



Programação Web

Introdução ao JavaScript



1. Conceitos Iniciais;
2. Ferramentas necessárias;
3. Tipos de JavaScript
4. Passo a passo para criar um JavaScript.

01. Conceitos Iniciais

JavaScript (Linguagem de Programação) é uma tecnologia que permite adicionar interatividade e dinamismo às páginas web de forma eficaz.

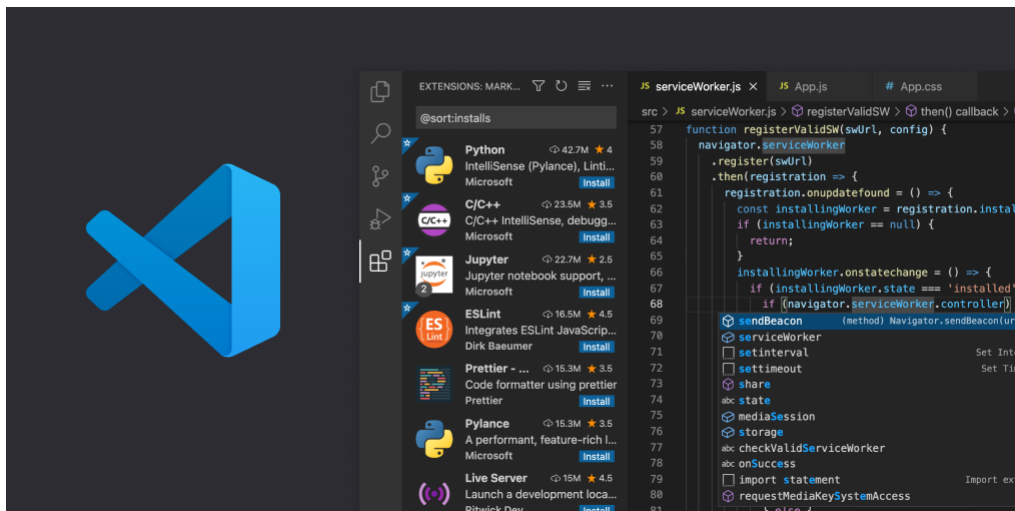
O JavaScript fornece uma ampla gama de funcionalidades, como **manipulação do DOM**, **controle de eventos** e **comunicação assíncrona com servidores**, permitindo que os desenvolvedores criem experiências de usuário envolventes e interativas. Com uma **sintaxe clara** e **fácil de aprender**, você pode escrever scripts para **validar formulários**, **criar animações**, e **realizar operações em tempo real**, tornando suas páginas mais interativas sem a necessidade de recarregar a página.

Ele compoe a **tríade** do front-end web (HTML, CSS e JS)



Ferramentas necessárias

Editor de texto (código): Visual Studio Code (Vs Code)



Navegador de Internet: Chorme (visualizar / interpretador)



Versionamento de código: GIT com GitLab



Outras ferramentas:

- **npm (Node Package Manager):** Gerenciador de Pacotes
- **Postman:** Para testar APIs.

Frameworks:

- **Front-end:** React, Angular, Vue.js.
- **Back-end:** Express.js (para Node.js), Koa, etc.
- **Ferramentas de Build:** Webpack, Parcel, Vite.



Tipos de JavaScript

Existem **4 formas** principais de incluir JavaScript em uma página web: **interno**, **inline**, **CDN** e **externo**. Vamos estudar cada um deles:

- **Interno:** Código dentro da tag <script> no próprio HTML.
- **Inline:** Código embutido em atributos de tags HTML.
- **Externo:** Código separado em arquivos .js vinculados ao HTML.
- **CDN:** Bibliotecas JavaScript hospedadas em redes de distribuição de conteúdo, como jQuery ou Bootstrap, carregadas de URLs públicas.

JS INTERNO

JavaScript Interno

O código JavaScript é colocado diretamente dentro da página HTML, dentro da tag **<script>**. É ideal para pequenos trechos de código que serão usados apenas naquela página específica.

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function sayHello() {
      console.log("Olá do JavaScript interno!");
    }
  </script>
</head>
<body>
  <button onclick="sayHello()">Evento de Clique</button>
</body>
</html>
```

JS INLINE

JavaScript Inline

O código JavaScript é inserido diretamente em **atributos** HTML, como onclick, onmouseover, etc. Embora fácil de usar para pequenas interações, não é recomendado em larga escala, pois dificulta a manutenção do código.

```
<!DOCTYPE html>
<html>
<body>
  <button onclick="alert('Olá do JavaScript em linha!')">Clique aqui.</button>
</body>
</html>
```

JS EXTERNO

JavaScript Externo

O código JavaScript é armazenado em um arquivo **separado** com a extensão **.js** e referenciado na página HTML usando a tag **<script>** com o atributo **src**. Este método é recomendado para grandes projetos, pois facilita a organização e a reutilização de código.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemplo JS Externo</title>
</head>
<style>
  #minha_div {
    width: 200px;
    height: 20px;
    background-color: lightblue;
    text-align: center;
  }
</style>
<body>
  <div id="minha_div">Passar o Mouse</div>

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

index.html

- Adicionamos um **CSS** apenas para facilitar a visualizar da área da div
- Acrescentamos uma **id** na div, para utilizar no seletor do **CSS** e do **JS**
- O script **JavaScript** é carregado **após** a renderização do HTML para garantir que os elementos estejam disponíveis no **DOM**, permitindo que o JavaScript os manipule corretamente. A **posição** do script é crucial; se carregado antes do HTML, pode interferir no seu funcionamento.

```
// Seleciona a div pelo ID
const minhaDiv = document.getElementById('minha_div');

// Adiciona o evento de mouseover
minhaDiv.addEventListener('mouseover', function() {
    minhaDiv.style.backgroundColor = 'yellow'; // Muda a cor de fundo quando o mouse
    passa sobre a div
});

// Adiciona o evento de mouseout para retornar à cor original
minhaDiv.addEventListener('mouseout', function() {
    minhaDiv.style.backgroundColor = 'lightblue'; // Retorna à cor original quando o
    mouse sai da div
});
```

script.js

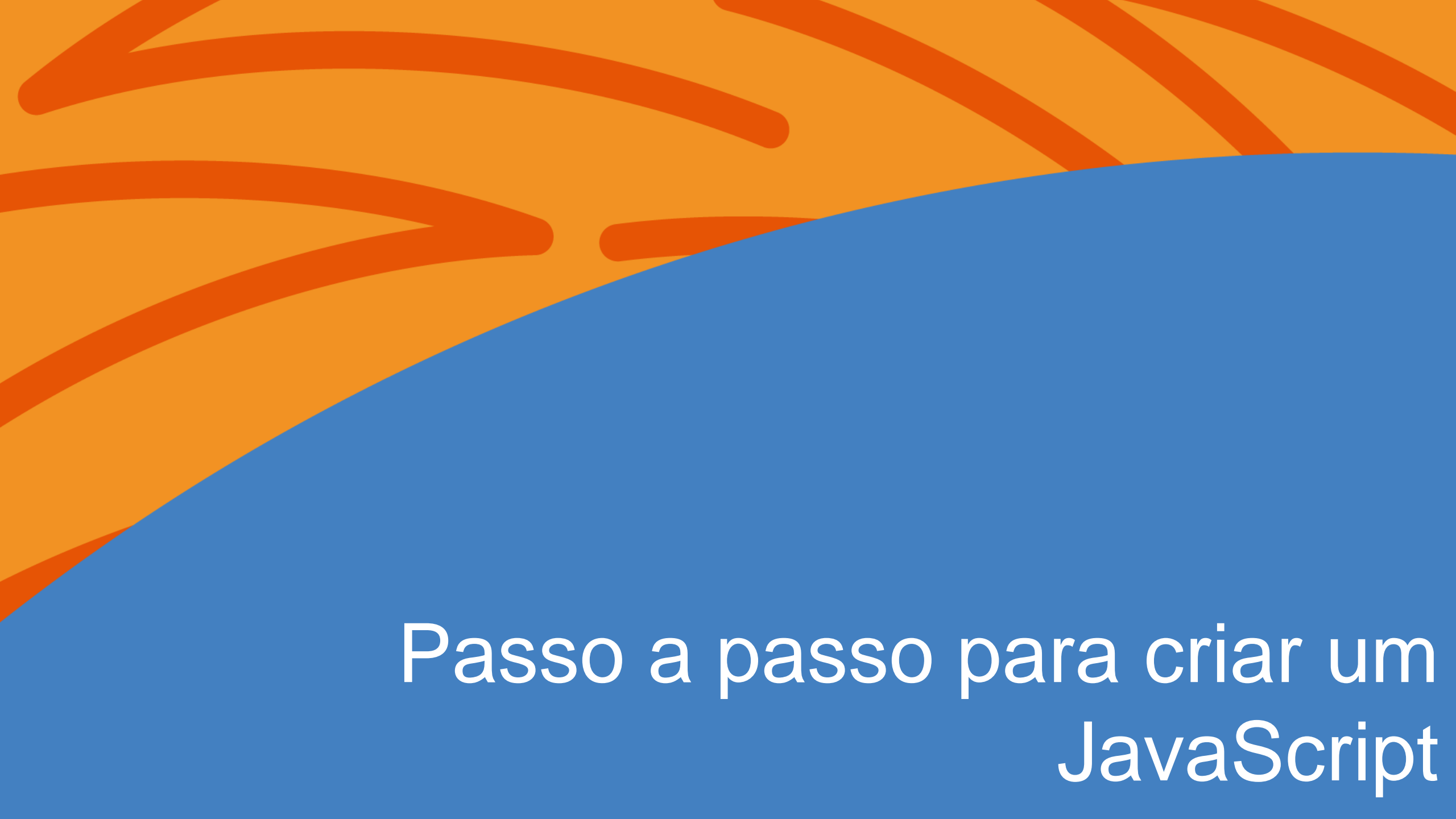
- **const minhaDiv:** Declara uma constante chamada minhaDiv.
- **document.getElementById('minha_div');** Seleciona o elemento HTML com o id "minha_div" e atribui à constante.
- **minhaDiv.addEventListener('mouseover', ...):** Adiciona um eventos à div para detectar quando o mouse sai dela.
- **minhaDiv.addEventListener('mouseover', ...):** Adiciona um eventos à div para detectar quando o mouse passa sobre ela.
- **function() {...}:** Define uma função que será executada quando o evento ocorrer.
- **minhaDiv.style.backgroundColor = 'yellow';:** Dentro da função, muda a cor de fundo da div para amarelo quando o mouse está sobre ela.

JS VIA CDN

Em vez de hospedar o código localmente, você pode incluir bibliotecas JavaScript de uma **CDN (Content Delivery Network)**. Isso permite o carregamento rápido de scripts populares, como jQuery, Bootstrap, etc., de servidores distribuídos ao redor do mundo.

```
<!DOCTYPE html>
<html>
<head>
  <!-- Incluindo jQuery via CDN -->
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <button id="myButton">Clique em mim</button>

  <script>
    // Usando jQuery carregado via CDN
    $("#myButton").click(function() {
      alert("Olá da JQuery carregada com CDN!");
    });
  </script>
</body>
</html>
```

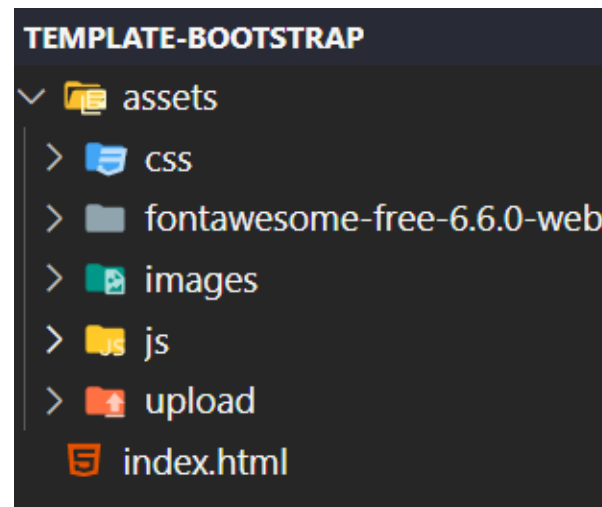



Passo a passo para criar um JavaScript

Agora que conhecemos os principais tipos de JavaScript, vamos apresentar um passo a passo para sua criação. Neste slide, vamos focar principalmente no **JavaScript externo**, pois ele oferece várias vantagens, incluindo:

- **Facilidade de Manutenção:** O código separado facilita a identificação e a correção de problemas.
- **Reutilização de Código:** Scripts externos podem ser utilizados em várias páginas, evitando a repetição.
- **Organização:** Ter o JavaScript em um arquivo separado torna o código HTML mais limpo e legível.

Nota: Utilizaremos a mesma organização que aplicamos nas aulas de **HTML** e **Bootstrap**.



```
<!DOCTYPE html>
<html lang="pt-br">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="assets/css/bootstrap.min.css">
</head>

<body>

  <script src="assets/js/bootstrap.bundle.min.js"></script>
</body>

</html>
```

```
<body>

  <button class="btn btn-primary" id="meu-botao">Clique aqui</button>

  <script src="assets/js/bootstrap.bundle.min.js"></script>
</body>
```

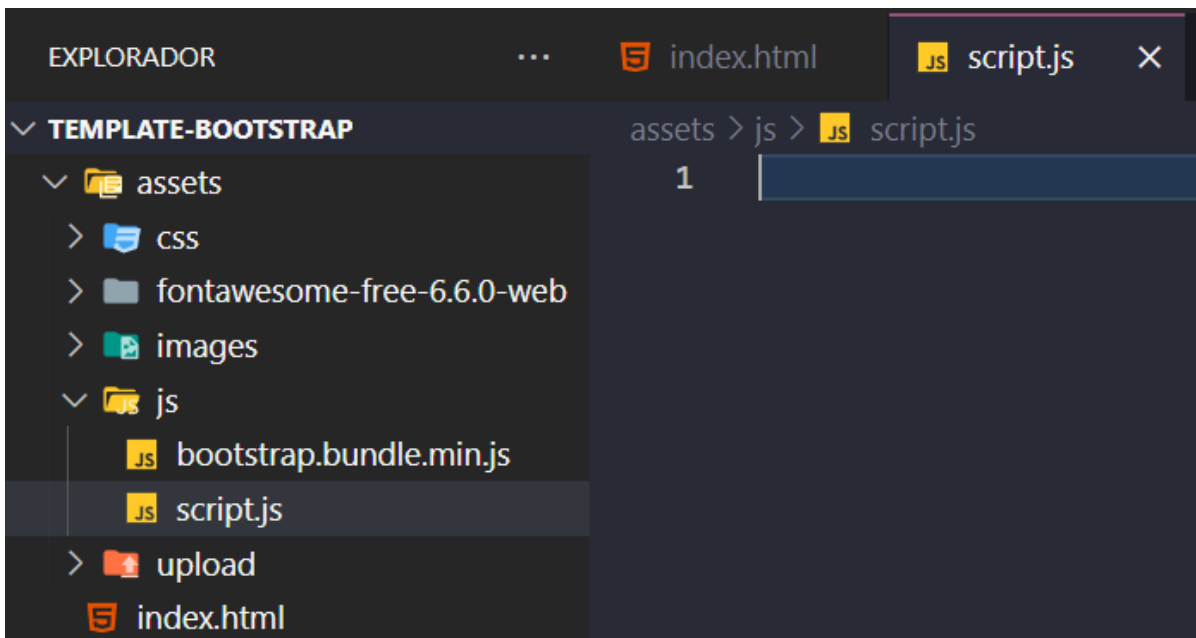
1- Criação ou reutilização do template

Crie a estrutura básica do nosso projeto. Você pode optar por seguir o nosso padrão do **Bootstrap** ou criar sua **própria estrutura**. Ao lado temos um exemplo de como fazer isso.

2- Elemento de teste

Iremos criar um botão para testar o JavaScript (JS) ao clicar nele.

- Usamos a classe do bootstrap apenas para estilizar
- A id será utilizada pelo JS para selecionar o elemento



3- Criar o JavaScript

Utilizando um dos métodos de criação de arquivo, crie um arquivo chamado **script.js**. Recomendado que salve ele dentro de “**assets/js**”, respeitando nossa organização.

```
<body>

  <button class="btn btn-primary" id="meu-botao">Clique aqui</button>

  <script src="assets/js/bootstrap.bundle.min.js"></script>
  <script src="assets/js/script.js"></script>
</body>

</html>
```

4- Incorporar o Script

Utilizando a tag `<script src>`, você pode inserir o arquivo externo script.js.

Nota: Certifique-se de respeitar o nome e o caminho onde o arquivo está salvo.

```
// Seleciona pela ID
const MEUBOTAO = document.getElementById('meu-botao');

// Adiciona o evento de clique
MEUBOTAO.addEventListener('click', function() {
    alert("Olá Mundo !")
});
```

5- Script de Teste

Nosso objetivo não é compreender todos os comandos do JavaScript neste momento, mas sim entender o passo a passo para inserir um arquivo JavaScript. Estudaremos os principais comandos em aulas futuras.

Por enquanto, observe que capturamos um elemento pela sua ID e armazenamos as informações desse elemento em uma constante chamada **MEUBOTAO**. O uso de letras maiúsculas para o nome da constante segue um padrão que adotaremos.

No segundo bloco de código, estamos adicionando um evento de clique ao botão, que chama uma função `alert()`. Essa função exibe uma mensagem no navegador quando o botão é clicado.



Siga o Senac em Minas nas Redes Sociais:

