



Intensivão de **JAVASCRIPT**

T

Apostila Completa

AULA 2

Guia passo a passo: Projeto Apple Watch



Parte 1

Instalando as ferramentas





Instalando as ferramentas – VS CODE

Caso você ainda não tenha o Visual Studio Code instalado, basta seguir os procedimentos abaixo. A instalação do VS Code é totalmente gratuita, e você pode usá-lo em seu computador sem precisar pagar nada. O link para fazer o download do programa é mostrado abaixo:

<https://code.visualstudio.com/>

- 1 – Baixe o arquivo de instalação correspondente ao seu sistema operacional (Windows, MacOS ou Linux).
- 2 – Execute o arquivo de instalação e siga as instruções na tela. Em geral, é só continuar clicando em "Próximo".
- 3 – No Windows, durante a instalação, marque a opção "Add to path" para adicionar o VS Code às suas variáveis de ambiente.



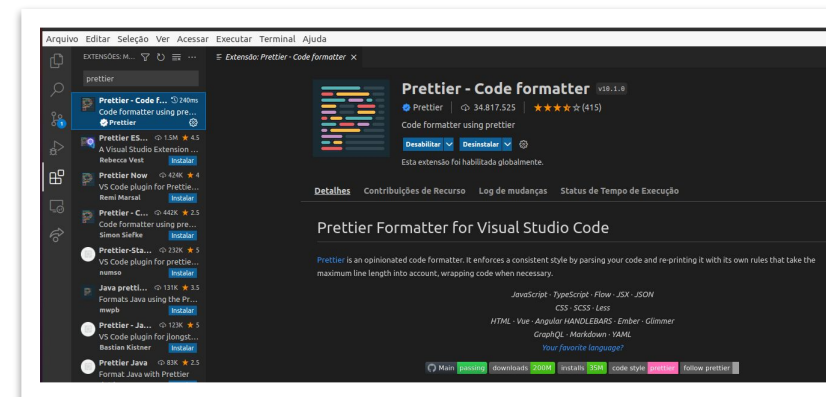


Instalando as ferramentas – VS CODE - EXTENSÕES

As extensões no Visual Studio Code são pequenos programas que adicionam recursos extras ao editor de código. Elas são como "plugins" que podem ser instalados para personalizar e estender as funcionalidades do VS Code.

A extensão Prettier adiciona a capacidade de formatar automaticamente o código, tornando-o mais legível e organizado.

Para instalar a extensão Prettier no Visual Studio Code:



- Clique no ícone de extensões no menu lateral esquerdo (ou use o atalho "Ctrl+Shift+X").
- Na barra de pesquisa, digite "Prettier".
- A extensão "Prettier - Code Formatter" deve aparecer nos resultados da pesquisa. Clique no botão "Instalar" ao lado dela.
- Após a instalação, você pode configurar o Prettier como o formatador padrão para o seu projeto. Para fazer isso, abra as configurações do Visual Studio Code (menu "File" > "Preferences" > "Settings" ou use o atalho "Ctrl+,").
- Na barra de pesquisa das configurações, digite "Default Formatter" e selecione a opção "Editor: Default Formatter".
- No campo de valor, digite "esbenp.prettier-vscode" e pressione Enter para salvar as alterações.
- Agora, o Prettier está instalado e configurado como o formatador padrão no Visual Studio Code. Você pode usar o comando "Format Document" (menu "Edit" > "Format Document" ou atalho "Shift+Alt+F") para formatar o código.

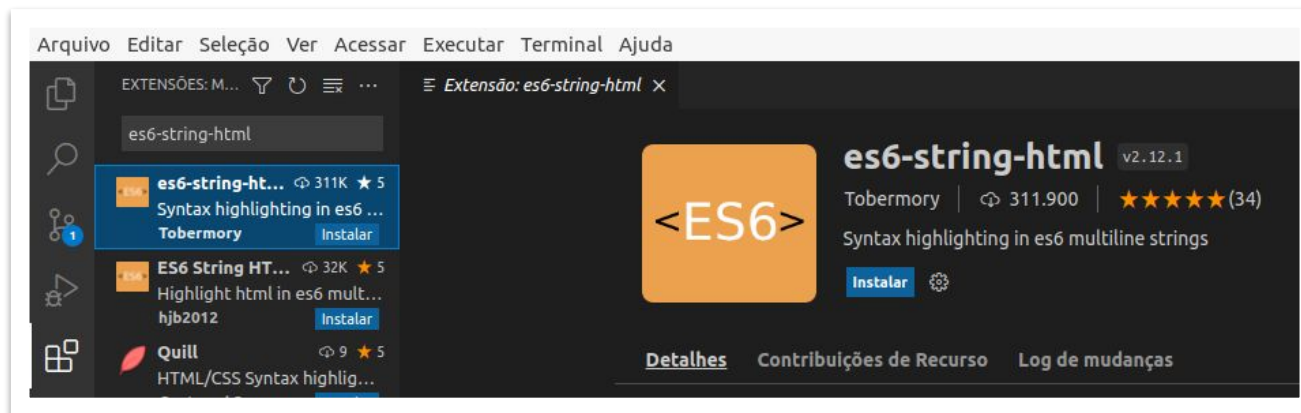


Instalando as ferramentas – VS CODE - EXTENSÕES

Já a extensão ES6 String HTML adiciona suporte para destacar a sintaxe do HTML dentro de strings de várias linhas no código JavaScript.

Para instalar a extensão ES6 String HTML no Visual Studio Code:

- Clique no ícone de extensões no menu lateral esquerdo (ou use o atalho "Ctrl+Shift+X").
- Na barra de pesquisa, digite "ES6 String HTML".
- A extensão "ES6 String HTML" deve aparecer nos resultados da pesquisa. Clique no botão "Instalar" ao lado dela.
- Após a instalação, a extensão irá adicionar suporte de realce de sintaxe para código HTML dentro de strings de várias linhas no ES6.



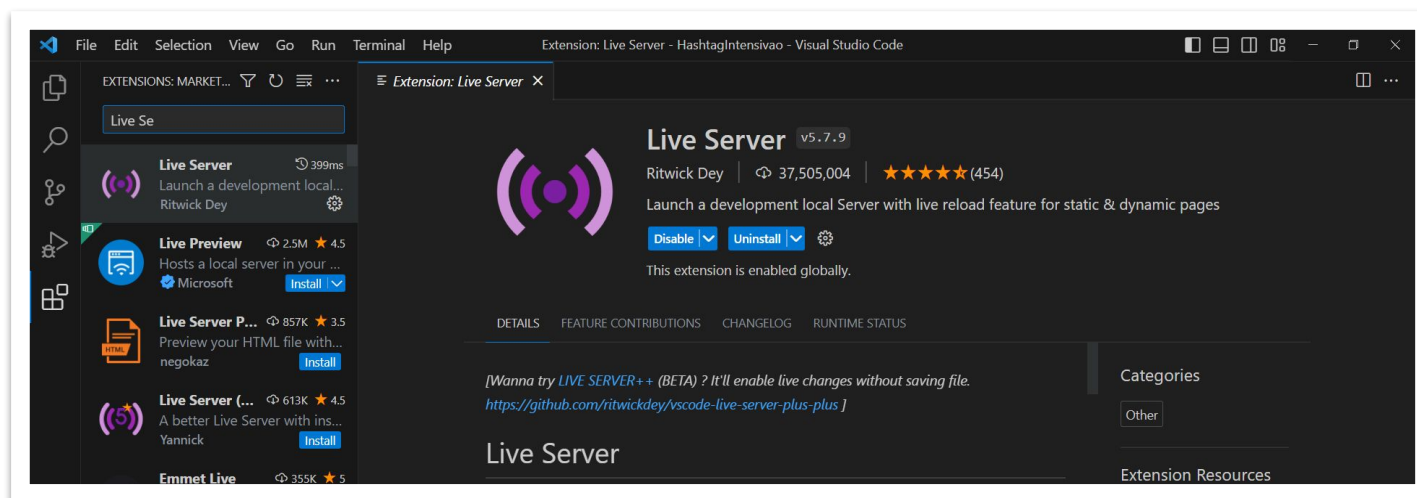


Instalando as ferramentas – VS CODE - EXTENSÕES

O Live Server é uma extensão muito útil para desenvolvimento web, pois facilita a visualização e a atualização instantânea das alterações feitas no código HTML.

Vou te explicar como baixar a extensão "Live Server" no VS Code:

- Abra o Visual Studio Code (VS Code) e vá para a seção de extensões.
- Pesquise por "Live Server" e clique em "Instalar" ao lado da extensão desenvolvida por Ritwick Dey.
- Aguarde a instalação e clique em "Gerenciar" para acessar as configurações da extensão.
- Personalize as configurações, se desejar.
- Abra um arquivo HTML no VS Code, clique com o botão direito do mouse e selecione "Abrir com Live Server".
- O Live Server iniciará um servidor local e abrirá o arquivo HTML no navegador padrão.





Instalando as ferramentas – Node.js

O Node.js é um ambiente de execução de código JavaScript do lado do servidor. Ele permite que você execute código JavaScript fora do navegador, o que significa que você pode criar aplicativos de servidor, scripts de linha de comando e muito mais usando JavaScript. Para instalar o Node.js, você pode seguir os seguintes passos:



- Acesse o site oficial do Node.js em <https://nodejs.org>.
- Na página inicial, você verá duas versões para download: LTS (Long Term Support) e Current. A versão LTS é recomendada para a maioria dos usuários, pois é mais estável e possui suporte a longo prazo. Selecione a versão LTS ou a versão mais recente, se preferir.
- Após selecionar a versão desejada, você será redirecionado para a página de download. Escolha o instalador adequado para o seu sistema operacional (Windows, macOS ou Linux) e clique no link para iniciar o download.
- Após o download ser concluído, execute o instalador e siga as instruções na tela para concluir a instalação.
- Após a instalação ser concluída, você pode verificar se o Node.js foi instalado corretamente abrindo o terminal ou prompt de comando e digitando o comando `node -v`. Se a versão do Node.js for exibida, significa que a instalação foi bem-sucedida.

Parte 2

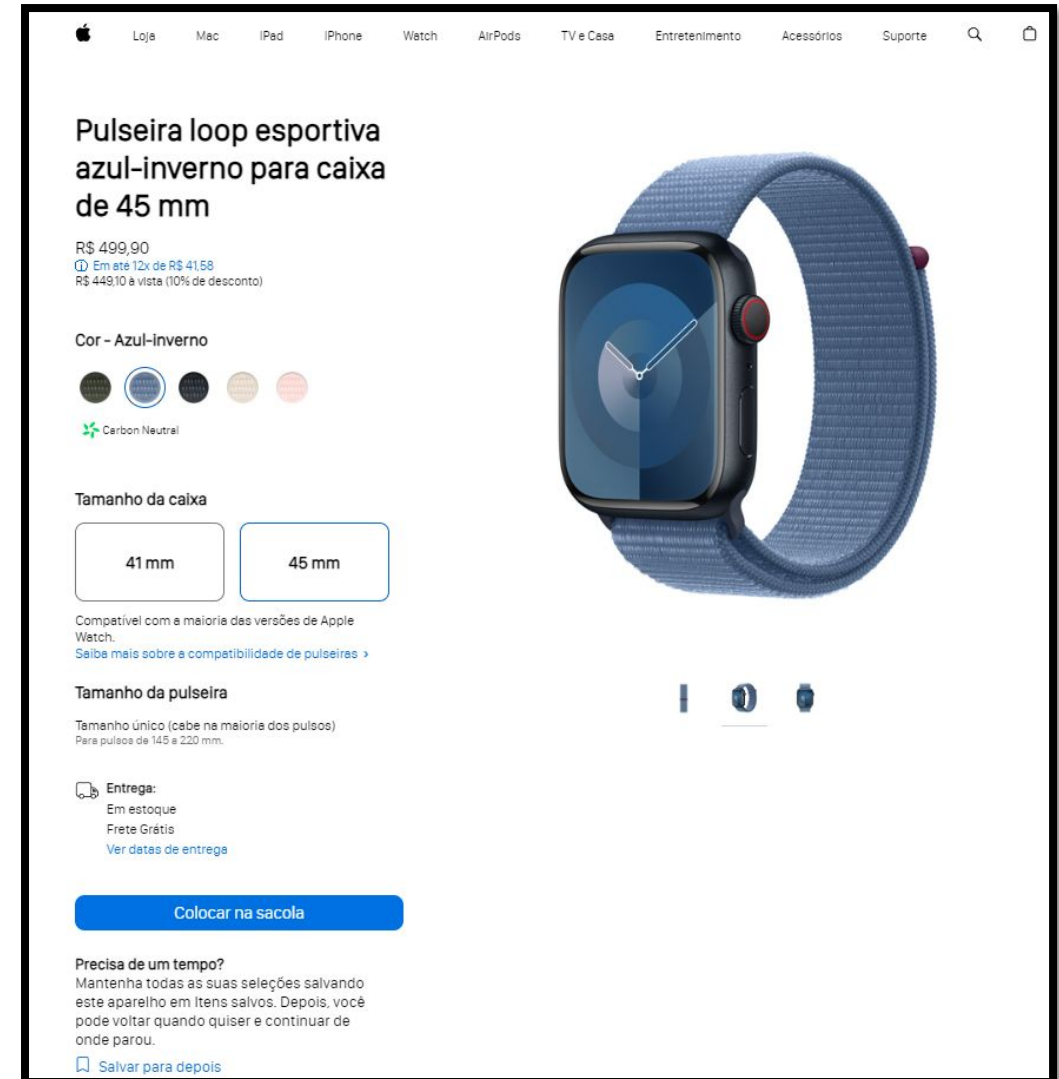
Apresentação do Projeto Apple Watch!

Apresentação do Projeto Apple Watch!

Bem-vindos à nossa plataforma interativa para personalização do **Apple Watch**, onde nos concentramos na implementação da lógica em JavaScript. A estrutura de HTML/CSS já está pronta, e agora nosso foco será permitir aos usuários manipular as opções de cor da pulseira e tamanho do Apple Watch. Inspirados pela ideia de oferecer uma experiência de personalização direto no navegador, nosso objetivo é tornar a customização do Apple Watch simples e envolvente.

Ao utilizar JavaScript, demonstraremos como a programação web pode facilitar a interação do usuário com as opções de personalização. Buscamos proporcionar uma experiência dinâmica e acessível, permitindo que os usuários visualizem as diferentes combinações de cor da pulseira e tamanho do Apple Watch em tempo real.

Dentro deste guia, você encontrará um passo a passo detalhado para implementar a lógica em JavaScript na nossa página de personalização. O conteúdo está excelente, e estamos confiantes de que você apreciará cada detalhe!



Parte 3

Tríade - HTML, CSS E Javascript

Tríade - HTML, CSS e Javascript

Bem-vindos à nossa aula sobre a criação de um audiobook do zero! Antes de mergulharmos no fascinante mundo da narração sonora, é crucial compreendermos a interação entre HTML, CSS e JavaScript – as ferramentas fundamentais que utilizaremos para dar vida ao nosso projeto. Essas três linguagens trabalham em conjunto para criar experiências web dinâmicas e envolventes.

- O **HTML (Hypertext Markup Language)** é a espinha dorsal do nosso conteúdo. Ele estrutura a informação, definindo os elementos que compõem a nossa página, como títulos, parágrafos, imagens e links. Imagine o HTML como o esqueleto do corpo, proporcionando a base necessária para a construção do conteúdo.
- O **CSS (Cascading Style Sheets)** entra em cena para adicionar estilo e estética à nossa página HTML. Enquanto o HTML fornece a estrutura, o CSS permite a personalização visual, aplicando cores, fontes, margens e outros estilos para criar uma experiência atraente e coesa. Analogamente, o CSS é como a roupa que vestimos sobre o esqueleto, tornando a aparência mais agradável e organizada.

Tríade - HTML, CSS e Javascript

- **JavaScript** é a linguagem de programação que dá vida à nossa experiência de audiobook, tornando-a interativa e dinâmica. Essencial para manipular HTML e CSS, o JavaScript entra em cena para adicionar, remover ou modificar conteúdo, respondendo às ações do usuário de forma inteligente. Em nosso projeto, ele não apenas gerencia a interface, mas também é responsável pela reprodução de áudio. Ao responder a cliques em botões ou comandos específicos, o JavaScript se torna o maestro por trás da cortina, coordenando a experiência auditiva de forma envolvente.

Juntos, HTML, CSS e JavaScript formam a tríade essencial para a criação de páginas web interativas e atraentes. Ao longo desta aula, exploraremos como essas linguagens se complementam para dar vida ao nosso projeto de audiobook, proporcionando não apenas uma experiência auditiva memorável, mas também uma compreensão mais profunda de como essas ferramentas se entrelaçam para criar a magia da web moderna. Vamos começar a jornada da criação do nosso audiobook!

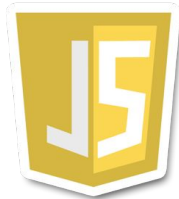


Parte 4

O Javascript

O Javascript

O que é JavaScript?



JavaScript é uma linguagem de programação de alto nível, dinâmica e interpretada, que é usada principalmente para tornar as páginas da web interativas. Com JavaScript, os desenvolvedores podem controlar o comportamento de elementos da página da web, criar animações, processar e manipular dados, entre outras coisas. A linguagem é executada no navegador do cliente, o que significa que o código é executado no dispositivo do usuário, em vez do servidor web, tornando a interação do usuário com a página da web mais rápida e eficiente.

O JavaScript desempenha um papel crucial na dinâmica e interatividade do nosso projeto de audiobook. Em essência ao ser incorporada nas páginas web, permite aos desenvolvedores controlar e aprimorar a experiência do usuário de maneira personalizada e eficiente.

Originalmente concebido em 1995 por Brendan Eich, o JavaScript foi criado para trazer animações e alertas às páginas da web. Ao longo do tempo, sua popularidade cresceu, e com o suporte da Microsoft em 1996, tornou-se uma linguagem-chave na programação web.

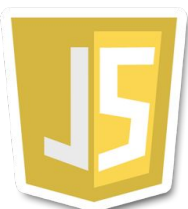
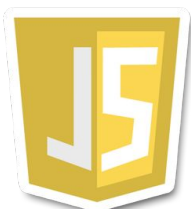
O Javascript

A evolução do JavaScript trouxe o ES6 (ECMAScript 2015), uma versão que introduziu recursos avançados, como declarações de variáveis mais flexíveis (let e const), classes, módulos e promessas. Esses aprimoramentos elevaram o JavaScript a uma linguagem mais poderosa e amigável.

Quando aplicamos o JavaScript ao nosso projeto de audiobook, ele se torna a força motriz por trás da interface interativa. Responsável por manipular HTML e CSS, o JavaScript permite a atualização dinâmica da interface do usuário. Ele opera no lado do cliente, garantindo uma resposta rápida e eficiente, sem depender constantemente do servidor.

No contexto específico do audiobook, o JavaScript assume o papel de maestro, controlando a reprodução de áudio, respondendo a interações do usuário, e adicionando camadas de dinamismo à experiência de audição. Com ele, podemos oferecer aos usuários a capacidade de controlar a reprodução, pular entre capítulos e interagir de maneira intuitiva com o conteúdo sonoro.

Em resumo, a implementação do JavaScript no nosso projeto de audiobook proporciona não apenas uma experiência interativa, mas também adiciona camadas de dinamismo e controle aos elementos sonoros, elevando a narrativa a uma experiência envolvente e personalizada para os usuários.



Parte 5

Primeiros passos



Primeiros Passos

Para iniciar o nosso projeto, vamos criar uma nova pasta com o nome "[IntensivaoAppleWatch](#)". Essa pasta será o local onde iremos armazenar todos os arquivos da nossa página Apple Watch. Dentro dessa pasta, colocaremos o arquivo Javascript, além de alguns arquivos adicionais que serão utilizados como o arquivo HTML, CSS, entre outros que estarão disponíveis para download.

Você pode fazer o download da pasta de todos os arquivos e renomear a pasta, ou manter a pasta de download chamada "[DoZero](#)".



Caso opte por criar uma nova pasta, vou te explicar passo a passo como criar uma pasta nova no Windows:

- Primeiro, abra o "Explorador de Arquivos" clicando no ícone da pasta amarela na barra de tarefas ou pressionando a tecla "Windows" + "E" no teclado.
- Navegue até o local onde você deseja criar a nova pasta. Por exemplo, você pode escolher criar a pasta na área de trabalho ou dentro de outra pasta existente.
- Com o local selecionado, clique com o botão direito do mouse em um espaço vazio da janela do Explorador de Arquivos. Um menu de opções será exibido.

Primeiros Passos

- No menu de opções, passe o cursor sobre a opção "Novo" e, em seguida, clique em "Pasta". Uma nova pasta será criada com o nome "Nova pasta" destacado.
- Digite o nome desejado para a nova pasta. No nosso caso, digite "IntensivaoAppleWatch" como o nome da pasta.
- Pressione a tecla "Enter" no teclado para confirmar o nome da pasta. A nova pasta será criada no local selecionado.



Nome	Status	Data de modificação	Tipo
 IntensivaoAppleWatch		26/09/2025 16:27	Pasta de arquivos

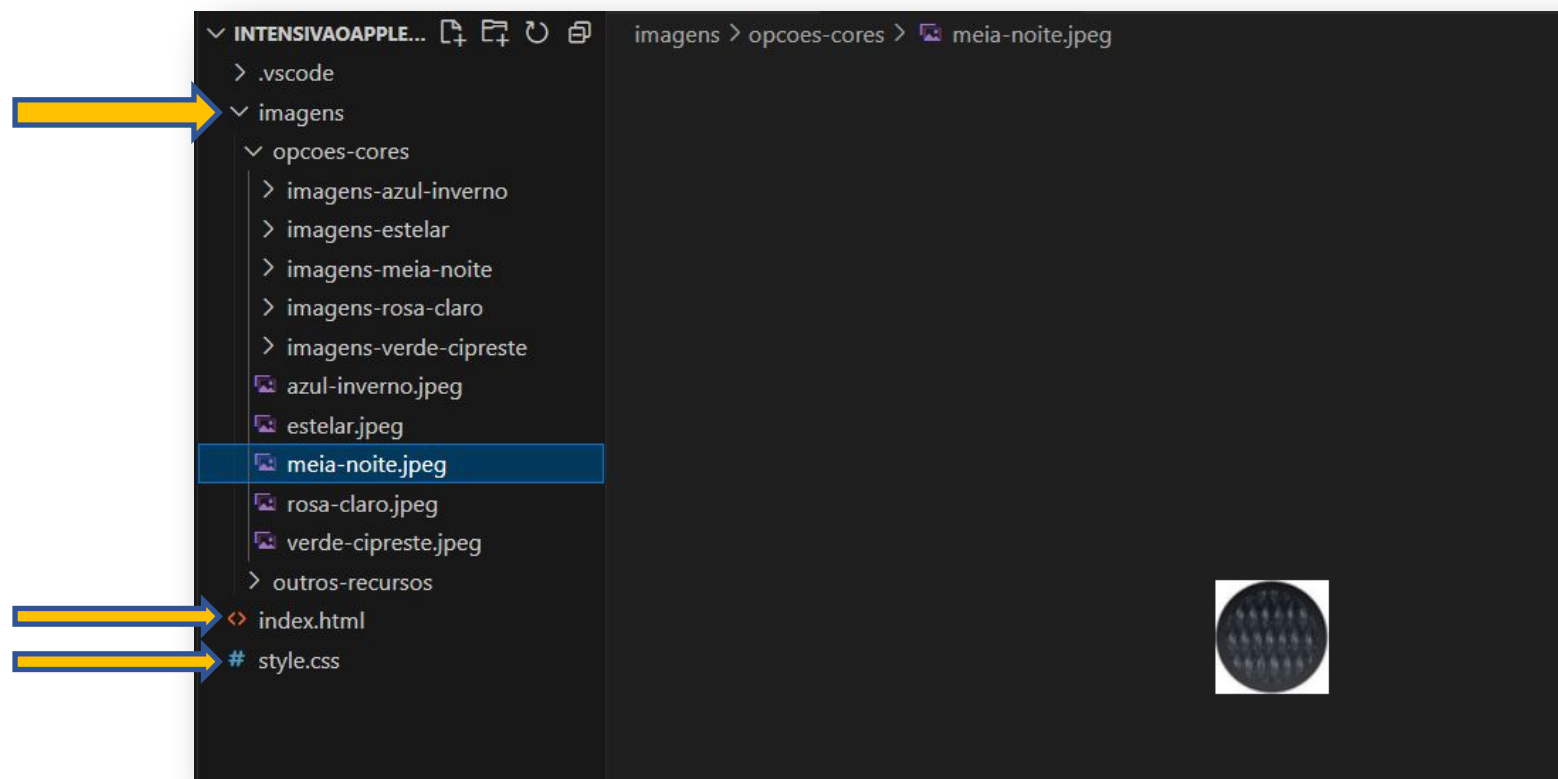
Primeiros Passos

Agora vamos abrir a nossa pasta no VS Code, que será o programa que utilizaremos para realizarmos a construção da página do Apple Watch. Para fazer isso, siga os passos abaixo:

- Certifique-se de ter o Visual Studio Code instalado no seu computador. Se ainda não tiver, você pode baixá-lo gratuitamente no site oficial do VS Code.
- Abra o Visual Studio Code clicando no ícone do programa na área de trabalho ou no menu Iniciar.
- No VS Code, clique em 'Arquivo' no menu superior e, em seguida, selecione 'Abrir Pasta'. Uma janela de seleção de pasta será exibida.
- Navegue até o local onde você criou a pasta 'IntensivaoAppleWatch'. Por exemplo, se você a criou na área de trabalho, vá até a área de trabalho.
- Selecione a pasta 'IntensivaoAppleWatch' e clique no botão 'Selecionar Pasta' na parte inferior direita da janela.
- A pasta 'IntensivaoAppleWatch' será aberta no VS Code. No painel lateral esquerdo, você encontrará o local específico para criar e guardar os arquivos do seu projeto.

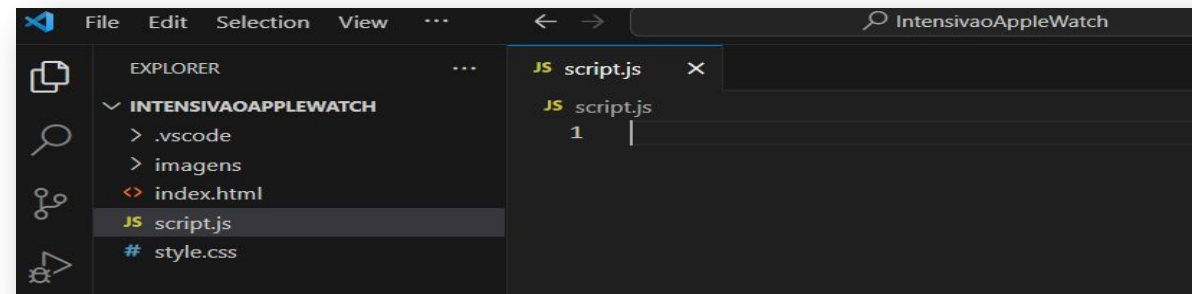
Primeiros Passos

Depois de baixar os arquivos do Projeto e abrir a pasta no VS Code, você encontrará diversos arquivos disponíveis. O arquivo `index.html` conterá os elementos da página, enquanto o arquivo `style.css` fornecerá a folha de estilos aplicada ao HTML. Além disso, haverá imagens representando as cores, pulseiras e Apple Watches.



Primeiros Passos

Dentro da pasta principal do projeto, vamos criar o arquivo Javascript. Para fazer isso, clique em "Novo Arquivo" e nomeie o arquivo de script.js.



Para associar o arquivo script.js ao seu index.html e adicionar funcionalidades à sua página Apple Watch, você precisa adicionar uma tag **<script>** no final do seu arquivo HTML, antes do fechamento da tag **</body>** ou na abertura da tag **<body>**.

```
<script src="./script.js"></script>
</body>
```

Ao vincular o arquivo script.js ao seu arquivo HTML, você permite que as funcionalidades e lógicas definidas no script.js sejam aplicadas aos elementos da sua página, como o Apple Watch. Certifique-se de que o nome do arquivo script.js corresponda exatamente ao nome do arquivo que você criou na pasta raiz do seu projeto.

Primeiros Passos

Opcional – Adicionar o atributo `defer` à tag `<script>` é uma prática recomendada para garantir que o script seja executado apenas após o carregamento completo do documento HTML. Com o uso de `defer`, o navegador inicia o download do arquivo `script.js` enquanto continua analisando e renderizando o HTML. A execução do script é adiada até que o HTML seja completamente carregado.

Essa abordagem é vantajosa porque permite que o navegador continue carregando e exibindo a página sem esperar pela execução do script. Isso pode melhorar a velocidade de carregamento da página e a experiência do usuário, especialmente em páginas com muitos elementos ou scripts JavaScript complexos.

Ao adicionar o atributo `defer` à tag `<script>`, você garante que o script seja executado de maneira eficiente, sem atrasar desnecessariamente a renderização da página. Isso pode contribuir para uma experiência de usuário mais fluida e responsiva.

```
<> index.html > html > body > nav > ul > li
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Hashtag Watch</title>
7      <link rel="stylesheet" href="./style.css" />
8    </head>
9    <body>
10     <script src="./script.js" defer></script>
11     <nav>
```

Parte 6

Construindo a Inteligência da página

Construindo a Inteligência da página

Lembrando alguns conceitos visto na aula anterior , uma variável em Javascript é um contêiner para armazenar dados. Ela pode conter diferentes tipos de informações, como números, strings, objetos, ou funções. As variáveis ajudam a tornar nosso código mais dinâmico e flexível.

Em Javascript, **const** e **let** são palavras-chave para declarar variáveis. **const** é usada para declarar uma variável cujo valor não pode ser alterado após a atribuição inicial, enquanto **let** declara uma variável com escopo de bloco, permitindo reatribuição de valor dentro do mesmo escopo. Essas diferenças nos permitem controlar a mutabilidade dos dados em nossos programas de forma mais precisa.

O tipo de valor de uma variável em JavaScript refere-se ao tipo de dado que ela contém. Isso pode incluir números, strings (texto), booleanos (verdadeiro/falso), arrays (listas de dados) e objetos (estruturas de dados complexas).

As aspas simples (") e as aspas duplas (") são usadas para delimitar strings em JavaScript. Ambos os tipos de aspas têm o mesmo propósito: criar uma sequência de caracteres. Por exemplo, **"hello"** e **'hello'** são strings idênticas.

O número **45** é um valor numérico, enquanto a string **"45"** é uma representação textual do número 45. É importante notar que, embora possam parecer semelhantes, em JavaScript, **45** é um número e **"45"** é uma string. Isso significa que podem ser tratados de maneira diferente dependendo do contexto, especialmente em operações matemáticas e comparações.

Construindo a Inteligência da página

Exemplos de atribuição de variáveis:

```
JS script.js > ...  
1  const variavelTexto = "hello";  
2  const variavelNumero = 45;  
3  const variavelNumeroEmFormaDeTexto = "45";  
4  let variavelMutavel = "Meu valor";  
5  variavelMutavel = "Eu posso mudar meu valor";  
6
```

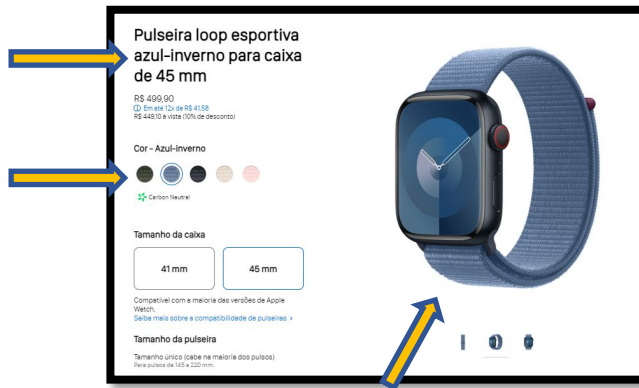
Um outro tipo de variável que vamos aprender e utilizar no nosso projeto é do tipo Objeto. Em JavaScript, um objeto é uma estrutura de dados complexa que pode conter múltiplos valores, conhecidos como propriedades, organizados em pares chave-valor. As propriedades de um objeto podem incluir não apenas valores simples, como números e strings, mas também outras estruturas de dados, como arrays e até mesmo funções.

Um objeto é definido entre chaves `{}`, e as propriedades são listadas dentro dessas chaves, separadas por vírgulas. Cada propriedade é composta por um nome (chave) e um valor, separados por dois pontos `:`.

```
JS script.js > ...  
1  const primeiraVariavel = {  
2    nomeDaInformação: "Texto associado ao campo",  
3  };  
4
```

Construindo a Inteligência da página

Vamos criar objetos para representar as cores das pulseiras. Esses objetos nos ajudarão a organizar e controlar as mudanças na nossa página, como imagens e títulos, com base nas cores selecionadas.



```
const verdeCipreste = {
  nome: "Verde-cipreste",
  nomePastaImagens: "imagens-verde-cipreste",
};
const azulInverno = {
  nome: "Azul-inverno",
  nomePastaImagens: "imagens-azul-inverno",
};
const meiaNoite = {
  nome: "Meia-noite",
  nomePastaImagens: "imagens-meia-noite",
};
const estelar = {
  nome: "Estelar",
  nomePastaImagens: "imagens-estelar",
};
const rosaClaro = {
  nome: "Rosa-claro",
  nomePastaImagens: "imagens-rosa-claro",
};
```

Construindo a Inteligência da página

Agora vamos aprender um tipo de dados, chamado array. Um array é como uma lista que pode conter vários itens, como números, strings ou até mesmo objetos.

```
const opcoesCores = [verdeCipreste, azulInverno, meiaNoite, estelar, rosaClaro];
```

A linha de código **const opcoesCores = [verdeCipreste, azulInverno, meiaNoite, estelar, rosaClaro];** cria uma lista chamada **opcoesCores** que contém várias opções de cores para pulseiras. Cada cor é representada por um objeto, como **verdeCipreste**, **azulInverno**, etc.

Imagine que é como fazer uma lista de compras, onde você lista todas as cores de pulseiras que você tem disponíveis. Cada cor é como um item na lista, e a lista completa é armazenada na variável **opcoesCores**. Isso facilita acessar e trabalhar com as cores disponíveis em seu código.

Índices em arrays são números que representam a posição de um elemento na lista. Começando do zero, cada elemento em um array possui um índice único que permite acessá-lo ou modificá-lo.

Construindo a Inteligência da página

Para acessar a lista **opcoesCores**, você pode usar os índices dos elementos dentro do array. Cada elemento dentro do array pode ser acessado usando seu índice correspondente. Por exemplo:

- Para acessar o primeiro elemento (verdeCipreste): **opcoesCores[0]**
- Para acessar o segundo elemento (azulInverno): **opcoesCores[1]**

E assim por diante...



Agora, para manipular os elementos desejados da nossa página através do JavaScript, iremos armazená-los em variáveis usando o método **document.getElementById()**, que utiliza os IDs atribuídos a eles.

Arquivo script.js – variável que armazena o elemento:

```
1  const tituloProduto = document.getElementById("titulo-produto");
```

Arquivo index.html – Elemento título:

```
<h1 id="titulo-produto">Pulseira loop esportiva azul-inverno para caixa de 45 mm</h1>
```

Construindo a Inteligência da página

Para organizar nosso projeto, vamos usar a ordem dos elementos que queremos manipular, começando com 0 e seguindo a sequência 1, 2, 3, se houver mais de dois elementos para selecionar nas opções. Por exemplo, as cores da pulseira serão manipuladas na ordem 0, 1, 2, 3, 4, pois são 5 cores. Já os tamanhos da pulseira (41 mm e 45 mm) serão manipulados por 0 e 1, e assim por diante com os demais elementos. Isso nos ajuda a entender e organizar como iremos trabalhar com cada parte do nosso projeto.

Pulseira loop esportiva verde-cipreste para caixa de 45 mm

R\$ 499,90

Em até 12x de R\$ 41,58

R\$ 449,10 à vista (10% de desconto)

0 Verde-cipreste

1

2

3

4

Carbon Neutral

Tamanho da caixa

41 mm 0

45 mm 1

Compatível com a maioria das versões de Apple Watch.

Saiba mais sobre a compatibilidade de pulseiras >

Tamanho da pulseira

Tamanho único (cabe na maioria dos pulsos)

Para pulsos de 145 a 220 mm.



0 1 2

1 0 2

29

Construindo a Inteligência da página

Os radio buttons, ou botões de rádio, são elementos do formulário HTML que permitem aos usuários escolher apenas uma opção de um conjunto predefinido de escolhas exclusivas. Quando um radio button é selecionado, ele indica que uma opção específica foi escolhida em um grupo de opções.

No nosso projeto, usamos radio buttons quando queremos que o usuário selecione uma única opção, como a cor da pulseira, o tamanho ou a imagem. Cada grupo de radio buttons representa um conjunto de opções exclusivas, garantindo que o usuário faça apenas uma escolha dentro desse conjunto.

```
<input type="radio" name="opcao-cor" id="0-cor" onclick="atualizarCorSelecionada()">  
<input type="radio" name="opcao-cor" id="1-cor" onclick="atualizarCorSelecionada()">  
<input type="radio" name="opcao-cor" id="2-cor" onclick="atualizarCorSelecionada()">  
<input type="radio" name="opcao-cor" id="3-cor" onclick="atualizarCorSelecionada()">  
<input type="radio" name="opcao-cor" id="4-cor" onclick="atualizarCorSelecionada()">
```



Construindo a Inteligência da página

Agora que entendemos como manipular e organizar os elementos, além de compreender o conceito de radio button, vamos começar criando a funcionalidade que permitirá aos usuários escolherem o tamanho da pulseira entre 41 mm e 45 mm.

Isso significa que vamos criar uma interação onde os usuários podem selecionar apenas um desses tamanhos por vez. Essa funcionalidade vai garantir que a escolha do tamanho da pulseira seja clara e precisa, conforme as opções disponíveis.


O elemento que representa os tamanhos no arquivo index.html:

```
<div id="tamanho-caixa">
  <h2>Tamanho da caixa</h2>
  <div id="opcoes-tamanho">
    <input type="radio" name="opcao-tamanho" id="0-tamanho" onclick="atualizarTamanho()">
    <label for="0-tamanho">41 mm</label>
    <input type="radio" name="opcao-tamanho" id="1-tamanho" onclick="atualizarTamanho()" checked>
    <label for="1-tamanho">45 mm</label>
  </div>
```

Vamos adicionar a propriedade **onclick**, que tem a finalidade de definir um evento de clique, ao elemento HTML. Atribuiremos a ele a função que criaremos chamada **atualizarTamanho()**.

Construindo a Inteligência da página

Vamos construir a funcionalidade **atualizarTamanho()** e para auxiliar a nossa função vamos criar uma lista que possuirá dois elementos do tipo texto (string) que será os tamanhos da pulseira:



```
const opcoesCores = [verdeCipreste, azulInverno, meiaNoite, estelar, rosaClaro];  
const opcoesTamanho = ["41 mm", "45 mm"];
```

Nossa função **atualizarTamanho()** irá iniciar com a seguinte estrutura:

```
function atualizarTamanho() {  
  const opcaoTamanhoSelecionado = document  
    .querySelector('[name="opcao-tamanho"]:checked')  
    .id.charAt(0);  
  alert(opcaoTamanhoSelecionado); // irá mostrar na página o valor que a variável está armazenando  
}
```

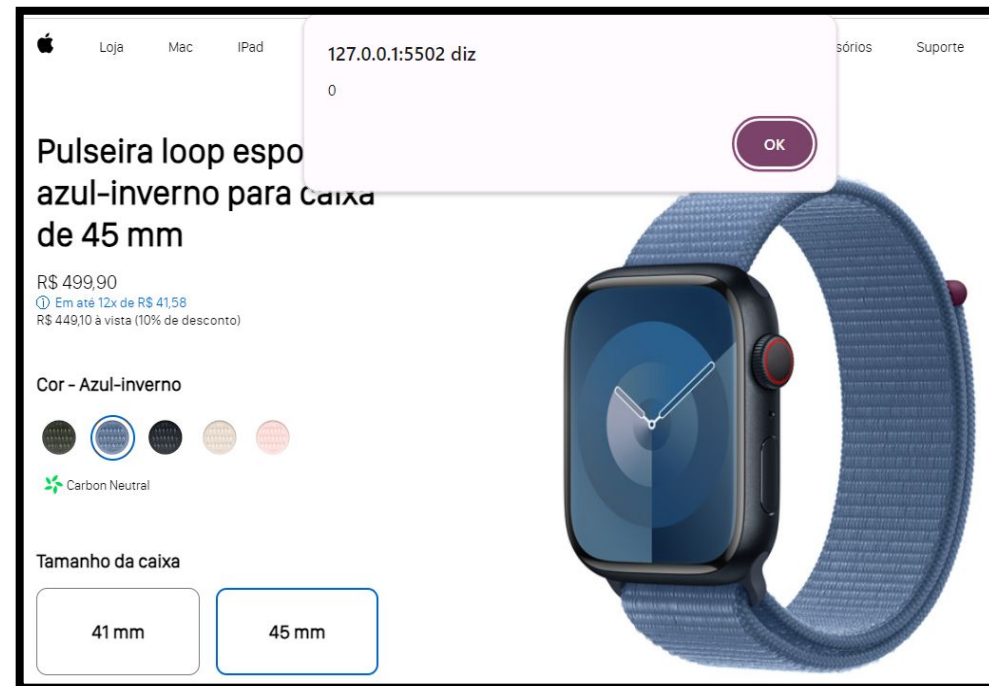

Construindo a Inteligência da página

O código `document.querySelector('[name="opcao-tamanho"]:checked')` procura por um elemento no documento HTML que tenha o atributo **name** igual a "opcao-tamanho" e que esteja marcado como selecionado (checked). Isso é comum quando temos radio buttons agrupados, como as opções de tamanho de pulseira.

```
<input type="radio" name="opcao-tamanho" id="1-tamanho" onclick="atualizarTamanho()" checked>
```

Depois, `.id.charAt(0)` é usado para acessar o atributo **id** desse elemento selecionado e pegar apenas o primeiro caractere desse ID. Isso nos ajuda a identificar o tamanho selecionado entre as opções disponíveis.

Por exemplo, se tivermos IDs como "0-tamanho" e "1-tamanho" para representar as opções de tamanho, usar `charAt(0)` nos permite pegar apenas o primeiro caractere, que seria "0" ou "1". Isso nos ajuda a trabalhar com essa informação de forma mais direta e eficiente.



Construindo a Inteligência da página

Vamos entender mais profundamente alguns conceitos que vimos no slide anterior.

Método em JavaScript:

Em JavaScript, um método é uma função que está associada a um objeto. Métodos são chamados em objetos para realizar operações específicas relacionadas a esse objeto. Por exemplo, em um objeto **carro**, você pode ter métodos como **ligar()**, **desligar()**, **acelerar()**, etc. Os métodos permitem que os objetos realizem ações ou manipulações específicas de acordo com o contexto em que estão inseridos.

charAt() é um método de string em JavaScript que retorna o caractere em uma determinada posição em uma string. Ele recebe um índice como argumento e retorna o caractere na posição correspondente. O índice começa em 0, ou seja, o primeiro caractere está na posição 0, o segundo na posição 1 e assim por diante. Por exemplo, **charAt(0)** retorna o primeiro caractere da string, **charAt(1)** retorna o segundo caractere e assim por diante.

Construindo a Inteligência da página

Continuando com nossa função, vamos atualizar a variável de controle **tamanhoSelecionado** com o valor da variável **opcaoTamanhoSelecionado**. Esta última armazena o elemento que representa o tamanho selecionado em nossa página.

```
✓ function atualizarTamanho() {  
  const opcaoTamanhoSelecionado = document  
    .querySelector('[name="opcao-tamanho"]:checked')  
    .id.charAt(0);  
  tamanhoSelecionado = opcaoTamanhoSelecionado;  
}
```

O próximo passo é conseguir manipular o título da página, trocando os valores de tamanho da pulseira de acordo com o que for selecionado pelo usuário:

```
function atualizarTamanho() {  
  const opcaoTamanhoSelecionado = document  
    .querySelector('[name="opcao-tamanho"]:checked')  
    .id.charAt(0);  
  tamanhoSelecionado = opcaoTamanhoSelecionado;  
  tituloProduto.innerText =  
    "Pulseira loop esportiva azul-inverno para caixa de " +  
    opcoesTamanho[tamanhoSelecionado];  
}
```

Construindo a Inteligência da página

A linha **tituloProduto.innerText = "Pulseira loop esportiva azul-inverno para caixa de " + opcoesTamanho[tamanhoSelecionado];** é responsável por atualizar o texto do elemento HTML referenciado pela variável **tituloProduto**.

- **tituloProduto**: É uma variável que armazena a referência do elemento HTML, o título do produto da página.
- **innerText**: É uma propriedade desse elemento HTML que permite definir ou obter o texto contido dentro dele.
- **"Pulseira loop esportiva azul-inverno para caixa de "**: É uma parte do texto que será exibido no elemento **tituloProduto**.
- **opcoesTamanho[tamanhoSelecionado]**: Aqui, estamos usando a variável **tamanhoSelecionado** para acessar o array **opcoesTamanho** e obter o valor associado ao tamanho escolhido pelo usuário. Lembrando que **opcoesTamanho** é um array que contém as opções de tamanho da pulseira.

Portanto, o texto exibido será algo como "Pulseira loop esportiva azul-inverno para caixa de 41mm" ou "Pulseira loop esportiva azul-inverno para caixa de 45mm", dependendo do tamanho selecionado pelo usuário.

Pulseira loop esportiva
azul-inverno para caixa
de 45 mm

Pulseira loop esportiva
azul-inverno para caixa
de 41 mm


Construindo a Inteligência da página

Quando selecionamos o tamanho da pulseira, uma outra interação acontece: a imagem diminui ou aumenta de acordo com a escolha feita. Essa mudança na imagem ocorre através de uma alteração no estilo (tamanho da imagem) e, para isso, já temos uma classe definida em nosso arquivo style.css.

```
.caixa-pequena {  
  transform: scale(0.9);  
}
```


O elemento de imagem do relógio no arquivo index.html que receberá a classe por via do Javascript, e irá ter ou não de acordo com a lógica implementada, ou seja, qual tamanho o usuário escolher.

```
<div id="visualizacao">  
  
```



Vamos armazenar esse elemento no nosso arquivo script.js para que possamos manipulá-lo adequadamente.

```
1  const tituloProduto = document.getElementById("titulo-produto");  
2  const imagemVisualizacao = document.getElementById("imagem-visualizacao");  
3
```



Construindo a Inteligência da página


E agora vamos verificar através da estrutura condicional `if/else` qual tamanho está sendo selecionado, para aplicarmos a classe da seguinte forma:

`if (opcoesTamanho[tamanhoSelecionado] === "41 mm"):`

Aqui estamos verificando se o tamanho selecionado, representado por **`opcoesTamanho[tamanhoSelecionado]`**, é igual a "41 mm".

Se a condição for verdadeira (ou seja, se o tamanho selecionado for "41 mm"), o código dentro das chaves **`{}`** do bloco **`if`** será executado.

- **`imagemVisualizacao.classList.add("caixa-pequena")`**
;: Isso adiciona a classe CSS chamada "caixa-pequena" ao elemento representado pela variável **`imagemVisualizacao`**. Isso geralmente significa que a imagem terá seu tamanho reduzido, pois a classe "caixa-pequena" contém regras de estilo CSS que alteram o tamanho da imagem.



```
function atualizarTamanho() {
  const opcaoTamanhoSelecionado = document
    .querySelector('[name="opcao-tamanho"]:checked')
    .id.charAt(0);
  tamanhoSelecionado = opcaoTamanhoSelecionado;
  tituloProduto.innerHTML =
    "Pulseira loop esportiva azul-inverno para caixa de " +
    opcoesTamanho[tamanhoSelecionado];

  if (opcoesTamanho[tamanhoSelecionado] === "41 mm") {
    imagemVisualizacao.classList.add("caixa-pequena");
  } else {
    imagemVisualizacao.classList.remove("caixa-pequena");
  }
}
```


Construindo a Inteligência da página

Se a condição do **if** for falsa (ou seja, se o tamanho selecionado não for "41 mm"), o código dentro das chaves **{}** do bloco **else** será executado.

- **imagemVisualizacao.classList.remove("caixa-pequena");**: Isso remove a classe CSS chamada "caixa-pequena" do elemento representado pela variável **imagemVisualizacao**. Se o tamanho selecionado não for "41 mm", isso significa que queremos que a imagem volte ao seu tamanho original, e essa classe é removida para restaurar o estilo padrão.

```
function atualizarTamanho() {  
  const opcaoTamanhoSelecioneado = document  
    .querySelector('[name="opcao-tamanho"]:checked')  
    .id.charAt(0);  
  tamanhoSelecioneado = opcaoTamanhoSelecioneado;  
  tituloProduto.innerText =  
    "Pulseira loop esportiva azul-inverno para caixa de " +  
    opcoesTamanho[tamanhoSelecioneado];  
  
  if (opcoesTamanho[tamanhoSelecioneado] === "41 mm") {  
    imagemVisualizacao.classList.add("caixa-pequena");  
  } else {  
    imagemVisualizacao.classList.remove("caixa-pequena");  
  }  
}
```


Construindo a Inteligência da página

Com a funcionalidade de tamanho implementada, avançaremos para a alteração da imagem do relógio. Faremos isso no seguinte elemento HTML:

```
<ul id="selecionar-imagem">
  <li><input type="radio" name="opcao-imagem" id="0-imagem" onclick="atualizarImagemSelecionada()">
  <li><input type="radio" name="opcao-imagem" id="1-imagem" onclick="atualizarImagemSelecionada()">
  <li><input type="radio" name="opcao-imagem" id="2-imagem" onclick="atualizarImagemSelecionada()">
</ul>
```

Vamos adicionar a propriedade **onclick** ao elemento HTML. Atribuiremos a ele a função que criaremos chamada **atualizarImagemSelecionada()**.

Também iremos atribuir uma variável controle, como fizemos com o tamanho:



```
let tamanhoSelecionado = 1;
let imagemSelecionada = 1;
```


Construindo a Inteligência da página

Vamos criar uma função semelhante à **atualizarTamanho**, mas selecionando a propriedade **name** que contém o valor do grupo das imagens, chamado "opcao-imagem". Atualizaremos a variável de controle com o elemento armazenado em uma nova variável criada dentro da função, chamada **opcaoImagemSelecionada**.

```
function atualizarImagemSelecionada() {  
  const opcaoImagemSelecionada = document  
    .querySelector('[name="opcao-imagem"]:checked')  
    .id.charAt(0);  
  imagemSelecionada = opcaoImagemSelecionada;  
}
```

Resumindo, a função chamada **atualizarImagemSelecionada()** busca o elemento de imagem selecionado pelo usuário. Ele encontra o elemento que tem o atributo **name** igual a "opcao-imagem" e que está marcado como selecionado (checked) no documento HTML. Em seguida, armazena o primeiro caractere do ID desse elemento na variável **imagemSelecionada**.

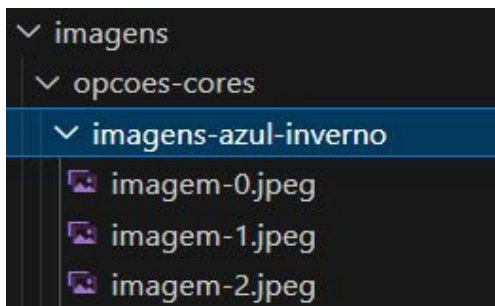
```
id='0-imagem"  
id='1-imagem"  
id='2-imagem"
```

Construindo a Inteligência da página

Para prosseguir, precisamos adicionar uma variável de controle para a cor selecionada. Estamos mantendo uma ordem numérica (0, 1, 2...) para gerenciar e manipular os elementos.

```
let tamanhoSelecionado = 1;  
let imagemSelecionada = 1;  
let corSelecionada = 1;
```

Para manipular e criar a lógica para alterar a imagem com a cor selecionada, precisamos observar como os nomes das imagens estão implementados.



Pasta "imagens": Esta é a pasta raiz que contém todas as opções de cores para as imagens das pulseiras.

Pasta "opcoes-cores": Dentro da pasta "imagens", há uma pasta chamada "opcoes-cores", que contém as diferentes opções de cores disponíveis para as pulseiras.

Pasta da cor selecionada: Por exemplo, se selecionarmos a cor "azul-inverno", iremos para a pasta "azul-inverno" dentro da pasta "opcoes-cores".

Imagens numeradas: Dentro da pasta da cor selecionada, as imagens estão numeradas de acordo com a ordem. Por exemplo, temos "0-imagem.jpeg", "1-imagem.jpeg", "2-imagem.jpeg" e assim por diante.

Portanto, para alterar a imagem com a cor selecionada, precisamos acessar a pasta correspondente à cor selecionada e carregar a imagem específica com base na ordem numerada das opções disponíveis. Isso nos permite exibir a imagem correta associada à cor selecionada pelo usuário.

Construindo a Inteligência da página

```
function atualizarImagemSelecionada() {  
  const opcaoImagemSelecionada = document  
    .querySelector('[name="opcao-imagem"]:checked')  
    .id.charAt(0);  
  imagemSelecionada = opcaoImagemSelecionada;  
  
  imagemVisualizacao.src = "./imagens/opcoes-cores/" + opcoesCores[corSelecionada].nomePastaImagens + "/imagem-" + imagemSelecionada + ".jpeg";  
}
```

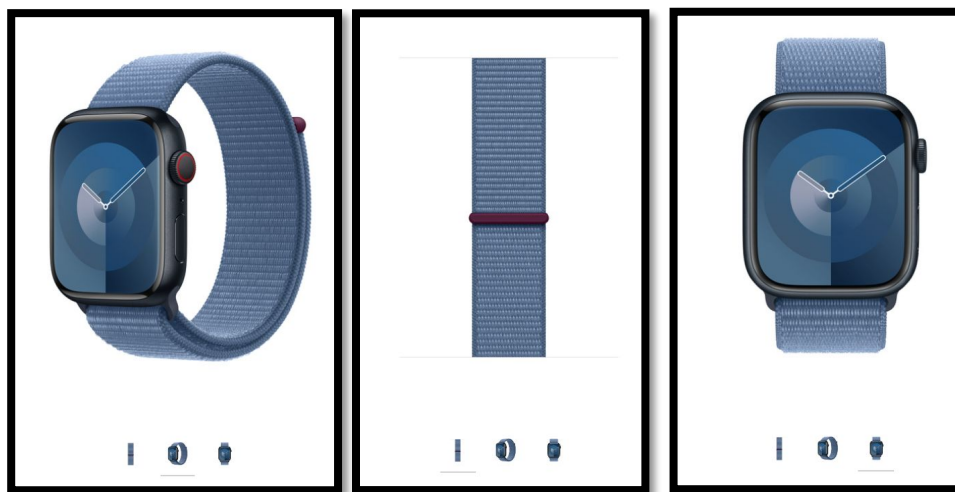
Vamos explicar as instruções que vamos adicionar em nosso projeto, dentro da função **atualizarImagemSelecionada()**:

- **imagemVisualizacao.src**: Define o atributo **src** do elemento HTML **imagemVisualizacao**, responsável por carregar a imagem a ser exibida.
- **"/imagens/opcoes-cores/"**: Especifica o caminho até a pasta raiz onde estão armazenadas as imagens das opções de cores das pulseiras.
- **opcoesCores[corSelecionada].nomePastaImagens**: Acessa a lista de opções de cores (**opcoesCores**) e seleciona a pasta correspondente à cor selecionada (**corSelecionada**).

Construindo a Inteligência da página

- **"/imagem-"**: Adiciona uma parte fixa do caminho para indicar o prefixo dos nomes dos arquivos de imagem.
- **imagemSelecionada**: Representa o número da imagem selecionada, que pode variar de acordo com a escolha do usuário.
- **".jpeg"**: Indica a extensão dos arquivos de imagem, neste caso, JPEG.

Juntando todas essas partes, a instrução monta o caminho completo para a imagem que será exibida com base na cor e na imagem selecionada pelo usuário. Isso garante que a imagem exibida seja a correta de acordo com as escolhas feitas na interface.



Construindo a Inteligência da página

Agora, vamos criar a última funcionalidade do nosso projeto: ao selecionar uma cor, as imagens do Apple Watch devem ser modificadas de acordo com a cor escolhida.

O próximo passo é manipular o elemento da opção de cores e adicionar a propriedade **onclick** com o valor da função que implementaremos, **atualizarCorSelecionada()**.

```
<div id="opcoes">
  <h2 id="nome-cor-selecionada">Cor - Azul-inverno</h2>
  <ul id="selecao-cores">
    <li><input type="radio" name="opcao-cor" id="0-cor" onclick="atualizarCorSelecionada()">
    <li><input type="radio" name="opcao-cor" id="1-cor" onclick="atualizarCorSelecionada()">
    <li><input type="radio" name="opcao-cor" id="2-cor" onclick="atualizarCorSelecionada()">
    <li><input type="radio" name="opcao-cor" id="3-cor" onclick="atualizarCorSelecionada()">
    <li><input type="radio" name="opcao-cor" id="4-cor" onclick="atualizarCorSelecionada()">
  </ul>
```

E vamos armazenar dentro de uma variável o elemento para manipular:

```
const tituloProduto = document.getElementById("titulo-produto");
const imagemVisualizacao = document.getElementById("imagem-visualizacao");
const nomeCorSelecionada = document.getElementById("nome-cor-selecionada");
```


Construindo a Inteligência da página

Além dos elementos que armazenamos e manipulamos, precisamos também dos elementos das imagens em miniatura, pois elas também devem ser modificadas conforme a cor selecionada pelo usuário.



```
4  const opcaoImagem0 = document.getElementById("0-imagem-miniatura");
5  const opcaoImagem1 = document.getElementById("1-imagem-miniatura");
6  const opcaoImagem2 = document.getElementById("2-imagem-miniatura");
```

Vamos criar uma função semelhante à **atualizarTamanho e atualizarImagemSelecionada**, mas selecionando a propriedade **name** que contém o valor do grupo das cores, chamado "opcao-cor". Atualizaremos a variável de controle com o elemento armazenado em uma nova variável criada dentro da função, chamada **numeroCorSelecionada**.

```
function atualizarCorSelecionada() {
  const numeroCorSelecionada = document
    .querySelector('[name="opcao-cor"]:checked')
    .id.charAt(0);
  corSelecionada = numeroCorSelecionada;
}
```

Construindo a Inteligência da página

Resumindo, a função, **atualizarCorSelecionada()**, obtém o número da cor selecionada pelo usuário. Ela localiza o elemento que está marcado como selecionado (checked) com o atributo **name** igual a "opcao-cor" no documento HTML e extrai o primeiro caractere do seu ID. Esse número é então armazenado na variável **corSelecionada**.

O próximo passo é atualizarmos o título da página, e para isso devemos realizar uma modificação na função **atualizarTamanho()**.

```
function atualizarCorSelecionada() {
  const numeroCorSelecionada = document
    .querySelector('[name="opcao-cor"]:checked')
    .id.charAt(0);
  corSelecionada = numeroCorSelecionada;

  tituloProduto.innerText =
    "Pulseira loop esportiva " +
    opcoesCores[corSelecionada].nome +
    " para caixa de " +
    opcoesTamanho[tamanhoSelecionado];
}
```

```
function atualizarTamanho() {
  const opcaoTamanhoSelecionado = document
    .querySelector('[name="opcao-tamanho"]:checked')
    .id.charAt(0);
  tamanhoSelecionado = opcaoTamanhoSelecionado;
  tituloProduto.innerText =
    "Pulseira loop esportiva " +
    opcoesCores[corSelecionada].nome +
    " para caixa de " +
    opcoesTamanho[tamanhoSelecionado];

  if (opcoesTamanho[tamanhoSelecionado] === "41 mm") {
    imagemVisualizacao.classList.add("caixa-pequena");
  } else {
    imagemVisualizacao.classList.remove("caixa-pequena");
  }
}
```

Construindo a Inteligência da página

Vamos explicar essa parte específica:

```
function atualizarCorSelecionada() {  
  const numeroCorSelecionada = document  
    .querySelector('[name="opcao-cor"]:checked')  
    .id.charAt(0);  
  corSelecionada = numeroCorSelecionada;  
  
  tituloProduto.innerText =  
    "Pulseira loop esportiva " +  
    opcoesCores[corSelecionada].nome +  
    " para caixa de " +  
    opcoesTamanho[tamanhoSelecionado];  
}
```


- **opcoesCores[corSelecionada].nome**: Aqui, **opcoesCores** é um array que contém objetos representando diferentes opções de cores. **corSelecionada** é o índice usado para selecionar a cor específica dentro desse array. **nome** é uma propriedade de cada objeto dentro do array **opcoesCores**, que armazena o nome da cor correspondente ao índice **corSelecionada**.

Dessa forma, **opcoesCores[corSelecionada].nome** acessa o nome da cor selecionada pelo usuário dentro do array **opcoesCores** e o concatena com outras partes da string para formar o texto completo que será exibido no elemento **tituloProduto.innerText**.

Construindo a Inteligência da página

Outra alteração é que vamos deixar o nome das cores em letras minúsculas, e para isso vamos utilizar o método **toLowerCase()**, que é usado para converter uma string em letras minúsculas. E vamos alterar ambas as funções **atualizarCorSelecionada()** e **atualizarTamanho()**.

```
function atualizarCorSelecionada() {  
  const numeroCorSelecionada = document  
    .querySelector('[name="opcao-cor"]:checked')  
    .id.charAt(0);  
  corSelecionada = numeroCorSelecionada;  
  
  tituloProduto.innerText =  
    "Pulseira loop esportiva " +  
    opcoesCores[corSelecionada].nome.toLowerCase() +  
    " para caixa de " +  
    opcoesTamanho[tamanhoSelecionado];  
}
```



Construindo a Inteligência da página

Agora precisamos alterar as imagens miniaturas para a cor selecionada, dentro da função vamos adicionar a seguinte estrutura:

```
opcaoImagem0.src =  
    "./imagens/opcoes-cores/" +  
    opcoesCores[corSelecionada].nomePastaImagens +  
    "/imagem-0.jpeg";  
  
opcaoImagem1.src =  
    "./imagens/opcoes-cores/" +  
    opcoesCores[corSelecionada].nomePastaImagens +  
    "/imagem-1.jpeg";  
  
opcaoImagem2.src =  
    "./imagens/opcoes-cores/" +  
    opcoesCores[corSelecionada].nomePastaImagens +  
    "/imagem-2.jpeg";
```

- **opcaoImagem0.src**: A propriedade **src** especifica o caminho da imagem que será exibida.
- **"./imagens/opcoes-cores/"**: Indica o diretório onde as imagens estão armazenadas.
- **opcoesCores[corSelecionada].nomePastaImagens**: Acessa a lista de opções de cores e seleciona a pasta correspondente à cor escolhida. **nomePastaImagens** é uma propriedade que contém o nome da pasta onde estão localizadas as imagens dessa cor.
- **"/imagem-0.jpeg"**: Completa o caminho da imagem com o nome do arquivo. Nesse caso, é a imagem número 0, que representa uma das opções do Apple Watch para a cor selecionada.

O código atualiza as fontes das imagens em miniatura para exibir as opções disponíveis do Apple Watch com base na cor selecionada. Ele faz isso alterando o número da imagem e a variável associada, mas a lógica é a mesma para todas as instruções, garantindo que as imagens sejam carregadas corretamente conforme a seleção da cor.

Construindo a Inteligência da página

Vamos adicionar a lógica já implementada na função **atualizarImagemSelecionada()**, que altere a imagem maior para a cor selecionada:

```
imagemVisualizacao.src =  
    "./imagens/opcoes-cores/" +  
    opcoesCores[corSelecionada].nomePastaImagens +  
    "/imagem-" +  
    imagemSelecionada +  
    ".jpeg";
```

E por fim alterar o subtítulo do nosso produto, que mostra o nome da cor selecionada:

```
nomeCorSelecionada.innerText = "Cor - " + opcoesCores[corSelecionada].nome;
```

Essa linha de código atualiza dinamicamente o texto do elemento **nomeCorSelecionada** para exibir a cor selecionada pelo usuário.

- **nomeCorSelecionada.innerText**: A propriedade **innerText** define o conteúdo de texto dentro do elemento **nomeCorSelecionada**.
- **"Cor - " + opcoesCores[corSelecionada].nome**: Concatena a string "Cor - " com o nome da cor selecionada pelo usuário, acessando o array **opcoesCores** na posição **corSelecionada** e recuperando o nome da cor a partir da propriedade **nome**.

Construindo a Inteligência da página

A estrutura final da nossa função **atualizarCorSelecionada()** :

```
function atualizarCorSelecionada() {  
  const numeroCorSelecionada = document.querySelector('[name="opcao-cor"]:checked').id.charAt(0);  
  corSelecionada = numeroCorSelecionada;  
  
  tituloProduto.innerText = "Pulseira loop esportiva " + opcoesCores[corSelecionada].nome.toLocaleLowerCase() + " para caixa de " + opcoesTamanho[tamanhoSelecionado];  
  
  opcaoImagem0.src = "./imagens/opcoes-cores/" + opcoesCores[corSelecionada].nomePastaImagens + "/imagem-0.jpeg";  
  
  opcaoImagem1.src = "./imagens/opcoes-cores/" + opcoesCores[corSelecionada].nomePastaImagens + "/imagem-1.jpeg";  
  
  opcaoImagem2.src = "./imagens/opcoes-cores/" + opcoesCores[corSelecionada].nomePastaImagens + "/imagem-2.jpeg";  
  
  imagemVisualizacao.src = "./imagens/opcoes-cores/" + opcoesCores[corSelecionada].nomePastaImagens + "/imagem-" + imagemSelecionada + ".jpeg";  
  
  nomeCorSelecionada.innerText = "Cor - " + opcoesCores[corSelecionada].nome;  
}
```

Com isso o seu projeto está com todas as funcionalidades aplicadas e funcionando!

Parte 7

Ajustes Opcionais

Ajustes Opcionais

String template, também conhecida como template literals em JavaScript, é uma forma de criar strings que permitem a incorporação de expressões e variáveis de forma mais legível e conveniente.

Em JavaScript, os string templates são definidos usando crases (``) em vez de aspas simples (") ou aspas duplas (""). Dentro dos string templates, podemos incluir variáveis, expressões e até mesmo outras strings, utilizando a sintaxe `${}` .

Os string templates oferecem uma maneira mais intuitiva e legível de concatenar strings e variáveis em comparação com a concatenação tradicional usando o operador `+`. Além disso, eles suportam strings de várias linhas, o que os torna úteis para criar templates de texto mais complexos.

Agora que você compreende o conceito, podemos atualizar as concatenações de strings em nosso projeto. Abaixo estão as imagens das funções já com o string template implementado, mantendo o mesmo formato das informações criadas durante o desenvolvimento do projeto.

Ajustes Opcionais

```
function atualizarCorSelecionada() {
  const numeroCorSelecionada = document
    .querySelector('[name="opcao-cor"]:checked')
    .id.charAt(0);
  corSelecionada = numeroCorSelecionada;
  nomeCorSelecionada.innerText = `Cor - ${opcoesCores[corSelecionada].nome}`;
  imagemVisualizacao.src = `./imagens/opcoes-cores/${opcoesCores[corSelecionada].nomePastaImagens}/imagem-${imagemSelecionada}.jpeg`;
  opcaoImagem0.src = `./imagens/opcoes-cores/${opcoesCores[corSelecionada].nomePastaImagens}/imagem-0.jpeg`;
  opcaoImagem1.src = `./imagens/opcoes-cores/${opcoesCores[corSelecionada].nomePastaImagens}/imagem-1.jpeg`;
  opcaoImagem2.src = `./imagens/opcoes-cores/${opcoesCores[corSelecionada].nomePastaImagens}/imagem-2.jpeg`;

  tituloProduto.innerText = `Pulseira loop esportiva ${opcoesCores[
    corSelecionada
  ].nome.toLowerCase()} para caixa de ${opcoesTamanho[tamanhoSelecionado]}`;
}
```

Ajustes Opcionais

```
function atualizarTamanho() {
  const opcaoTamanhoSelecioneado = document
    .querySelector('[name="opcao-tamanho"]:checked')
    .id.charAt(0);
  tamanhoSelecioneado = opcaoTamanhoSelecioneado;
  if (opcoesTamanho[tamanhoSelecioneado] === "41 mm") {
    imagemVisualizacao.classList.add("caixa-pequena");
    tituloProduto.innerText = `Pulseira loop esportiva ${opcoesCores[
      corSelecioneada
    ].nome.toLowerCase()} para caixa de ${opcoesTamanho[tamanhoSelecioneado]}`;
    return;
  }
  imagemVisualizacao.classList.remove("caixa-pequena");
  tituloProduto.innerText = `Pulseira loop esportiva ${opcoesCores[
    corSelecioneada
  ].nome.toLowerCase()} para caixa de ${opcoesTamanho[tamanhoSelecioneado]}`;
}

function atualizarImagemSelecioneada() {
  const opcaoImagemSelecioneada = document
    .querySelector('[name="opcao-imagem"]:checked')
    .id.charAt(0);
  imagemSelecioneada = opcaoImagemSelecioneada;
  imagemVisualizacao.src = `./imagens/opcoes-cores/${opcoesCores[corSelecioneada].nomePastaImagens}/imagem-${imagemSelecioneada}.jpeg`;
}
```


Parte Bônus

Vídeos

Complementares

Vídeos Complementares



10 Erros Que Todo Dev Full Stack Iniciante Comete (e Como Evitar)

Hashtag Dev • 1K views • 2 months ago

[10 Erros Que Todo Dev Full Stack Iniciante Comete \(e Como Evitar\)](#)



Freela Full Stack de R\$1.000 - Dashboard com Login e Gráficos (HTML + JS + ...)

Hashtag Dev • 179 views • 1 month ago

[Freela Full Stack de R\\$1.000 - Dashboard com Login e Gráficos \(HTML + JS + Node\)](#)



Curso Completo de JavaScript: Adicionando Interatividade à Sua Landing Page

Hashtag Dev • 239 views • 1 month ago

[Curso Completo de JavaScript: Adicionando Interatividade à Sua Landing Page](#)

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



@hashtagprogramacao



youtube.com/@HashtagProgramacao

youtube.com/@DevHashtag

