src="assets/img/arrow.svg" alt="Seta" class="breadcrumb\_\_img"/>

<a href="about.html" class="breadcrumb\_\_link">Sobre nós</a>

</div>

</div>

<div class="about\_\_content">

<img src="assets/img//dog-cat.jpg" alt="Dog Cat" class="about\_\_img" />

<p class="about\_\_intro">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam viverra velit massa, quis semper nunc consectetur sed. Pellentesque vehicula cursus blandit. Praesent porttitor tempor magna sit amet maximus. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis pellentesque, nulla at feugiat bibendum, metus nunc gravida lorem, nec ultricies velit nulla nec justo. Fusce quis vestibulum eros. Pellentesque habitant morbi.</p>

<div class="about\_\_split">

<img src="assets/img/about.png" alt="Gatito" class="about\_\_img2" />

<div class="about\_\_txt">

<p class="about\_\_p">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam viverra velit massa, quis semper nunc consectetur sed. Pellentesque vehicula cursus blandit. Praesent porttitor tempor magna sit amet maximus. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis pellentesque, nulla at feugiat bibendum, metus nunc gravida lorem, nec ultricies velit nulla nec justo. Fusce quis vestibulum eros. Pellentesque habitant morbi.</p>

<p class="about\_\_p">Praesent faucibus dui sed nulla egestas lobortis. Integer molestie nunc in turpis sagittis commodo. Morbi diam nulla, efficitur nec euismod pretium, lobortis egestas erat. Nullam at laoreet arcu, sed luctus sapien. Aliquam porta vehicula lorem. Donec laoreet eget tellus et finibus. Vivamus semper consequat metus ac blandit. Maecenas non purus eget metus rhoncus eleifend. In hac habitasse platea dictumst. Nulla ut lacinia augue, vehicula ullamcorper tellus.</p>

<p class="about\_\_p">Donec in dictum elit. Vestibulum pulvinar vitae orci in malesuada. Morbi facilisis mi quis tortor tincidunt consequat. Quisque luctus imperdiet ultrices. Aliquam pretium, velit vel consequat sollicitudin, massa tortor pharetra elit, vitae consequat leo dolor non felis. Donec sodales scelerisque massa, sagittis maximus orci elementum venenatis. Vestibulum ornare sollicitudin metus, sed lacinia ligula blandit eget. Aenean at magna ut arcu dictum accumsan a at felis.</p>

</div>

</div>

</div>

</article>

<footer class="footer">

<img src="assets/img/footer-logo.png" alt="Footer logo" class="footer\_\_logo" />

<p class="footer\_\_txt">© 2020 Gatito Blog . All rights reserved.</p>

</footer>

</body>

</html>COPIAR CÓDIGO

## SCSS

.about {

@include for-phone-only {

width: 90%;

}

width: 80%;

margin: 0 auto 40px;

font-family: Arial;

&\_\_title {

background-color: $light-grey;

text-align: center;

padding: 20px;

}

&\_\_h1 {

@include fontSize(22px);

color: $dark-grey;

letter-spacing: 1px;

margin-top: 5px;

margin-bottom: 10px;

margin-left: 0;

margin-right: 0;

}

.breadcrumb {

@include flx(justify-content, center);

align-items: center;

&\_\_link {

@include fontSize(12px);

text-decoration: unset;

color: $dark-grey;

&:hover {

text-decoration: underline;

color: $purple;

}

}

&\_\_img {

width: 10px;

margin: 8px;

}

}

&\_\_img {

@extend %double-margin-bottom;

width: 100%;

}

&\_\_intro {

font-weight: 100;

color: $dark-grey;

width: 70%;

margin: 30px auto;

text-align: center;

@include fontSize(16px);

@include for-phone-only {

width: 90%;

}

@include for-tablet-only {

width: 80%;

}

}

&\_\_split {

@include flx(justify-content, center);

flex-wrap: wrap;

align-items: center;

}

&\_\_img2 {

width: $img-width;

@include for-phone-only {

width: 80%;

}

}

&\_\_txt {

width: calc(100% - #{$img-width});

@include for-phone-only {

width: 90%;

text-align: center;

}

}

&\_\_p {

padding: 10px 20px;

@include fontSize(14px);

}

}COPIAR CÓDIGO

# 08Estilizando componentes da página

SEM TRANSCRIÇÃO

# 09O que aprendemos?

Nesta aula, aprendemos:

* O que é Nesting, e quais as suas vantagens em comparação ao CSS.
  + A tradução literal de Nesting é “aninhamento”, assim, ele permite colocarmos elementos um dentro do outro, evitando a repetição, facilitando na manutenção e deixando o código CSS mais curto.
* Como usar o Parent Selector.
  + O Parent Selector é um seletor & que permite referenciar o elemento “pai”. Sendo possível adicionar pseudo-classes como &:hover, pseudo\_elementos como &:before, e também sufixos para as classes. No CSS uma classe que seria: .component .component\_\_text {}, no Sass fica: .component { &\_\_text {} }.
* O que é Interpolação e como fazê-lo.
  + Interpolação é usada para injetar valores em expressões como variáveis e funções nos nossos seletores. Isso pode ser muito útil ao se passar parâmetros que podem ser alterados.

# 01Projeto da aula anterior

Aqui você pode visualizar e baixar o projeto inicial do curso.

Visualize os arquivos: [style.scss](https://github.com/alura-cursos/projeto-sass/blob/main/scss/style.scss" \t "_blank) e [main.about](https://github.com/alura-cursos/projeto-sass/blob/main/about.html" \t "_blank) no Github!

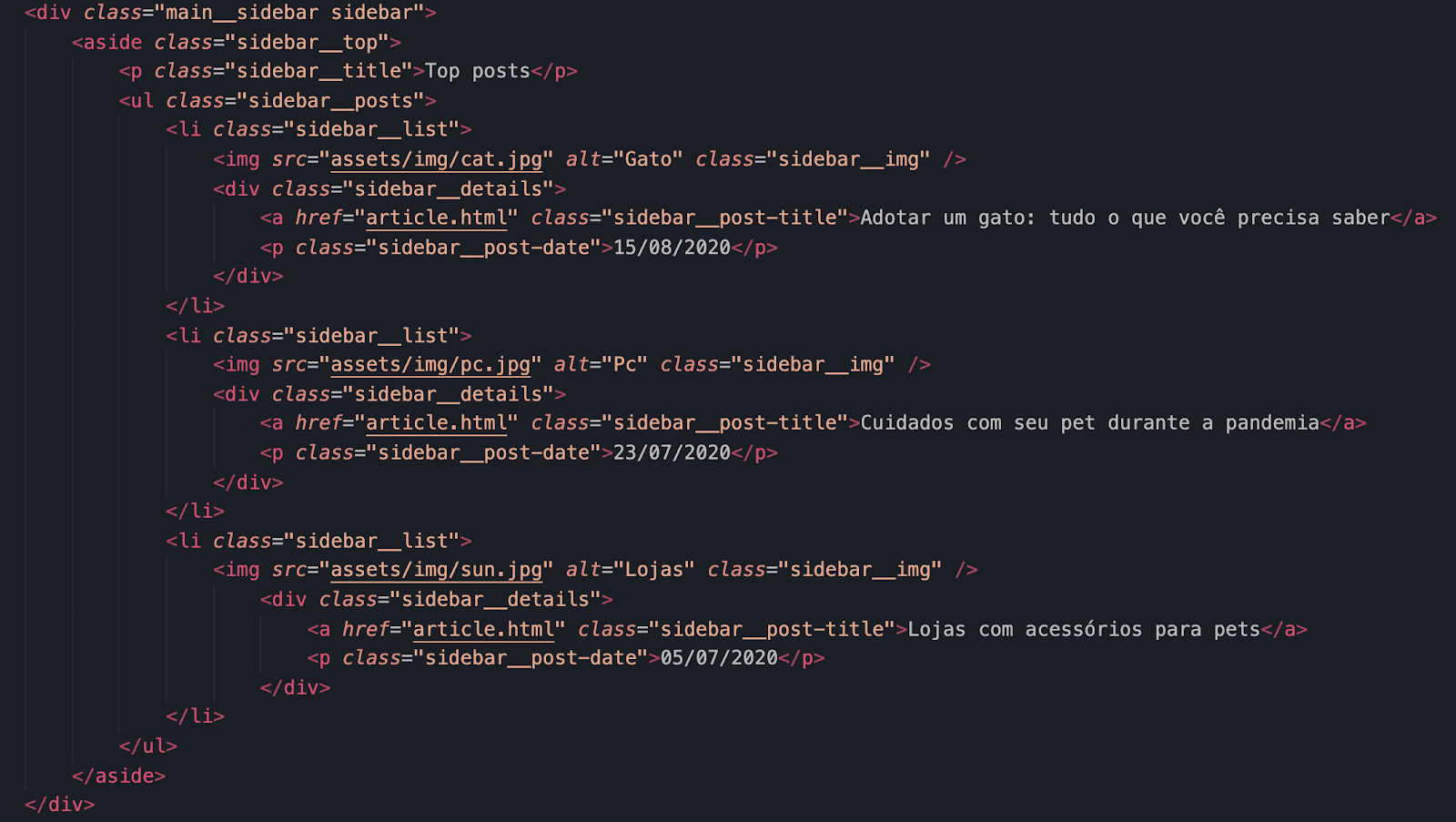
# 02Placeholder Selector (%) e Extend

SEM TRANSCRIÇÃO

# 03Faça como eu fiz: Alterar imagens e títulos da sidebar

Durante a aula 3.1 criamos a sidebar e adicionamos um estilo padrão para ela, porém deixamos os 3 posts iguais, agora vamos alterar os dois últimos posts para que eles fiquem com um conteúdo diferente do primeiro. Você pode colocar a imagem e o título que preferir.

Aqui abaixo vou deixar um código de exemplo com a alteração que fiz:



# 04Mixins e Include

SEM TRANSCRIÇÃO

# 05Faça como eu fiz: Criar página article.html

Nessa aula criamos a página do artigo do nosso blog, nela é onde ficam os detalhes e imagens dos posts que são criados no nosso blog.

No nosso vscode, vamos criar um novo arquivo chamado article.html e dentro dele podemos copiar o conteúdo da página about e remover tudo que tem dentro da tag <article>, vamos alterar o nome da classe para “article” e adicionar uma div dentro chamada article\_\_title para o título da nossa página. Dentro dela vamos colocar um h1 para o título do post e uma div para o breadcrumb da página e dentro desse breadcrumb teremos o link da home, o ícone de uma setinha e o link da página atual.

Fora desse div do título vamos adicionar a imagem do post logo abaixo com a classe article\_\_img. Depois uma div para os detalhes do post e dentro duas tags de <span> para data e tag do blog. E por último uma div para o conteúdo do post chamada article\_\_content e dentro dela teremos vários parágrafos, como já fizemos em aulas anteriores o texto usado aqui também vai ser em lorem ipsum, apenas como placeholder.

Abaixo temos o código final do HTML da página article.html:

<!DOCTYPE html>

<html>

<head>

<title>Gatito PetShop</title>

<meta name="viewport" content="width=device-width, initial-scale=1" />

<link rel="stylesheet" href="css/style.css">

<link rel="stylesheet" href="css/reset.css">

</head>

<body>

<header class="header">

<figure class="header\_\_logo">

<img src="assets/img/logo.png" alt="Logo" class="header\_\_img" />

</figure>

<nav class="header\_\_menu menu">

<ul class="menu\_\_list">

<li class="menu\_\_links"><a href="index.html" class="menu\_\_link">Página inicial</a></li>

<li class="menu\_\_links"><a href="about.html" class="menu\_\_link">Sobre nós</a></li>

<li class="menu\_\_links"><a href="collection.html" class="menu\_\_link">Cães</a></li>

<li class="menu\_\_links"><a href="collection.html" class="menu\_\_link">Gatos</a></li>

<li class="menu\_\_links"><a href="collection.html" class="menu\_\_link">Pássaros</a></li>

<li class="menu\_\_links"><a href="contact.html" class="menu\_\_link">Contato</a></li>

</ul>

</nav>

</header>

<article class="article">

<div class="article\_\_title">

<h1 class="article\_\_h1">Cuidados com os animais durante a quarentena</h1>

<div class="article\_\_breadcrumb breadcrumb">

<a href="index.html" class="breadcrumb\_\_link">Home</a>

<img src="assets/img/arrow.svg" alt="Seta" class="breadcrumb\_\_img" />

<a href="article.html" class="breadcrumb\_\_link">Cuidados com os animais durante a quarentena</a>

</div>

</div>

<img src="assets/img/dog.jpg" alt="Cachorro" class="article\_\_img" />

<div class="article\_\_details">

<span class="article\_\_date">20/08/2020</span>

<span class="article\_\_tag">Cães</span>

</div>

<div class="article\_\_content">

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam viverra velit massa, quis semper nunc consectetur sed. Pellentesque vehicula cursus blandit. Praesent porttitor tempor magna sit amet maximus. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis pellentesque, nulla at feugiat bibendum, metus nunc gravida lorem, nec ultricies velit nulla nec justo. Fusce quis vestibulum eros. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Nunc cursus fringilla blandit. Donec gravida nibh et nibh luctus porta. In hac habitasse platea dictumst. Donec vel mollis ligula, eu tristique velit.</p>

<p>Praesent faucibus dui sed nulla egestas lobortis. Integer molestie nunc in turpis sagittis commodo. Morbi diam nulla, efficitur nec euismod pretium, lobortis egestas erat. Nullam at laoreet arcu, sed luctus sapien. Aliquam porta vehicula lorem. Donec laoreet eget tellus et finibus. Vivamus semper consequat metus ac blandit. Maecenas non purus eget metus rhoncus eleifend. In hac habitasse platea dictumst. Nulla ut lacinia augue, vehicula ullamcorper tellus.</p>

<p>Donec in dictum elit. Vestibulum pulvinar vitae orci in malesuada. Morbi facilisis mi quis tortor tincidunt consequat. Quisque luctus imperdiet ultrices. Aliquam pretium, velit vel consequat sollicitudin, massa tortor pharetra elit, vitae consequat leo dolor non felis. Donec sodales scelerisque massa, sagittis maximus orci elementum venenatis. Vestibulum ornare sollicitudin metus, sed lacinia ligula blandit eget. Aenean at magna ut arcu dictum accumsan a at felis.</p>

<p>Curabitur non nibh ac est suscipit ornare placerat non nisi. Phasellus eleifend, leo eu molestie pharetra, nunc erat cursus libero, nec lacinia turpis tellus et nulla. Phasellus commodo leo vel tortor ultricies, eu efficitur diam pellentesque. Ut dolor velit, tincidunt id quam quis, fermentum faucibus nibh. Aliquam tincidunt rhoncus felis. Ut nisi dolor, consequat quis facilisis at, efficitur sit amet neque. Sed tortor dolor, interdum ac mi in, dignissim pretium mauris.</p>

<p>Morbi quis imperdiet erat. Nunc vel magna aliquam, posuere mauris sed, auctor mauris. Quisque eget nibh at lacus sagittis dapibus. Donec placerat imperdiet odio at ultrices. Aliquam erat volutpat. Ut egestas pulvinar feugiat. In pellentesque est sed nibh egestas, sed ullamcorper felis tincidunt. Aenean nec dictum tortor. Suspendisse quis fermentum metus, vel viverra odio. Pellentesque ut blandit dui. Suspendisse in interdum erat, at lobortis nulla. Aenean posuere dictum mauris, a auctor elit eleifend vel.</p>

</div>

</article>

<footer class="footer">

<img src="assets/img/footer-logo.png" alt="Footer logo" class="footer\_\_logo" />

<p class="footer\_\_txt">© 2020 Gatito Blog . All rights reserved.</p>

</footer>

</body>

</html>COPIAR CÓDIGO

# 06Media queries e mixins

SEM TRANSCRIÇÃO

# 07Mixins

Abaixo temos afirmações sobre mixins, assinale as **alternativas corretas**:



# 08Faça como eu fiz: Adicionando Media Query nas páginas

Agora que aprendemos como criar e usar Media Query com mixins, vamos deixar o layout das nossas outras páginas responsivas, primeiro, na página de Sobre nós vamos adicionar um width 90% para celular e, no intro, um width de 90% para o celular e 80% para tablet. Na img2 vamos setar um width 80% para o celular e para o texto width de 90% e texto alinhado ao centro, ficando assim:

.about {

@include for-phone-only {

width: 90%;

}

width: 80%;

margin: 0 auto 40px;

font-family: Arial;

}

&\_\_intro {

font-weight: 100;

color: $dark-grey;

width: 70%;

margin: 30px auto;

text-align: center;

font-size: 16px;

@include for-phone-only {

width: 90%;

}

@include for-tablet-only {

width: 80%;

}

}

&\_\_img2 {

width: $img-width;

@include for-phone-only {

width: 80%;

}

}

&\_\_txt {

width: calc(100% - #{$img-width});

@include for-phone-only {

width: 90%;

text-align: center;

}

}COPIAR CÓDIGO

Agora para a página do Artigo um width de 90% para celular, conforme está o código abaixo:

.article {

@include for-phone-only {

width: 90%;

}

width: 80%;

margin: 0 auto 40px;

font-family: Arial;

}COPIAR CÓDIGO

Continue com os seus estudos, e se houver dúvidas, não hesite em recorrer ao nosso fórum!

# 09Function

SEM TRANSCRIÇÃO

# 10Faça como eu fiz: Function

Vamos usar a At Rule @function para criar uma função que transforma pixels em REM, pois o rem é mais acessível que os pixels e funciona melhor entre os diferentes browsers. Para transformar pixels em rem é necessário fazer uma operação matemática, e é justamente isso que nossa função vai fazer.

Em nosso arquivo style.scss, abaixo dos mixins de layout responsivo, vamos criar nossa function usando o @function e atribuindo um nome, que no caso será “calculateRem”. Depois disso, vamos passar uma variável como parâmetro, que será o tamanho que vamos transformar de px em rem, e abrir as chaves para o conteúdo da nossa função, ficando então @function calculateRem($size) { }.

Para o conteúdo, adicionaremos o return de um cálculo matemático onde: $size / 16px \* 1rem. Nossa função fica @function calculateRem($size) { @return $size / 16px \* 1rem; }.

Essa função pode ser chamada diretamente em qualquer lugar que você queira transformar px em rem.

Resultado da function criada:

@function calculateRem($size) {

@return $size / 16px \* 1rem;

}COPIAR CÓDIGO

Agora podemos substituir todos os lugares que temos a propriedade font-size pela função criada, ficando:

@include fontSize(22px);COPIAR CÓDIGO

# 11Import

SEM TRANSCRIÇÃO

# 12Para saber mais: At Rules

Grande parte das funções extras do Sass vem no formato de **At rules**, e nesta aula vimos alguns exemplos como mixins, function, import, extend e include, mas existem diversas At Rules:

@use: carrega mixins, functions e variáveis de outras folhas de estilos Sass e combina o CSS de diversas folhas de estilo juntos.

@forward: carrega uma folha de estilo Sass e torna os mixins, functions e variáveis disponíveis quando a folha de estilo é carregada pela regra do @use.

@import: estende as regras de CSS para carregar styles, mixins, functions e variáveis de outras folhas de estilo.

@mixins e @include: facilitam a reutilização de trechos de código.

@function: define funções customizadas que podem ser utilizadas em expressões.

@extend: permite que os seletores herdem estilos uns dos outros.

@at-root: coloca estilos dentro dele na raiz do documento CSS.

@error: faz com que a compilação falhe com uma mensagem de erro.

@warn: imprime um aviso sem parar totalmente a compilação.

@debug: imprime uma mensagem para fins de debugging.

E também fluxos de controle, como: @if, @each, @for e @while.

# 13O que aprendemos?

Nesta aula, aprendemos:

* Como criar um placeholder selector e como usá-lo com extend.
  + O Placeholder Selector é um tipo especial de seletor que é bem parecido com um seletor convencional mas com um % na frente e ele também não é exibido no arquivo CSS após ser compilado.
* O que são os mixins.
  + Mixins permitem definir estilos que podem ser reutilizados, e isso evita o uso de classes repetitivas e não semânticas como .float-left, e também distribui coleções de estilo para bibliotecas.
* Como deixar o site responsivo usando media queries com mixins.
  + Os media queries já são usados no CSS convencional, porém, com os mixins, é possível setar seus valores em apenas um lugar e reutilizá-lo em todo o código e, caso seja necessário, a alteração dos valores será feita em apenas um lugar.
* Como criar funções no Sass.
  + As funções no Sass permitem definir operações complexas que podem ser reutilizadas em todo o projeto, facilitando abstração de comportamentos em comum.
* Como importar arquivos no Sass.
  + Vimos como abstrair o nosso CSS, deixando ele separado para cada uma das páginas. Dessa maneira a manutenção fica mais fácil e, para os CSS abstraídos funcionarem, nós usamos o @import, que importa os arquivos dentro de um principal.

# 01Projeto da aula anterior

Aqui você pode visualizar e baixar o projeto inicial do curso.

Visualize os arquivos: [index.html](https://github.com/alura-cursos/projeto-sass/blob/main/index.html), [style.scss](https://github.com/alura-cursos/projeto-sass/blob/main/scss/style.scss), [article.html](https://github.com/alura-cursos/projeto-sass/blob/main/article.html), [\_header.scss](https://github.com/alura-cursos/projeto-sass/blob/main/scss/_header.scss), [\_footer.scss](https://github.com/alura-cursos/projeto-sass/blob/main/scss/_footer.scss), [\_article.scss](https://github.com/alura-cursos/projeto-sass/blob/main/scss/_article.scss) e [\_about.scss](https://github.com/alura-cursos/projeto-sass/blob/main/scss/_about.scss) no Github!

# 02Iteração com @each

SEM TRANSCRIÇÃO

# 03Faça como eu fiz: Substituir propriedades de margin pelos novos placeholder selectors

Durante a aula nós utilizamos os placeholder selectores criados dentro do @each nas margens da nossa página inicial, agora vamos para a página de Sobre nós.

Nossa imagem tem um margin bottom 20px, então podemos substituir por @extend %double-margin-bottom, lembrando que apesar de os valores terem sido criados no arquivo **style.scss** eles funcionam no about, por conta da importação.

E por último no arquivo do article temos @extend %half-margin-top, %default-margin-bottom, %no-margin-left, %no-margin-right; no H1, @extend %double-margin-bottom; para a img, @extend %no-margin-top, %no-margin-bottom, %default-margin-left, %default-margin-right; para a data e para a tag.

O código final para cada uma das páginas fica conforme está abaixo:

\_article.scss:

.article {

@include for-phone-only {

width: 90%;

}

width: 80%;

margin: 0 auto 40px;

font-family: Arial;

&\_\_title {

background-color: $light-grey;

text-align: center;

padding: 20px;

}

&\_\_h1 {

@include fontSize(22px);

color: $dark-grey;

letter-spacing: 1px;

@extend %half-margin-top, %default-margin-bottom, %no-margin-left, %no-margin-right;

}

.breadcrumb {

@include flx(justify-content, center);

align-items: center;

&\_\_link {

@include fontSize(12px);

@extend %no-decoration;

color: $dark-grey;

&:hover {

@extend %u-decoration;

color: $purple;

}

}

&\_\_img {

width: 10px;

margin: 8px;

}

}

&\_\_img {

@extend %double-margin-bottom;

width: 100%;

}

&\_\_details {

@include flx(justify-content, center);

align-items: center;

}

&\_\_date {

@extend %no-margin-top, %no-margin-bottom, %default-margin-left, %default-margin-right;

@include fontSize(12px);

font-style: italic;

}

&\_\_tag {

@include fontSize(14px);

padding: 5px 7px;

color: $purple;

border: 1px solid $purple;

margin: 0 10px;

}

&\_\_content {

font-weight: 100;

@include fontSize(14px);

color: $dark-grey;

padding: 10px 20px;

}

}COPIAR CÓDIGO

\_about.scss:

.about {

@include for-phone-only {

width: 90%;

}

width: 80%;

margin: 0 auto 40px;

font-family: Arial;

&\_\_title {

background-color: $light-grey;

text-align: center;

padding: 20px;

}

&\_\_h1 {

@include fontSize(22px);

color: $dark-grey;

letter-spacing: 1px;

margin-top: 5px;

margin-bottom: 10px;

margin-left: 0;

margin-right: 0;

}

.breadcrumb {

@include flx(justify-content, center);

align-items: center;

&\_\_link {

@include fontSize(12px);

text-decoration: unset;

color: $dark-grey;

&:hover {

text-decoration: underline;

color: $purple;

}

}

&\_\_img {

width: 10px;

margin: 8px;

}

}

&\_\_img {

@extend %double-margin-bottom;

width: 100%;

}

&\_\_intro {

font-weight: 100;

color: $dark-grey;

width: 70%;

margin: 30px auto;

text-align: center;

@include fontSize(16px);

@include for-phone-only {

width: 90%;

}

@include for-tablet-only {

width: 80%;

}

}

&\_\_split {

@include flx(justify-content, center);

flex-wrap: wrap;

align-items: center;

}

&\_\_img2 {

width: $img-width;

@include for-phone-only {

width: 80%;

}

}

&\_\_txt {

width: calc(100% - #{$img-width});

@include for-phone-only {

width: 90%;

text-align: center;

}

}

&\_\_p {

padding: 10px 20px;

@include fontSize(14px);

}

}COPIAR CÓDIGO

# 04If/else + Values

SEM TRANSCRIÇÃO

# 05Faça como eu fiz: Página “Collection”

Nessa aula criamos a página de collection do nosso blog, nela é onde ficam todos os posts do nosso blog em lista. Como ficou o seu código?

Abaixo temos todo o código html da nossa página collection.html:

<!DOCTYPE html>

<html>

<head>

<title>Gatito PetShop</title>

<meta name="viewport" content="width=device-width, initial-scale=1" />

<link rel="stylesheet" href="css/style.css">

<link rel="stylesheet" href="css/reset.css">

</head>

<body>

<header class="header">

<figure class="header\_\_logo">

<img src="assets/img/logo.png" alt="Logo" class="header\_\_img" />

</figure>

<nav class="header\_\_menu menu">

<ul class="menu\_\_list">

<li class="menu\_\_links"><a href="index.html" class="menu\_\_link">Página inicial</a></li>

<li class="menu\_\_links"><a href="about.html" class="menu\_\_link">Sobre nós</a></li>

<li class="menu\_\_links"><a href="collection.html" class="menu\_\_link">Cães</a></li>

<li class="menu\_\_links"><a href="collection.html" class="menu\_\_link">Gatos</a></li>

<li class="menu\_\_links"><a href="collection.html" class="menu\_\_link">Pássaros</a></li>

<li class="menu\_\_links"><a href="contact.html" class="menu\_\_link">Contato</a></li>

</ul>

</nav>

</header>

<div class="collection">

<div class="collection\_\_breadcrumb breadcrumb">

<a href="index.html" class="breadcrumb\_\_link">Home</a>

<img src="assets/img/arrow.svg" alt="Seta" class="breadcrumb\_\_img" />

<a href="collection.html" class="breadcrumb\_\_link">Posts</a>

</div>

<div class="collection\_\_post">

<figure class="collection\_\_figure">

<img src="assets/img/dog.jpg" alt="Cachorro" class="collection\_\_img" />

</figure>

<div class="collection\_\_about">

<a href="article.html" class="collection\_\_title">Cuidados com os animais durante a quarentena</a>

<p class="collection\_\_date">20/08/2020</p>

<span class="collection\_\_tag">Cães</span>

</div>

</div>

<div class="collection\_\_post half">

<figure class="collection\_\_figure">

<img src="assets/img/party.jpg" alt="Cachorro festa" class="collection\_\_img" />

</figure>

<div class="collection\_\_about">

<a href="article.html" class="collection\_\_title half">Como manter a saúde do seu pet</a>

<p class="collection\_\_date">09/08/2020</p>

<span class="collection\_\_tag">Cães</span>

</div>

</div>

<div class="collection\_\_post half">

<figure class="collection\_\_figure">

<img src="assets/img/car.jpg" alt="Cachorro carro" class="collection\_\_img" />

</figure>

<div class="collection\_\_about">

<a href="article.html" class="collection\_\_title half">Como transportar seu pet com segurança</a>

<p class="collection\_\_date">23/07/2020</p>

<span class="collection\_\_tag">Cães</span>

</div>

</div>

<div class="collection\_\_post">

<figure class="collection\_\_figure">

<img src="assets/img/sun.jpg" alt="Cachorro oculos" class="collection\_\_img" />

</figure>

<div class="collection\_\_about">

<a href="article.html" class="collection\_\_title">Cuidados com os animais no verão</a>

<p class="collection\_\_date">12/07/2020</p>

<span class="collection\_\_tag">Cães</span>

</div>

</div>

<div class="collection\_\_post half">

<figure class="collection\_\_figure">

<img src="assets/img/kid.jpg" alt="Cachorro e criança" class="collection\_\_img" />

</figure>

<div class="collection\_\_about">

<a href="article.html" class="collection\_\_title half">5 raças de cachorro para quem tem crianças</a>

<p class="collection\_\_date">02/07/2020</p>

<span class="collection\_\_tag">Cães</span>

</div>

</div>

<div class="collection\_\_post half">

<figure class="collection\_\_figure">

<img src="assets/img/adopt.jpg" alt="Cachorro adotado" class="collection\_\_img" />

</figure>

<div class="collection\_\_about">

<a href="article.html" class="collection\_\_title half">Lugares para adotar cachorros em São Paulo</a>

<p class="collection\_\_date">16/06/2020</p>

<span class="collection\_\_tag">Cães</span>

</div>

</div>

</div>

<footer class="footer">

<img src="assets/img/footer-logo.png" alt="Footer logo" class="footer\_\_logo" />

<p class="footer\_\_txt">© 2020 Gatito Blog . All rights reserved.</p>

</footer>

</body>

</html>COPIAR CÓDIGO

E também o SCSS base para a estilização, antes de adicionarmos a condicional:

$half-width: 50%;

$full-width: 100%;

.collection {

display: flex;

flex-wrap: wrap;

justify-content: center;

align-items: center;

width: 80%;

margin-left: auto;

margin-right: auto;

font-family: Arial;

@include for-phone-only {

width: 90%;

}

&\_\_post {

display: flex;

flex-wrap: wrap;

justify-content: center;

align-items: center;

}

&\_\_figure {

width: 50%;

}

&\_\_img {

width: 100%;

}

&\_\_about {

display: flex;

flex-wrap: wrap;

justify-content: center;

align-items: center;

width: 50%;

text-align: center;

}

&\_\_title {

color: $dark-grey;

text-decoration: unset;

letter-spacing: 1px;

width: 100%;

margin: 10px 0;

&:hover {

color: $purple;

text-decoration: underline;

}

}

&\_\_date {

@extend %no-margin-top, %no-margin-bottom, %default-margin-left, %default-margin-right;

@include fontSize(12px);

font-style: italic;

width: 20%;

@include for-phone-only {

@include fontSize(10px);

}

@include for-tablet-only {

width: 30%;

}

}

&\_\_tag {

@include fontSize(14px);

width: 50px;

padding: 5px 7px;

color: $purple;

border: 1px solid $purple;

@extend %no-margin-top, %no-margin-bottom, %default-margin-left, %default-margin-right;

@include for-phone-only {

width: 30px;

@include fontSize(12px);

}

@include for-tablet-only {

width: 35px;

@include fontSize(12px);

}

}

&\_\_breadcrumb {

background-color: $light-grey;

text-align: center;

padding: 30px;

width: 100%;

display: flex;

justify-content: center;

align-items: center;

}

.breadcrumb {

&\_\_link {

@include fontSize(18px);

text-decoration: unset;

color: $dark-grey;

letter-spacing: 1px;

&:hover {

color: $purple;

text-decoration: underline;

}

}

&\_\_img {

width: 14px;

margin: 10px;

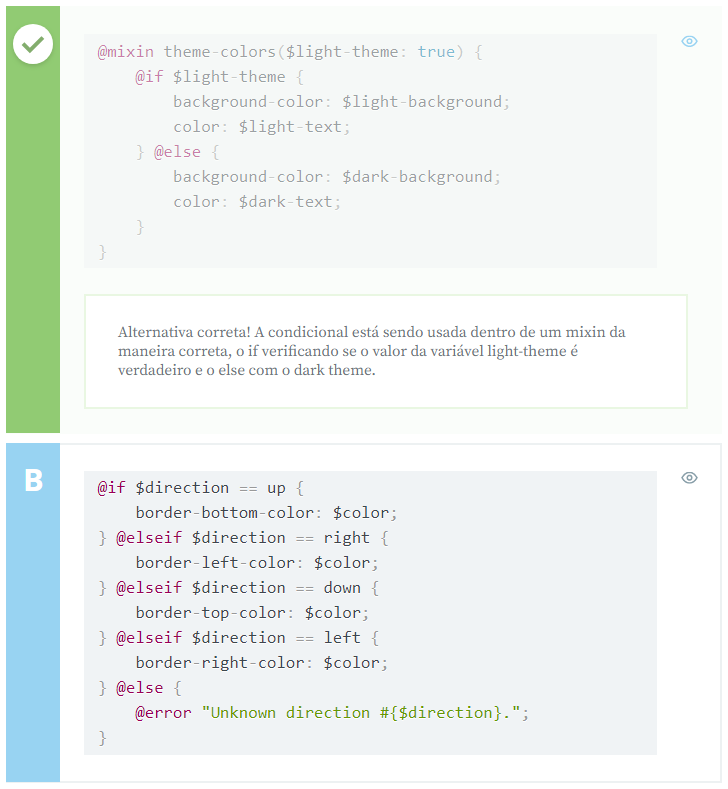
}

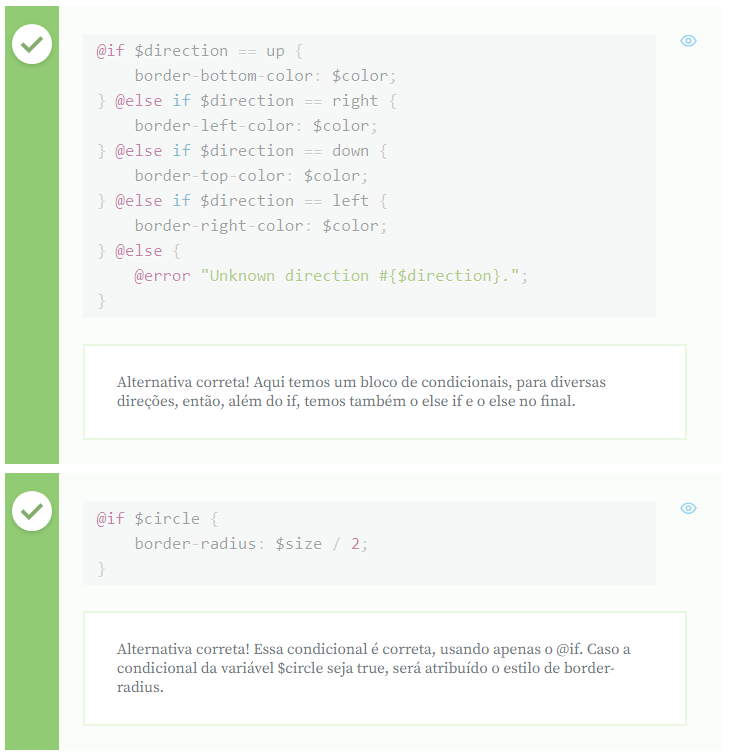
}

}COPIAR CÓDIGO

# 06Condicionais

Quais das alternativas abaixo são formas corretas de se fazer condicionais no Sass?





# 07While

SEM TRANSCRIÇÃO

# 08Faça como eu fiz: Página “Contato”

Vamos utilizar o while e criar uma página de contato no nosso site. Vamos criar um novo arquivo chamado **contact.html** e colar a base da estrutura que estamos usando para todas as páginas, com header e footer. Antes de iniciar o desenvolvimento do conteúdo da página precisamos adicionar um novo link no nosso Menu para a nova página, aqui no header da página de contato vamos dar um enter aqui na lista de links e adicionar uma nova li, com o nome Contato e linkando para o nova página, vamos copiar esse item da lista de links e adicioná-lo em cada uma das nossas páginas, que são a **Home**, o **article**, a **collection** e a **Sobre nós**.

Depois de salvar e atualizar a página do navegador, podemos notar que temos um novo link no nosso menu, e ao clicar nele somos redirecionados para nova página de contato. Ainda não temos nenhum conteúdo dentro da página e é isso que vamos adicionar agora, teremos uma imagem e ao lado as informações de contato. Logo abaixo do header vamos criar uma div chamada Contact e dentro dela teremos uma imagem, com a tag figure e a tag img e também uma div chamada info para as informações de contato que vamos passar para o nosso cliente, primeiro teremos um texto de Entre em contato, depois um subtítulo (Será um prazer tirar suas dúvidas e atender seu pet), abaixo o telefone e por último o horário de funcionamento.

Vamos salvar, e agora estilizar nossa página, para a disposição da imagem e do texto vamos chamar nosso mixin do display flex, com justify content center e também as propriedades de align items center, flex wrap, margin left e right auto, margin top com o nosso placeholder selector e width de 90%. Para o figure teremos um width de 50% e a imagem dentro dele 100%. As informações terão width de 40%, text align center e fonte Arial. Agora vamos voltar para o nosso arquivo html e adicionar um número para identificar cada uma das tags <p> que criamos, ficando “contact\_\_p-1”, “contact\_\_p-2”, “contact\_\_p-3” e “contact\_\_p-4” e nós vamos fazer isso para usar a função do while, como em outras linguagens de programação o while permite fazer um loop um determinado número de vezes, aqui vamos usar o loop para ir diminuindo o tamanho da fonte, teremos uma fonte maior para o título, menor para o subtítulo e assim vai ate o ultimo texto que terá a menor fonte de todas.

Voltando para o nosso arquivo **scss**, logo abaixo da classe info vamos criar uma variavel $x=1 que determinará o inicio do nosso loop e abaixo vamos colocar o @while $x < 5 { }, sendo $x o início do loop e **5 o número máximo do loop**. Dentro dele chamaremos a classe com o índice que criamos de forma variável &\_\_p-#{$x} e dentro da classe vamos incluir nossa função de font-size e o valor será 40px/$x, e fora das chaves vamos alterar o valor da variável $x para $x: $x + 1; dessa forma, a fonte inicial será 40px e a cada vez que ele passar no loop o valor de x aumentará mais um, e o 40 será dividido pelo novo valor. Vamos salvar e ir para o navegador, conforme esperado, o título está com a maior fonte e o tamanho vai diminuindo, podemos inspecionar para ver os tamanhos que resultaram da nossa divisão. Agora para finalizar vamos fazer a mesma coisa, porém com o padding-top, para dar espaçamento entre os textos, vamos adicionar a propriedade do padding e dessa vez iremos multiplicar o valor iniciar pelo $x, para ir aumentando a distância entre os textos, ficando padding-top: 5px\*$x. Pronto, nosso loop funcionou e agora temos uma nova tela com as informações de contato.

# 09O que aprendemos?

Nesta aula, aprendemos:

* Como fazer iterações no Sass.
  + Como em outras linguagens de programação, é possível fazer iterações no Sass com for, each ou while usando os At Rules.
* Como criar condicionais no Sass com valores (ex: boolean).
  + Podemos fazer condicionais no Sass com @if, @else if e @else, para atribuir determinado trecho de estilo dependendo de uma condição específica, por exemplo setar um tema dark ou light no nosso site.
* Como usar o While no Sass.
  + O while também funciona com at rule, ficando @while <expression> { ... }. Ele valida se a expressão é verdadeira, e enquanto ela for válida, ele irá executar o trecho de código contido entre as chaves.

# 10Conclusão

SEM TRANSCRIÇÃO