Parte superior do formulário

Parte inferior do formulário

**Front-end UI**

**Efeito paralaxe e carrossel**

**Faça esse curso de HTML e CSS e:**

* Aprenda a recolher informações importantes do layout no Adobe XD
* Entenda propriedades de tipografia como letter-spacing e line-height
* Veja efeitos de paralaxe: fixo e verdadeiro
* Conheça o funcionamento da perspectiva do efeito paralaxe verdadeiro
* Construa um carrossel completo utilizando apenas HTML, CSS e JavaScript

**Aulas**

[Layout e ambiente de desenvolvimento](https://cursos.alura.com.br/course/front-end-ui-paralaxe-carrossel/section/11174/tasks)

[0 / 1133min](https://cursos.alura.com.br/course/front-end-ui-paralaxe-carrossel/section/11174/tasks)

* + Apresentação
  + Preparando o ambiente
  + A base do projeto
  + Para saber mais: Variáveis CSS
  + Para saber mais: EMMET
  + Browser-sync
  + Browser-sync
  + Estrutura do cabeçalho
  + Estilização do cabeçalho
  + Faça como eu fiz
  + O que aprendemos?

[Chamada e Parallax](https://cursos.alura.com.br/course/front-end-ui-paralaxe-carrossel/section/11175/tasks)

[0 / 724min](https://cursos.alura.com.br/course/front-end-ui-paralaxe-carrossel/section/11175/tasks)

* + Projeto da aula anterior
  + Definindo espaços
  + Tamanhos de viewport
  + Imagem do produto
  + A descrição do produto
  + Faça como eu fiz
  + O que aprendemos?

[Começando o carrossel](https://cursos.alura.com.br/course/front-end-ui-paralaxe-carrossel/section/11176/tasks)

[0 / 927min](https://cursos.alura.com.br/course/front-end-ui-paralaxe-carrossel/section/11176/tasks)

* + Projeto da aula anterior
  + Seção do carrossel
  + Estrutura do carrossel
  + Estilização do carrossel
  + Iniciando o JavaScript
  + Para saber mais: Classes no JS
  + Exportando e importando uma classe JS no HTML
  + Faça como eu fiz
  + O que aprendemos?

[Melhorando o carrossel](https://cursos.alura.com.br/course/front-end-ui-paralaxe-carrossel/section/11177/tasks)

[0 / 937min](https://cursos.alura.com.br/course/front-end-ui-paralaxe-carrossel/section/11177/tasks)

* + Projeto da aula anterior
  + A base do carrossel
  + Preparando o filme
  + Movimentação
  + Para saber mais: bind(this)
  + Interações
  + data-attributes
  + Faça como eu fiz
  + O que aprendemos?

[Perspectiva e toques finais](https://cursos.alura.com.br/course/front-end-ui-paralaxe-carrossel/section/11178/tasks)

[0 / 716min](https://cursos.alura.com.br/course/front-end-ui-paralaxe-carrossel/section/11178/tasks)

* + Projeto da aula anterior
  + True parallax
  + Perspectiva
  + Montando o rodapé
  + Faça como eu fiz
  + O que aprendemos?
  + Conclusão

# 01Apresentação

**SEM TRANSCRIÇÃO**

# 02Preparando o ambiente

Para este curso, você precisará baixar alguns arquivos. O zip abaixo contém as imagens de produto e do carrossel e também o arquivo "reset.css".

O link para o download do zip [arquivos\_gatitostore.zip](https://drive.google.com/file/d/1m9yyoLGsA4uRf6vUT2f4miWQNg-FF0w5/view?usp=sharing).

O link para a visualização do [layout](https://xd.adobe.com/view/d925039a-fb08-472f-b1a0-b739d215d82e-8c76/) no Adobe XD.

# 03A base do projeto

**SEM TRANSCRIÇÃO**

# 04Para saber mais: Variáveis CSS

Variáveis CSS são uma das ferramentas do CSS que permitem algumas facilidades, como manutenção do código e organização.

Primeiro é necessário declarar uma variável com a seguinte estrutura dentro de um seletor:

.seletor {

--nome-da-variavel: valor;

}COPIAR CÓDIGO

Quando declaramos variáveis que serão usadas em toda a página, é interessante usar o seletor html porque é o elemento raiz do documento HTML. Só que chamar o seletor html pode gerar alguns problemas de sobrescrita porque o peso de um seletor de tag é 1 ponto, ou seja, qualquer outro seletor de tag consegue facilmente sobrescrever os valores das variáveis. Para evitar esse problema, usamos o seletor :root que é a mesma coisa que o seletor html, só que o seu peso é 2, dificultando a sobrescrita por seletores comuns de tags.

Exemplo de declaração de variável:

:root {

--branco: #FFFFFF;

}COPIAR CÓDIGO

Para chamar a variável criada basta usar a seguinte sintaxe:

.cor-branca {

color: var(--branco);

}COPIAR CÓDIGO

# 05Para saber mais: EMMET

Para saber mais sobre o funcionamento do EMMET e sua instalação em outros editores de texto, temos um Alura + explicando seus conceitos:

[Alura + EMMET](https://cursos.alura.com.br/emmet-instalacao-e-comandos-no-atom-c311)

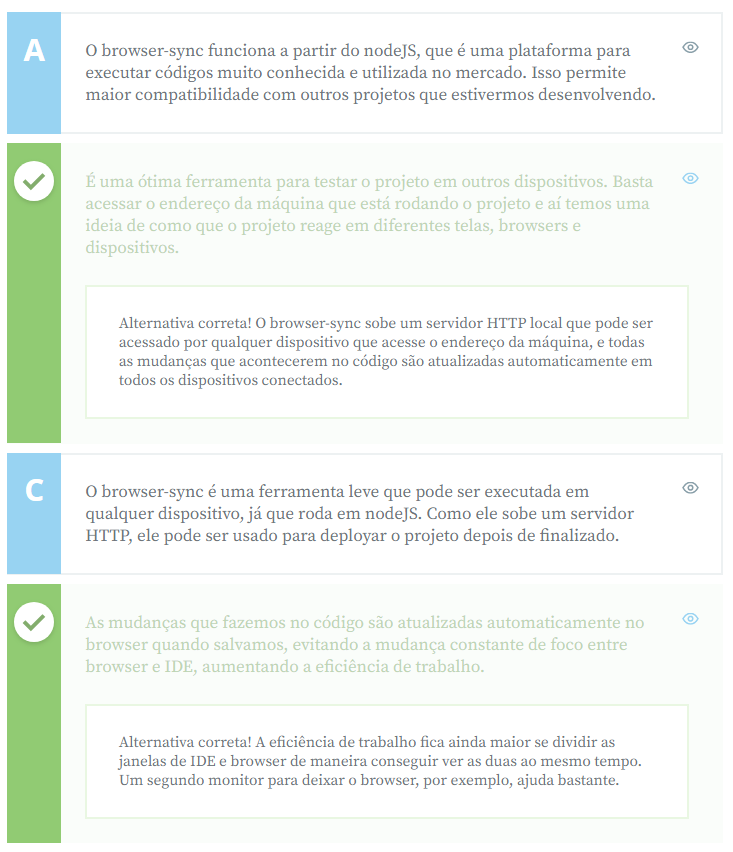
# 06Browser-sync

**SEM TRANSCRIÇÃO**

# 07Browser-sync

O browser-sync é uma ferramenta que ajuda bastante no desenvolvimento de um projeto, principalmente quando estamos trabalhando com o front-end na parte estrutural (HTML) e visual (CSS).

Quais alternativas apresentam vantagens de se usar browser-sync?



# 08Estrutura do cabeçalho

**SEM TRANSCRIÇÃO**

# 09Estilização do cabeçalho

**SEM TRANSCRIÇÃO**

# 10Faça como eu fiz

Praticar ajuda bastante no aprendizado de um novo conceito. Assim, é muito importante que você implemente o que foi apresentado nesta aula.

Não deixe de sanar suas dúvidas antes de dar continuidade ao curso. Estaremos te esperando no fórum da Alura caso alguma dúvida surja.

# 11O que aprendemos?

Nesta aula, aprendemos a:

* Recolher informações importantes do Adobe XD
* Preparação e estruturação de arquivos
* Melhorar o ambiente de desenvolvimento com Browser-sync

# 01Projeto da aula anterior

Caso queira começar daqui, você pode baixar o projeto da aula anterior nesse [link](https://github.com/alura-cursos/gatitoshop/archive/Aula1.zip).

# 02Definindo espaços

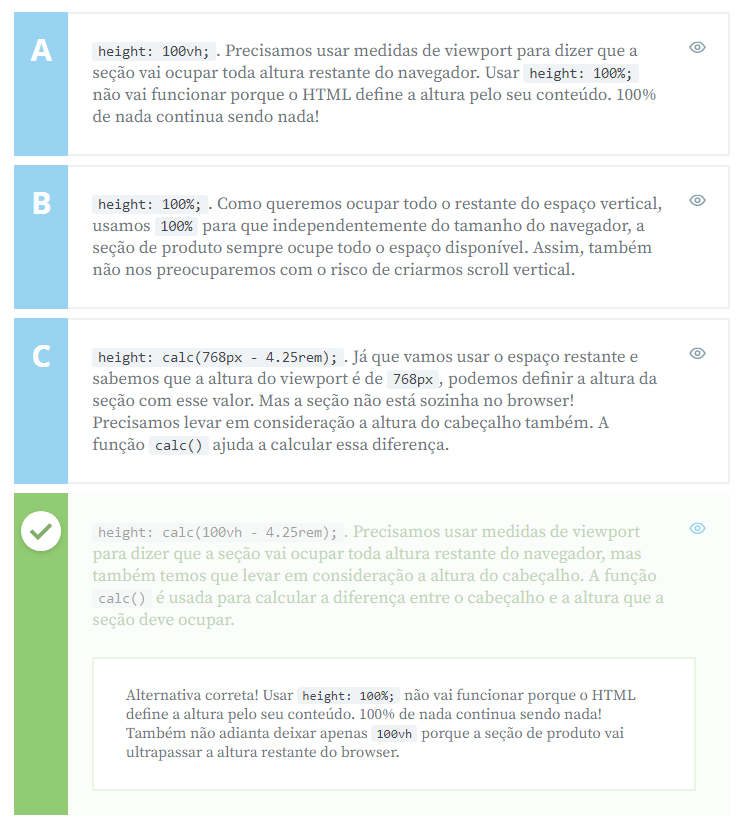
**SEM TRANSCRIÇÃO**

# 03Tamanhos de viewport

Durante o desenvolvimento do projeto nos encontramos em uma situação que a seção de produto precisa ocupar o restante do espaço disponível do navegador. A largura já é de 100% por conta do tipo de display que a tag <section> possui por padrão, mas agora a altura precisa ser definida manualmente.

O projeto foi desenhado tendo em mente um browser com viewport de 1366px de largura por 768px de altura.

Escolha a alternativa que pode ser usada para definir a altura da seção e que tenha a justificativa correta.



# 04Imagem do produto

**SEM TRANSCRIÇÃO**

# 05A descrição do produto

**SEM TRANSCRIÇÃO**

# 06Faça como eu fiz

Praticar ajuda bastante no aprendizado de um novo conceito. Assim, é muito importante que você implemente o que foi apresentado nesta aula.

Não deixe de sanar suas dúvidas antes de dar continuidade ao curso. Estaremos te esperando no fórum da Alura caso alguma dúvida surja.

# 07O que aprendemos?

Nesta aula, aprendemos:

* A importância do viewport e suas medidas
* O funcionamento de um parallax fixo

# 01Projeto da aula anterior

Caso queira começar daqui, você pode baixar o projeto da aula anterior nesse [link](https://github.com/alura-cursos/gatitoshop/archive/Aula2.zip).

# 02Seção do carrossel

**SEM TRANSCRIÇÃO**

# 03Estrutura do carrossel

**SEM TRANSCRIÇÃO**

# 04Estilização do carrossel

**SEM TRANSCRIÇÃO**

# 05Iniciando o JavaScript

**SEM TRANSCRIÇÃO**

# 06Para saber mais: Classes no JS

Classes são usadas para transformar referências do mundo real em código, até mesmo para coisas abstratas.

Em nosso projeto fizemos um desenho com uma analogia sobre o que seria o carrossel. Cada um dos "materiais" necessários para montar o carrossel (tela, rolo de filme, botões e indicadores) são representados por atributos dentro de uma classe, ou seja, atributos são características que o carrossel possui.

Por exemplo, podemos indicar o nome do carrossel:

class Carrossel {

nome: ‘Carrossel da Alura’

}COPIAR CÓDIGO

O carrossel possui também um comportamento esperado, que é rodar. Então, além dos atributos que representam o carrossel, uma classe também possui um modo de representar um comportamento ou uma ação, e esse modo é conhecido como método ou função.

Esse comportamento pode ser rodar ao clique de um botão, que podemos representar assim:

//Arquivo Carrossel.js

class Carrossel {

nome = "Carrossel da Alura"

girar() {

//Dentro da função tem o código que faz o carrossel girar

}

}COPIAR CÓDIGO

Links de referência:

* <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script>
* <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import>
* <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules>
* <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes>

# 07Exportando e importando uma classe JS no HTML

No projeto vamos usar classes JS para montar o carrossel. Para que a classe possa ser usada no HTML, alguns passos precisam ser executados. Um deles é ter um serviço que coloque o projeto em um servidor HTTP, que foi o que fizemos com o browser-sync.

Selecione a alternativa que contém todos os passos restantes para utilizar a classe no HTML.









# 08Faça como eu fiz

Praticar ajuda bastante no aprendizado de um novo conceito. Assim, é muito importante que você implemente o que foi apresentado nesta aula.

Não deixe de sanar suas dúvidas antes de dar continuidade ao curso. Estaremos te esperando no fórum da Alura caso alguma dúvida surja.

# 09O que aprendemos?

Nesta aula, aprendemos:

* O processo de abstração do carrossel
* As propriedades CSS overflow e object-fit
* A criação e chamada de classes JS

# 01Projeto da aula anterior

Caso queira começar daqui, você pode baixar o projeto da aula anterior nesse [link](https://github.com/alura-cursos/gatitoshop/archive/Aula3.zip).

# 02A base do carrossel

**SEM TRANSCRIÇÃO**

# 03Preparando o filme

**SEM TRANSCRIÇÃO**

# 04Movimentação

**SEM TRANSCRIÇÃO**

# 05Para saber mais: bind(this)

Durante quase todo o processo de construção da lógica do carrossel usamos this.{alguma coisa} para criar atributos e fazer chamada de funções. No momento em que vamos colocar uma função para um evento, nós acabamos usando a função .bind(this). Mas por que?

this.proximo.addEventListener('click', this.proximoSlide.bind(this))COPIAR CÓDIGO

Primeiro precisamos entender quem é this no momento em que colocamos a função no eventListener. Montando uma função rápida para ver o vem de retorno:

// Arquivo carousel.js

...

this.proximo.addEventListener('click', this.getThis)

getThis() {

console.log(this) //O botão vai aparecer no console

}

...COPIAR CÓDIGO



Então this sem a função bind() terá o botão como referência, e o botão não terá acesso a todas as funções e atributos da classe que criamos. Vamos ver o que acontece com this depois da função bind():

// Arquivo carousel.js

...

this.proximo.addEventListener('click', this.getThis.bind(this))

getThis() {

console.log(this) //A classe Carousel vai aparecer no console

}

...COPIAR CÓDIGO



Agora this virou o objeto de Carousel, e esse objeto tem acesso a todos os atributos e funções necessários.

# 06Interações

**SEM TRANSCRIÇÃO**

# 07data-attributes

Para pegar elementos do DOM dentro do JavaScript usamos a função do document que é .querySelector('{algum seletor}'). Além disso, existe o costume de se usar seletores de **classes** para selecionar esses elementos. Mas no projeto acabamos usando outro, que é o seletor de atributo: [data-alguma-coisa].

No exemplo do que fizemos no projeto:

<button class="carousel\_\_indicador carousel\_\_indicador--ativo" data-indicador="0" aria-label="Slide 1"></button>COPIAR CÓDIGO

Esse atributo data-indicador="0" é o que nós chamamos de **data-attribute** e ele pode nos trazer algumas vantagens quando trabalhamos com JavaScript, como por exemplo, passar valores.

Selecione a alternativa correta sobre o uso de **data-attributes**:



# 08Faça como eu fiz

Praticar ajuda bastante no aprendizado de um novo conceito. Assim, é muito importante que você implemente o que foi apresentado nesta aula.

Não deixe de sanar suas dúvidas antes de dar continuidade ao curso. Estaremos te esperando no fórum da Alura caso alguma dúvida surja.

# 09O que aprendemos?

Nesta aula, aprendemos a:

* Criação de atributos na classe
* Passar valores para a classe
* Criação e chamada de funções dentro da classe
* Colocar funções da classe em eventos de elementos do DOM

# 01Projeto da aula anterior

Caso queira começar daqui, você pode baixar o projeto da aula anterior nesse [link](https://github.com/alura-cursos/gatitoshop/archive/Aula4.zip).

# 02True parallax

**SEM TRANSCRIÇÃO**

# 03Perspectiva

O efeito de parallax acontece quando existe diferença de velocidade de movimento entre um elemento e outro. No caso do **parallax fixo**, a imagem de produto ficou fixa enquanto todo o restante da página se movimentou. Mas para chegar no **true parallax** a imagem precisa se mover junto com o restante da página, só que com uma velocidade diferente.

Escolha a alternativa correta sobre o efeito de **true parallax**:



# 04Montando o rodapé

**SEM TRANSCRIÇÃO**

# 05Faça como eu fiz

Praticar ajuda bastante no aprendizado de um novo conceito. Assim, é muito importante que você implemente o que foi apresentado nesta aula.

Não deixe de sanar suas dúvidas antes de dar continuidade ao curso. Estaremos te esperando no fórum da Alura caso alguma dúvida surja.

# 06O que aprendemos?

Nesta aula, aprendemos:

* O efeito paralaxe verdadeiro
* As propriedades perspective e overflow
* Outros valores da propriedade transform:

# 07Conclusão

**SEM TRANSCRIÇÃO**