INTENSIVÃO DE JAVASCRIPT

Apostila Completa
Aula 3

Guia passo a passo para construir seu próprio site de E-Commerce a partir do zero!

Parte 1

Estilização - Aula Extra

Estilização - Aula Extra

Se por acaso você ainda não teve a oportunidade de assistir à **aula extra sobre o refinamento do estilo CSS dos elementos** que construímos até este ponto do nosso projeto, não se preocupe.

Ao lado está o código que foi modificado. Você pode dar uma olhada nas mudanças feitas e implementá-las no seu projeto quando achar conveniente. As modificações foram feitas no **arquivo index.html**.

index.html > html

```
<body class="bg-stone-200 flex flex-col">
  <section
    id="carrinho"
    class="fixed top-0 right-[-360px] bg-slate-950 h-screen w-[360px] text-slate-200 z-50"
  >
    <div id="cabecalho-carrinho" class="flex justify-between m-4">
      <p>Meu carrinho</p>
      <button id="fechar-carrinho">
        <i class="fa-solid fa-circle-xmark"></i>
      </button>
    </div>
    <section id="produtos-carrinho">
      <p>Camisa Larga com Bolsos</p>
      <p>Casaco Reto com Lã</p>
      <p>Jaqueta com Efeito Camurça</p>
    </section>
    <p id="preco-total">$200</p>
    <button class="bg-slate-200 text-slate-900 p-1">Finalizar Compra</button>
  </section>
  <header class="flex text-xl bg-slate-950 px-8 py-4 justify-between sticky top-0 shadow-xl shadow-slate-400 z-10">
    <img
      class="h-16"
      src="./assets/logo/hashtag.svg"
      alt="Logo do Magazine Hashtag."
    />
    <div class="text-slate-200 flex gap-2 items-end mx-2">
      <button id="abrir-carrinho" class="mx-2"><i class="fa-solid fa-cart-shopping"></i></button>
      <button class="mr-2"><i class="fa-solid fa-user"></i></button>
    </div>
  </header>
  <main>
    <section id="container-produto" class="flex flex-wrap p-10 justify-center gap-10"></section>
  </main>
  <script type="module" src="main.js"></script>
</body>
```

Parte 2

Estilização - Carrinho de Compras

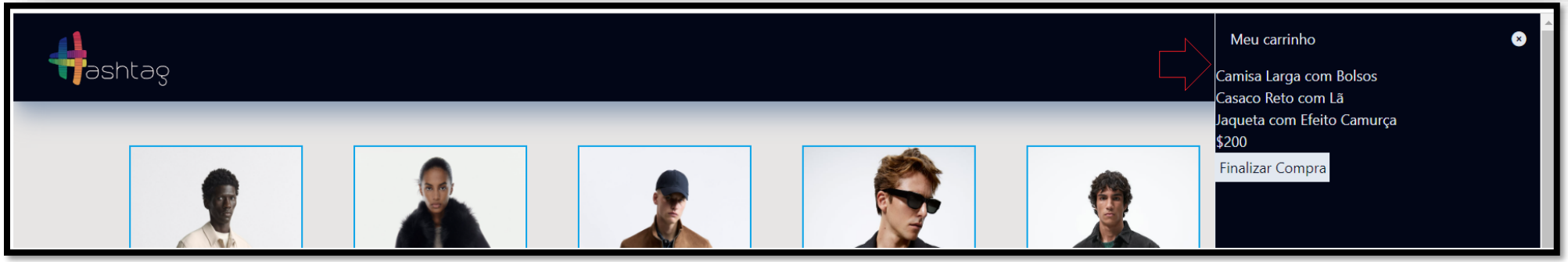
Estilização - Carrinho de Compras

E antes de continuarmos com a inteligência do nosso projeto, vamos adicionar mais alguns estilos ao nosso carrinho de compras.

Então dentro do **arquivo index.html**, no elemento **<section> de "id='carrinho'"** que encapsula toda a estrutura do nosso carrinho, vamos adicionar as classes a seguir:

A **classe "duration-200"** que se refere a um valor de duração de animação pré-definido, e a **classe "border-l border-slate-200"** que se refere a estilos de borda para um elemento. Com essas classes implementadas o nosso carrinho de compras terá uma animação curta de 200 milissegundos e uma borda com largura e cor específicas conforme estilo do Tailwind CSS.

```
17  <section
18    id="carrinho"
19    class="fixed top-0 right-[-360px] bg-slate-950 h-screen w-[360px] text-slate-200 z-50 duration-200 border-l border-slate-200"
20  >
```



Estilização - Carrinho de Compras

Vamos adicionar um padding dentro desse mesmo elemento **<section>** utilizando a **classe 'p-5'**.

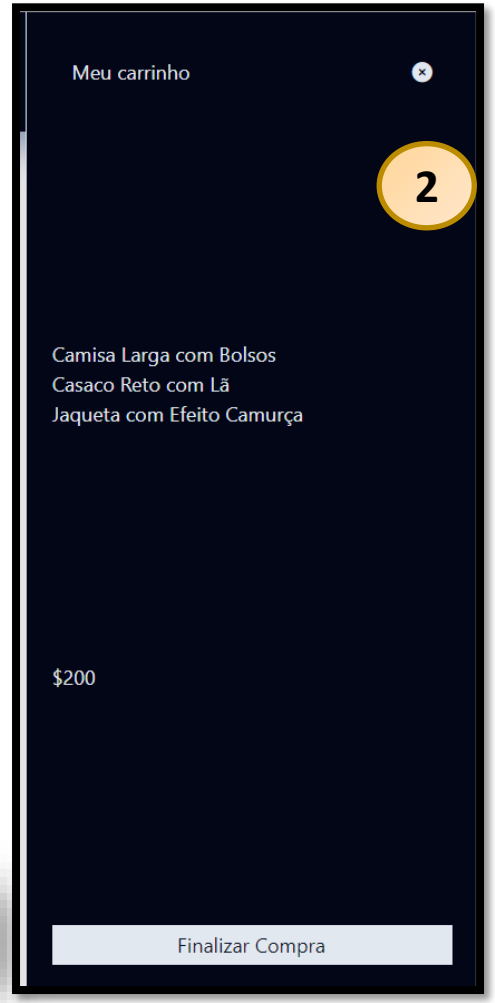
Transformaremos ele em um container flexível com a **classe flex** e adicionaremos a **classe flex-col** para que os elementos filhos dessa <section> se posicionem em coluna vertical.

Com a **classe justify-between** criaremos um espaçamento uniforme para alinhar os elementos **(Imagem 1)**.

Finalizando essa etapa, o protótipo do nosso carrinho de compras deve estar como a imagem ao lado **(Imagem 2)**:

1

```
<section
  id="carrinho"
  class="fixed top-0 right-[-360px] bg-slate-950 h-screen w-[360px] text-slate-200 z-50 duration-200 border-l border-slate-200 p-5 flex flex-col justify-between"
>
```



Estilização - Carrinho de Compras

Agora vamos trabalhar na estrutura do nosso cartão de produtos, o componente que será exibido dentro do nosso carrinho de compras.

Inicialmente vamos substituir o elemento ** Camisa Larga com Bolsos ** pela **tag <article>** que é um elemento HTML que tem um propósito específico: ela é usada para marcar um conteúdo independente e auto-suficiente em uma página web. Geralmente, é utilizada para agrupar informações que podem ser lidas, compartilhadas e entendidas por conta própria, como notícias, posts de blog, e no nosso caso, um **"cartão de produtos"**.

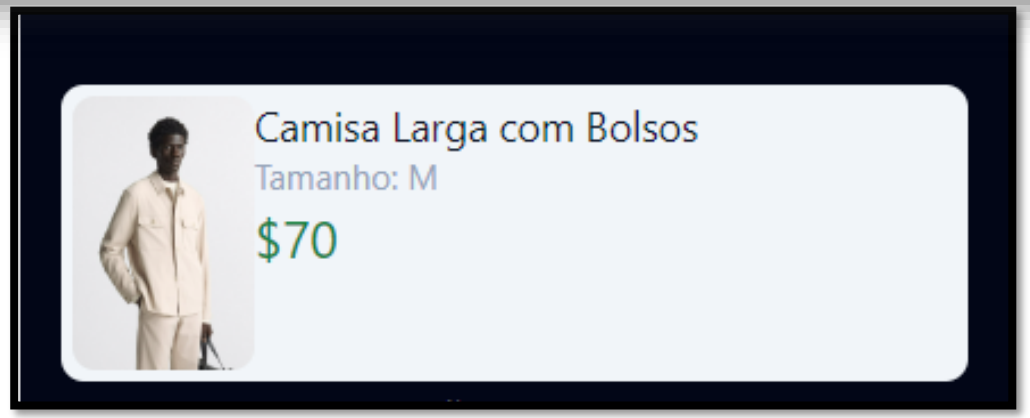
Por hora, vamos deixar um exemplo estático para conseguir visualizar as mudanças que estamos realizando nesse elemento e depois vamos aplicar o dinamismo com o Javascript.

```
27      <section id="produtos-carrinho">
28        <article>
29          <img src="./assets/img/product-1.jpg" alt="Camisa Larga com Bolsos">
30        </article>
```

Estilização - Carrinho de Compras

Abaixo temos o modelo de cartão de produtos do nosso carinho com seus elementos de imagem, nome do produto, preço e também a estilização inicial deles:

```
27    <section id="produtos-carrinho">
28      <article class="flex bg-slate-100 rounded-lg p-1">
29        <img
30          src="./assets/img/product-1.jpg"
31          alt="Carrinho: Camisa Larga com Bolsos"
32          class="h-24 rounded-lg"
33        />
34        <div>
35          <p class="text-slate-900 text-sm">Camisa Larga com Bolsos</p>
36          <p class="text-slate-400 text-xs">Tamanho: M</p>
37          <p class="text-green-700 text-lg">$70</p>
38        </div>
39      </article>
```



A **classe "flex**" é aplicada para permitir um arranjo flexível dos elementos internos, facilitando o alinhamento.

- "bg-slate-100" dá ao componente um fundo suave em uma tonalidade slate, contribuindo para a estética.

A **classe "rounded-lg"** confere bordas arredondadas com uma sutileza agradável, suavizando a aparência.
- "p-1" acrescenta um espaçamento interno mínimo, garantindo margens internas equilibradas.

A imagem possui uma altura fixa de 24 pixels (**h-24**), mantendo proporções adequadas.

O nome do produto é estilizado com **"text-slate-900"** e **"text-sm"**. Informações sobre o tamanho da camisa são exibidas em **"text-slate-400"** com **"text-xs"**.
O preço é enfatizado em verde através de **"text-green-700"** e **"text-lg"**, atraindo a atenção.

Estilização - Carrinho de Compras

Adicionaremos um botão, responsável por excluir o cartão de produtos, no nosso carrinho e para posicionarmos ele usaremos as **classes relative e absolute**, que são frequentemente usadas em conjunto para gerenciar o posicionamento. Além de estilizarmos o botão com as propriedades de padding e cor. Dessa forma, nosso carrinho de compras terá esse formato:

```
27    <section id="produtos-carrinho">
28      <article class="flex bg-slate-100 rounded-lg p-1 relative">
29        <button id="fechar-carrinho" class="absolute top-0 right-2">
30          <i class="fa-solid fa-circle-xmark text-slate-500 hover:text-slate-800"></i>
31        </button>
32        <img
33          src="./assets/img/product-1.jpg"
34          alt="Carrinho: Camisa Larga com Bolsos"
35          class="h-24 rounded-lg"
36        />
37        <div class="py-2">
38          <p class="text-slate-900 text-sm">Camisa Larga com Bolsos</p>
39          <p class="text-slate-400 text-xs">Tamanho: M</p>
40          <p class="text-green-700 text-lg">$70</p>
41        </div>
42      </article>
```

Parte 3

Adicionar ao Carrinho com Javascript

Adicionar ao Carrinho com Javascript

Vamos desenvolver a funcionalidade **"Adicionar ao Carrinho"** usando programação em JavaScript. À medida que acompanhamos cada etapa dessa implementação, seu botão será capaz de adicionar itens ao carrinho de compras do site.

No arquivo **menuCarrinho.js**, vamos adicionar a etapa lógica para incluir um produto no carrinho. Para isso, criaremos uma **função** chamada **"adicionarAoCarrinho"**. Também removeremos do arquivo **index.html** o **elemento <article>** por completo, que representa o cartão de produtos do carrinho, incluindo todos os elementos internos. Em seguida, vamos armazenar esse conjunto de elementos em uma variável chamada **"cartaoProdutoCarrinho"**. Faremos isso usando crases (``) para aplicar o **String Template**, permitindo adaptações dinâmicas no conteúdo conforme necessário.

menuCarrinho.js

src > menuCarrinho.js > ...

```
19  function adicionarAoCarrinho() {
20      const cartaoProdutoCarrinho = `<article class="flex bg-slate-100 rounded-lg p-1 relative">
21      <button id="fechar-carrinho" class="absolute top-0 right-2">
22        <i class="fa-solid fa-circle-xmark text-slate-500 hover:text-slate-800"></i>
23      </button>
24      <img
25        src="./assets/img/product-1.jpg"
26        alt="Carrinho: Camisa Larga com Bolsos"
27        class="h-24 rounded-lg"
28      />
29      <div class="py-2">
30        <p class="text-slate-900 text-sm">Camisa Larga com Bolsos</p>
31        <p class="text-slate-400 text-xs">Tamanho: M</p>
32        <p class="text-green-700 text-lg">$70</p>
33      </div>
34    </article>`;
35  }
```

Adicionar ao Carrinho com Javascript

Criaremos uma segunda variável chamada **"containerProdutosCarrinho"**. Essa variável está vinculada ao elemento HTML **<section>** com o **ID "produtos-carrinho"**. Este elemento é o container que representa o carrinho de compras, onde os produtos serão exibidos ou adicionados.

```
20    const containerProdutosCarrinho = document.getElementById("produtos-carrinho");
```

Vamos implementar ao código **"containerProdutosCarrinho.innerHTML += cartaoProdutoCarrinho;"** que acrescenta o conteúdo do **"cartaoProdutoCarrinho"** ao interior do elemento representado pela variável **"containerProdutosCarrinho"**. Isso significa que o HTML contido em **"cartaoProdutoCarrinho"** é adicionado ao final do conteúdo existente dentro desse container. É uma forma de inserir dinamicamente o cartão do produto no carrinho de compras na página.

```
36    containerProdutosCarrinho.innerHTML += cartaoProdutoCarrinho;
```

Adicionar ao Carrinho com Javascript

Vamos realizar algumas modificações no nosso projeto. A primeira etapa envolve criar um novo arquivo chamado **"cartaoProduto.js"** na pasta **"src"**. Em seguida, vamos retirar a estrutura completa do cartão de produtos do arquivo **"main.js"**. Essa estrutura está dentro do trecho de código **"for (const produtoCatalogo of catalogo)"**. A seguir, moveremos essa estrutura completa para o nosso recém-criado arquivo "cartaoProduto.js".

Essa estrutura será incorporada dentro de uma função denominada **"renderizarCatalogo()"**. Dessa forma, a exportaremos para possibilitar o uso em diferentes partes do nosso projeto.

cartaoProduto.js

```
src > cartaoProduto.js > ...
1   export function renderizarCatalago() {
2       for (const produtoCatalogo of catalogo) {
3           const cartaoProduto = `<div class="border-solid border-2 border-sky-500 w-48 m-2"
4           id="card-produto-${produtoCatalogo.id}">
5           <img
6           src="./assets/img/${produtoCatalogo.imagem}"
7           alt="Produto 1 do Magazine Hashtag."
8           style="height: 300px"
9           />
10          <p class="marca">${produtoCatalogo.marca}</p>
11          <p>${produtoCatalogo.nome}</p>
12          <p>$${produtoCatalogo.preco}</p>
13          <button>Adicionar</button>
14          </div>`;
15          document.getElementById("container-produto").innerHTML += cartaoProduto;
16      }
17  }
```

Adicionar ao Carrinho com Javascript

Neste ponto, é importante importar a **função "renderizarCatalogo()"** para o arquivo "main.js". Depois de importada, você pode chamá-la simplesmente digitando **"renderizarCatalogo();"** no arquivo. Isso fará com que a função seja executada **(Imagem 1)**.

main.js

```
1  import { renderizarCatalago } from "./src/cartaoProduto";
2  import { inicializarCarrinho } from "./src/menuCarrinho";
3
4  renderizarCatalago();
5  inicializarCarrinho();
```

1

Além disso, avançaremos com uma organização eficiente. Vamos transferir o nosso **objeto "catalogo"** para um novo arquivo chamado **"utilidades.js"** dentro da pasta **"src"**. Dentro desse arquivo, iremos colar o objeto "catalogo" e, em seguida, exportá-lo. Essa abordagem permitirá o compartilhamento do catálogo com outros componentes do projeto **(Imagem 2)**.

src > utilidades.js > catalogo

2

```
1  export const catalogo = [
2    {
3      id: 1,
4      marca: 'Zara',
5      nome: 'Camisa Larga com Bolsos',
6      preco: 70,
7      imagem: 'product-1.jpg',
8      feminino: false,
9    },
10   {
11     id: 2,
12     marca: 'Zara',
13     nome: 'Casaco Reto com Lã',
14     preco: 85,
15     imagem: 'product-2.jpg',
16     feminino: true,
17   },
18   {
19     id: 3,
20     marca: 'Zara',
21     nome: 'Jaqueta com Efeito Camurça',
22     preco: 60,
23     imagem: 'product-3.jpg',
24     feminino: false,
25   },
```

Adicionar ao Carrinho com Javascript

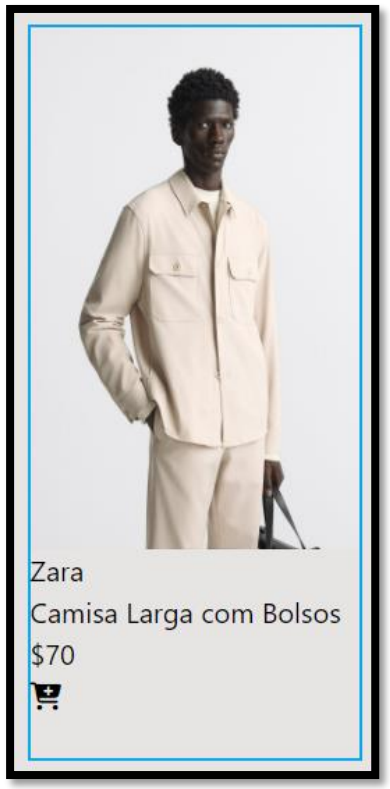
Agora, direcionando nosso foco para o **arquivo "cartaoProduto.js"**, importaremos o objeto "catalogo". Essa ação nos permitirá utilizar o catálogo de produtos de forma eficaz na construção dos cartões individuais de produto.

```
src > JS cartaoProduto.js > ...
  1   import { catalogo } from "./utilidades";
```

É relevante compreender que o **arquivo "utilidades.js"** tem um papel crucial. Ele age como um espaço central para armazenar informações que serão valiosas em diferentes partes do nosso projeto, proporcionando uma estrutura organizada para a gestão de dados compartilhados.

Adicionar ao Carrinho com Javascript

1-Para dar estilo ao nosso **botão "Adicionar"**, faremos uso de um ícone de "cart" (carrinho) disponível no site Font Awesome. Vamos aprimorar o botão ao substituir o texto "Adicionar" pelo código HTML do **ícone "cart-plus"**.



2- Nesse ponto, vamos adicionar algumas classes de estilos ao elemento que engloba todo o nosso cartão de produtos, a ** **.

```
<div class="border-solid border-2 border-sky-500 w-48 m-2 flex flex-col p-2"
```

3-Vamos deletar a **propriedade style** do elemento **<img>**, já que estamos utilizando as classes utilitárias do Tailwind.

```
10    excluir   X style="height: 300px"
```

4-A terá uma **classe justify-between**, para criar os espaçamentos entre os elementos do nosso cartão.

Adicionar ao Carrinho com Javascript

5-As seguintes classes vão ser aplicadas ao nosso botão:

```
<button class="bg-slate-950 hover:bg-slate-700 text-slate-200">
<i class="fa-solid fa-cart-plus"></i>
</button>
```

6-
Vamos adicionar mais uma classe na nossa ** .**
A **classe group** que permite criar interações específicas para os elementos filhos dentro do contexto do elemento pai.

```
<div class="border-solid border-2 border-sky-500
 w-48 m-2 flex flex-col p-2 justify-between group"
```

7- Vamos adicionar a **classe scale-110** à imagem, juntamente com a **classe duration-300**, para criar o efeito de aumento da imagem quando o usuário passar o cursor sobre ela. Isso ocorre devido à transformação de escala que amplia a imagem em 10% (fator 1.1) com uma transição suave de 300 milissegundos.

```
7        <img
8        src="./assets/img/${produtoCatalogo.imagem}"
9        alt="Produto 1 do Magazine Hashtag."
10       class="group-hover:scale-110 duration-300"
11       />
```

8-Aterar o tamanho dos textos dentro do nosso cartão de produtos:

```
12       <p class="text-sm">${produtoCatalogo.marca}</p>
13       <p class="text-sm">${produtoCatalogo.nome}</p>
14       <p class="text-sm">$${produtoCatalogo.preco}</p>
```

Adicionar ao Carrinho com Javascript

9- Removeremos as **classes border-2 e border-sky-500 da **, pois ela servia para limitar o nosso cartão de produtos enquanto construíamos a estrutura dele.

10- Para limitar o nosso cartão de produtos e ele ter uma aparência melhor, adicionaremos um sombreamento nele. Então no elemento ** ** adicionamos as **classes shadow-xl** e **shadow-slate-400**.

11- Adicionaremos a **classe rounded-lg** na ** **, para criarmos um layout levemente arredondado ao nosso cartão.

```
<div class="border-solid w-48 m-2 flex flex-col p-2
 justify-between shadow-xl shadow-slate-400  rounded-lg group"
```

12- No elemento **<img>** vamos criar uma margem com a **classe my-3** e também vamos dar o efeito de arredondamento com a **classe rounded-lg**.

```
<img
src="./assets/img/${produtoCatalogo.imagem}"
alt="Produto 1 do Magazine Hashtag."
class="group-hover:scale-110 duration-300 my-3 roudend-lg"
/>
```

- Lembrando que todas essas modificações foram feitas no **arquivo "cartaoProduto.js"** do nosso projeto.

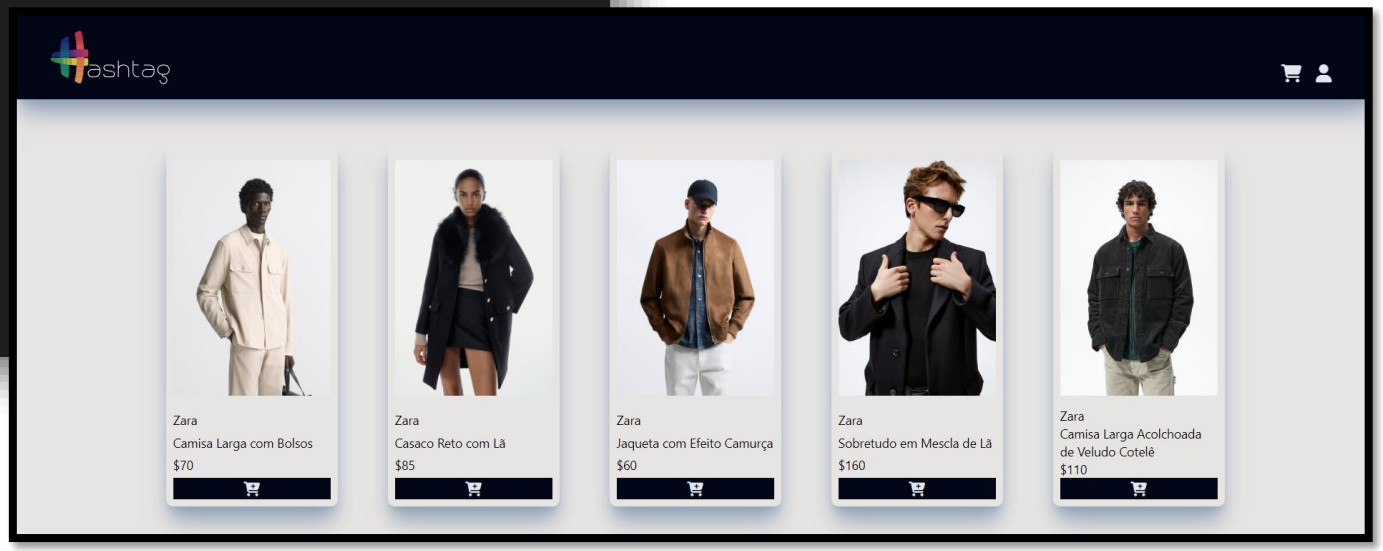
Adicionar ao Carrinho com Javascript

Após aplicarmos as mudanças nos estilos do nosso cartão de produtos, o resultado final do nosso código e do nosso site ficou assim:

cartaoProduto.js

```
import { catalogo } from "./utilidades";

export function renderizarCatalago() {
    for (const produtoCatalogo of catalogo) {
        const cartaoProduto = `<div class="border-solid w-48 m-2 flex flex-col p-2 justify-between shadow-xl shadow-slate-400 rounded-lg group"
        id="card-produto-${produtoCatalogo.id}">
        <img
        src="./assets/img/${produtoCatalogo.imagem}"
        alt="Produto 1 do Magazine Hashtag."
        class="group-hover:scale-110 duration-300 my-3 roudend-lg"
        />
        <p class="text-sm">${produtoCatalogo.marca}</p>
        <p class="text-sm">${produtoCatalogo.nome}</p>
        <p class="text-sm">$${produtoCatalogo.preco}</p>
        <button class="bg-slate-950 hover:bg-slate-700 text-slate-200">
        <i class="fa-solid fa-cart-plus"></i>
        </button>
        </div>`;
        document.getElementById("container-produto").innerHTML += cartaoProduto;
    }
}
```



Adicionar ao Carrinho com Javascript

Agora que finalizamos todas as mudanças de estilo para o nosso cartão de produtos, é hora de continuar desenvolvendo a funcionalidade do botão de adição. Lembre-se de que criamos uma função chamada **"adicionarAoCarrinho()"**, que contém o código HTML completo do nosso cartão de produto. Esse código será exibido dentro do nosso carrinho de compras.

Para isso vamos importar no arquivo **cartaoProduto.js**, a nossa função **"adicionarAoCarrinho()".**

```
src > JS cartaoProduto.js > ...
1   import { catalogo } from "./utilidades";
2   import { adicionarAoCarrinho } from "./menuCarrinho";
```

Após a importação da nossa função, vamos adicionar ao botão de adicionar o **ID** que será gerado dinamicamente através de uma interpolação de string que substituirá o valor **${produtoCatalogo.id}** pelo valor do ID do produto específico do catálogo.

```
16      <button id="adicionar-${produtoCatalogo.id}" class="bg-slate-950 hover:bg-slate-700 text-slate-200">
17      <i class="fa-solid fa-cart-plus"></i>
18      </button>
```

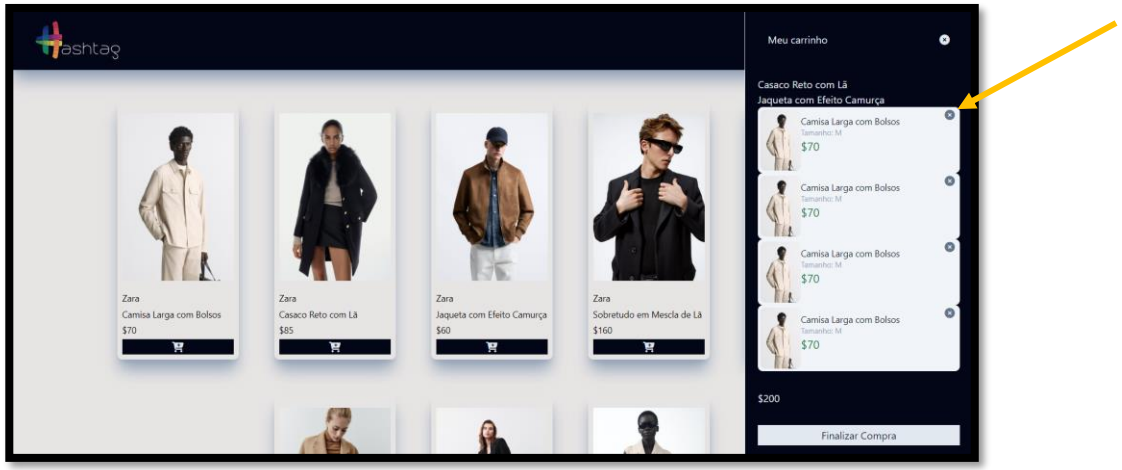
Adicionar ao Carrinho com Javascript

Dentro da nossa **função "renderizarCatalago()"** do arquivo **cartaoProduto.js** vamos adicionar mais uma iteração utilizando **'for'**.

O código irá percorrer os produtos no catálogo e configurar a ação de clique para cada botão de adicionar ao carrinho. Isso garante que, ao clicar em qualquer um dos botões, a função **adicionarAoCarrinho** seja acionada, permitindo a adição do produto correspondente ao carrinho de compras.

```
23      for (const produtoCatalogo of catalogo) {
24          document.getElementById(`adicionar-${produtoCatalogo.id}`).addEventListener('click', adicionarAoCarrinho);
25      }
```

Mas perceba que a funcionalidade de adicionar ao carrinho ainda não está 100% correta, já que apenas um produto está sendo adicionado ao carrinho de compras, independente do produto que o botão é clicado:



Adicionar ao Carrinho com Javascript

Para que a funcionalidade "adicionar ao carrinho" esteja totalmente funcional, começaremos por remover os elementos ** ** (como ** Casaco Reto com Lã ** e ** Jaqueta com Efeito Camurça **) do arquivo **index.html**. Esses elementos eram apenas visuais e foram usados para verificar o posicionamento do carrinho de compras.

Em seguida, faremos a importação do objeto **"catalogo"** para o arquivo **menuCarrinho.js**. Além disso, modificaremos a função **"adicionarAoCarrinho()"** para adicionar dinamismo à renderização dos produtos específicos para cada botão correspondente.

```
21  export function adicionarAoCarrinho(idProduto) {
22      const produto = catalogo.find((p) => p.id === idProduto);
```

Nesta função, recebemos o **ID de um produto como parâmetro**. Em seguida, usamos esse ID para percorrer a matriz **catalogo**. Utilizando o método **find()**, iteramos sobre cada produto (**p**) na lista até encontrar aquele que possui o mesmo ID passado como parâmetro. Uma vez encontrado, as informações desse produto são armazenadas na variável **produto**.

Com o produto identificado e suas informações disponíveis na variável **produto**, você pode prosseguir com as ações que deseja realizar. Isso poderia incluir etapas como adicionar o produto ao carrinho de compras ou executar qualquer outra lógica relacionada ao produto.

Adicionar ao Carrinho com Javascript

Finalizando a inteligência do nosso botão de adicionar ao carrinho, vamos dar dinamismo aos nossos elementos, para que eles sejam capazes de capturarem as informações de cada cartão de produtos específico.

```
export function adicionarAoCarrinho(idProduto) {
 const produto = catalogo.find((p) => p.id === idProduto);
 const containerProdutosCarrinho = document.getElementById("produtos-carrinho");
 const cartaoProdutoCarrinho = `<article class="flex bg-slate-100 rounded-lg p-1 relative">
   <button id="fechar-carrinho" class="absolute top-0 right-2">
     <i class="fa-solid fa-circle-xmark text-slate-500 hover:text-slate-800"></i>
   </button>
   <img
     src="./assets/img/${produto.imagem}"
     alt="Carrinho: ${produto.nome}"
     class="h-24 rounded-lg"
   />
   <div class="py-2">
     <p class="text-slate-900 text-sm">${produto.nome}</p>
     <p class="text-slate-400 text-xs">Tamanho: M</p>
     <p class="text-green-700 text-lg">$${produto.preco}</p>
   </div>
 </article>`;
 containerProdutosCarrinho.innerHTML += cartaoProdutoCarrinho;
}
```

Nesta estrutura, o botão de fechar agora tem um ID dinâmico **${produto.id}** para identificar qual produto ele está fechando. Também estamos usando os atributos do produto, como imagem, nome, tamanho e preço, para renderizar o cartão do produto dinamicamente.

Com essa implementação, a funcionalidade "adicionar ao carrinho" agora renderiza os produtos correspondentes no carrinho, proporcionando uma experiência interativa ao usuário.

Adicionar ao Carrinho com Javascript

Agora que modificamos a nossa função **"adicionarAoCarrinho()"**, precisamos alterar a forma que executamos/chamamos ela no nosso arquivo cartaoProduto.js, na função "renderizarCatalogo()".

() => adicionarAoCarrinho(produtoCatalogo.id): Isso é uma função de seta (arrow function). Quando o botão for clicado, essa função será chamada. A função está enviando o **id** do **produtoCatalogo** para a função **adicionarAoCarrinho()**.

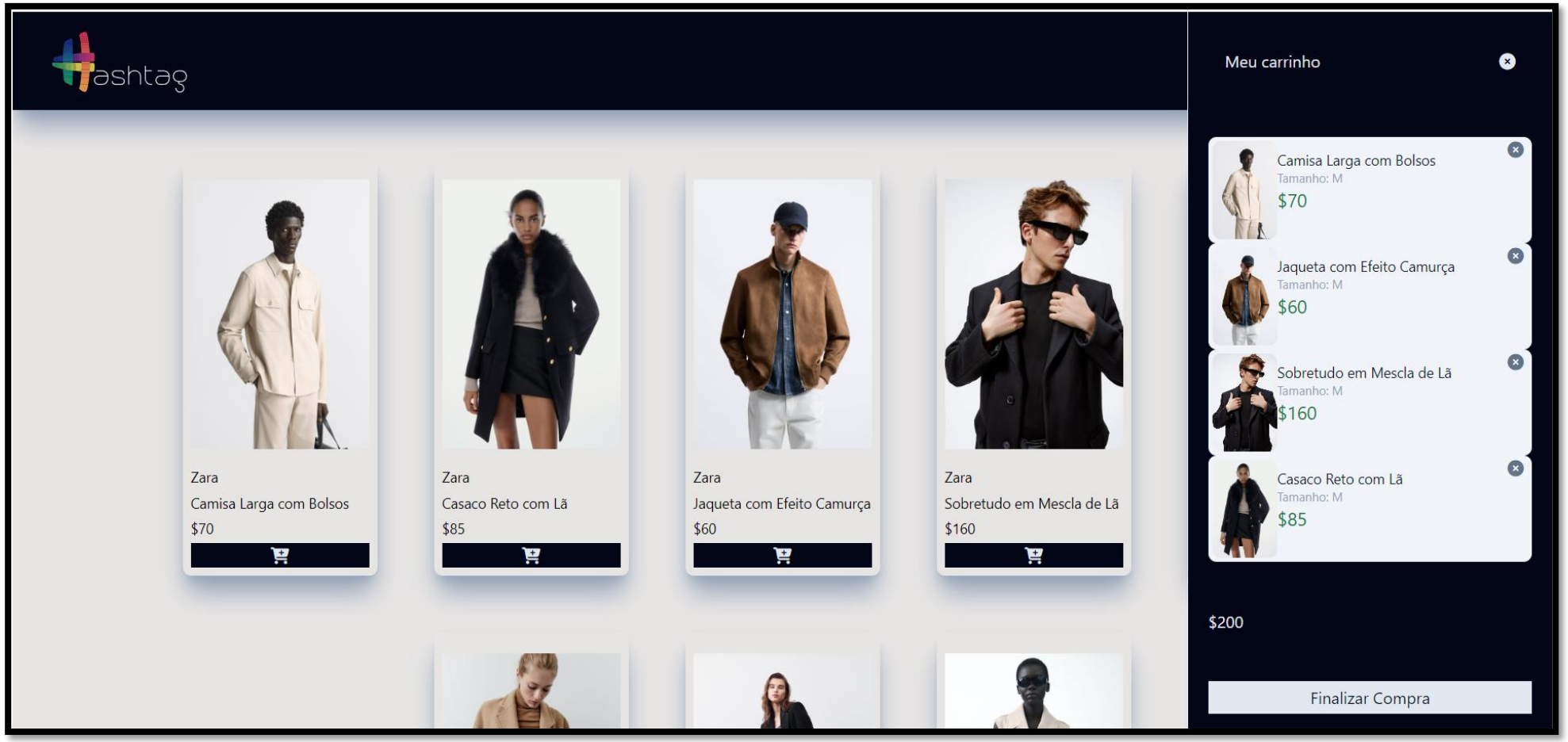
Em resumo, essa linha de código cria uma interação: quando o botão associado for clicado, a função **adicionarAoCarrinho()** será acionada, passando o **id** do produto associado ao botão como argumento. Isso permitirá que o produto seja adicionado ao carrinho corretamente.

src > JS cartaoProduto.js > ...

```
1  import { catalogo } from "./utilidades";
2  import { adicionarAoCarrinho } from "./menuCarrinho";
3
4  export function renderizarCatalago() {
5      for (const produtoCatalogo of catalogo) {
6          const cartaoProduto = `<div class="border-solid w-48 m-2 flex flex-col p-2 justify-between shadow-xl shadow-slate-400 rounded-lg group"
7          id="card-produto-${produtoCatalogo.id}">
8          <img
9          src="./assets/img/${produtoCatalogo.imagem}"
10         alt="Produto 1 do Magazine Hashtag."
11         class="group-hover:scale-110 duration-300 my-3 roudend-lg"
12         />
13         <p class="text-sm">${produtoCatalogo.marca}</p>
14         <p class="text-sm">${produtoCatalogo.nome}</p>
15         <p class="text-sm">$${produtoCatalogo.preco}</p>
16         <button id="adicionar-${produtoCatalogo.id}" class="bg-slate-950 hover:bg-slate-700 text-slate-200">
17         <i class="fa-solid fa-cart-plus"></i>
18         </button>
19         </div>`;
20         document.getElementById("container-produto").innerHTML += cartaoProduto;
21     }
22
23     for (const produtoCatalogo of catalogo) {
24         document.getElementById(`adicionar-${produtoCatalogo.id}`).addEventListener('click', () => adicionarAoCarrinho(produtoCatalogo.id));
25     }
26 }
```

Adicionar ao Carrinho com Javascript

Finalizamos a aula 3 do nosso intensivão Javascript com a funcionalidade Adicionar ao Carrinho completa:



Parte Bônus

Vídeos Complementares

Vídeos Complementares

Arrow Functions em JavaScript

Hashtag Programação

https://www.youtube.com/watch?v=khlpCir4auw&list=RDCMUCafFexaRoRylOKdzGBU6Pgg&index=2

Funções Anônimas em JavaScript

Hashtag Programação

734 visualizações • há 8 dias

https://www.youtube.com/watch?v=2EAfSeKZNiQ

Ainda não segue a gente no Instagram e nem é inscrito no nosso canal do Youtube? Então corre lá!

@hashtagtreinamentos

youtube.com/hashtag-treinamentos