Projeto de Extensão Beira-Linha: Iniciação à programação web

Monitores: Daniel Victor Rocha Costa, Fabia Rodrigues de Oliveira, Gabriel Ferreira Pereira

2/2025

💻 Capítulo 1: Introdução ao JavaScript

Este capítulo vai desvendar o que é o JavaScript e seu imenso potencial no mundo do desenvolvimento web.

O **JavaScript (JS)** é uma das linguagens de programação mais populares do mundo. Originalmente, ele foi criado para dar vida às páginas web, tornando-as **interativas**.

- Antes do JS, as páginas eram estáticas, como um panfleto.
- Com o JS, elas se tornaram dinâmicas, como um aplicativo.

O Potencial é Ilimitado:

1. **Desenvolvimento Web Completo:** Hoje, o JS é usado tanto no lado do cliente (navegador) quanto no lado do servidor (com Node.js), permitindo a criação de **aplicações web** complexas e completas.
2. **Desenvolvimento Mobile:** Com frameworks como React Native, você pode usar JS para criar aplicativos nativos para iOS e Android.
3. **Múltiplas Plataformas:** É usado para desenvolver jogos, softwares de desktop e até controlar robôs.

Objetivo	Descrição
O que é e para que serve o uso da linguagem programação	Programação é a arte de dar instruções a um computador. JS é a linguagem que usamos para dar essas instruções e criar interatividade no navegador.

História do JavaScript	Criado em 1995 por Brendan Eich, ele se tornou o padrão da internet para interatividade.

- **Front-end** é tudo que o usuário **vê e interage** diretamente no navegador (HTML, CSS e JS).
- **Back-end** é o que está por **trás da cena** (servidores, bancos de dados, lógica de negócio).

Onde Inserir JS (Ligar o HTML ao JS)

Existem três formas principais de adicionar código JavaScript a uma página HTML, mas a **mais recomendada** para organização e performance é a terceira:

1. **Inline (Dentro da Tag):** Para eventos muito simples.
   `<button onclick="alert('Clicado!')">Clique</button>`

2. **Interno (Na Tag `<script>` no HTML):** Para códigos curtos e específicos da página.

```
<script>

   console.log("Olá Mundo!");

</script>
```

3. **Externo (Arquivo .js):** O padrão da indústria. Você cria um arquivo separado (`script.js`) e o anexa ao HTML no final do body, ou seja, uma linha antes do fechamento `</body>`.

```
<!DOCTYPE html>

<html>

<head>

    <title>Minha Página</title>

</head>
```

```
<body>
    <h1>Aguardando JS...</h1>
    <script src="script.js"></script>
</body>
</html>
```

⬅️**inserção do js**

🧠 Capítulo 2: Fundamentos Essenciais do JavaScript

Para começar a interagir com o navegador, precisamos entender a lógica básica de qualquer linguagem de programação. Este capítulo é fundamental, mas vamos direto ao ponto.

Variáveis e Tipos de Dados

Variáveis são "caixas" que usamos para armazenar informações (dados) que podem mudar ao longo do tempo.

Palavra-chave	Uso	Exemplo
let	Variável que **pode** ter seu valor reatribuído. (Mais comum)	let nome = "Ana";
const	Variável que **não pode** ser reatribuída. (Para valores fixos)	const PI = 3.14;

Tipos de Dados Essenciais:

Tipo	O que Armazena	Exemplo
String	Textos (sempre entre aspas)	"Olar!", "123"

Number	Números (inteiros ou decimais)	10, 3.14
Boolean	Valores lógicos: true (verdadeiro) ou false (falso)	true

- Exemplos:

```
const idade = 30; // Number

let nomeUsuario = "Pedro"; // String

let estaLogado = true; // Boolean

console.log(nomeUsuario + " tem " + idade + " anos."); // Saída no console: Pedro tem 30 anos.
```

Operadores

Usamos operadores para realizar ações com os dados.

1. Operadores Aritméticos

Usados para realizar operações matemáticas.

- + (Adição)
- - (Subtração)
- * (Multiplicação)
- / (Divisão)
- % (Resto da Divisão - Módulo)

Exemplo: let resultado = 5 + 3; // resultado é 8
let resto = 10 % 3; // resto é 1

2. Operadores de Atribuição

Usados para atribuir valores a variáveis. O mais comum é o sinal de igual.

- = (Atribui o valor da direita à variável da esquerda)

- `+=` (Adição e Atribuição: `x += y` é o mesmo que `x = x + y`)
- `-=` (Subtração e Atribuição)

Exemplo:let x = 10;
x += 5; // x agora é 15

3. Operadores de Comparação

Usados para comparar dois valores e retornar um `Boolean` (`true` ou `false`).

- `==` (Igualdade **apenas de valor**. Ex: `10 == "10"` é `true`)
- `===` (Igualdade de **valor E tipo**. Mais seguro. Ex: `10 === "10"` é `false`)
- `!=` (Diferente em valor)
- `!==` (Diferente em valor ou tipo)
- `>` (Maior que)
- `<` (Menor que)
- `>=` (Maior ou igual a)
- `<=` (Menor ou igual a)

Exemplo:

```
let a = 10;
let b = "10";
console.log(a == b); // true (apenas valor)
console.log(a === b); // false (valor e tipo)
```

4. Operadores Lógicos

Usados para combinar expressões condicionais.

- `&&` (E lógico: Retorna `true` se **ambas** as condições forem verdadeiras)
- `||` (OU lógico: Retorna `true` se **pelo menos uma** das condições for verdadeira)

Exemplo:

```
let idade = 20;
let possuiCNH = true;

// Precisa ser maior de 18 E ter CNH
let podeDirigir = idade >= 18 && possuiCNH; // true
```

Estruturas Condicionais (Decisões)

As **condicionais** permitem que nosso código tome decisões com base em

se uma condição é verdadeira (`true`) ou falsa (`false`).

- Exemplo:

```
const media = 7.5;

 if (media >= 7) {

console.log("Parabéns, aprovado!");

 } else if (media >= 5) {

 console.log("Recuperação.");

 } else {

 console.log("Reprovado.");

 }
```

Estruturas de Repetição (Loops)

Os **loops** executam um bloco de código repetidas vezes, economizando tempo e esforço. O mais comum é o `for`.

- Exemplo:

```
// 1. Onde começa (i = 0)

// 2. Condição para continuar (i < 5)

// 3. O que acontece a cada volta (i++)

for (let i = 0; i < 5; i++) {

    console.log("Contagem: " + i);

    // Este código será executado 5 vezes (0, 1, 2, 3, 4)

}
```

Funções

As **Funções** são blocos de código que executam uma tarefa específica e podem ser reutilizados várias vezes. Elas são essenciais para manter o código organizado e evitar repetição.

Uma função pode **receber informações** (argumentos ou parâmetros) e **devolver um resultado** (retorno).

1. Como Declarar uma Função

Existem algumas formas de criar funções em JavaScript, mas a mais comum para iniciantes é a **Declaração de Função** e a **Função de Seta (Arrow Function)**, que é mais moderna e compacta.

Tipo de Função	Sintaxe	Uso
Declaração Comum	`function nome(parametro) { ... }`	O jeito tradicional e legível.
Função de Seta (Arrow Function)	`const nome = (parametro) => { ... }`	Sintaxe mais curta, ideal para funções simples e *callbacks* (como no `addEventListener`).

⚠️ ATENÇÃO !

Aos que não fazem as oficinas de Lógica de Programação, recomendamos fortemente que procurem conteúdos relacionados a matéria na Internet. Abaixo cliquem na imagem para acessar o curso ou procurem pelo nome no Youtube.

⭐ Minicursos introdutórios gratuitos no Youtube:

Curso de Javascript - Rafaella Ballerini

Curso GRÁTIS de Javascript e ECMAscript para inicantes - Curso em Vídeo

⭐ Curso introdutório na W3School:

https://www.w3schools.com/js/default.asp

🎨 Capítulo 3: Alterando nosso HTML com JavaScript (DOM)

O principal poder do JavaScript na Web é a capacidade de **alterar o HTML e o CSS** da página em tempo real.

O que é o DOM?

O **DOM (Document Object Model)** é como o navegador organiza todos os elementos do seu HTML em uma estrutura de árvore. O JavaScript usa essa "árvore" para acessar e modificar qualquer elemento (como um <h1>, um <button>

ou um).

Em resumo: O DOM é a ponte entre seu HTML e seu JavaScript.

Acessando Elementos: Os Seletores

Para alterar algo, precisamos primeiro encontrá-lo. Usamos **Seletores** para "agarrar" os elementos do HTML.

Método	**O que ele Busca**	**Uso**
document.getElementById("idNome")	Busca um elemento pela sua **ID** única.	**Mais rápido e direto.**
document.querySelector(".classeNome")	Busca o **primeiro** elemento que corresponda a um seletor CSS.	**Mais versátil.**
document.querySelectorAll(".classeNome")	Busca **TODOS** os elementos que correspondam a um seletor CSS.	**Retorna uma lista (NodeList).**

- Exemplo:

```javascript
// Exemplo de Seleção
const titulo = document.querySelector("h1"); // Seleciona o primeiro <h1>
const botaoID = document.getElementById("meuBotao"); // Seleciona o elemento com id="meuBotao"

// 1. Alterando o conteúdo de um elemento (Texto)
titulo.textContent = "Título Alterado por JS!";

// 2. Alterando o estilo de um elemento (CSS)
titulo.style.color = "blue";
titulo.style.fontSize = "3rem";
```

🧠 Eventos (Tornando a Página Viva)

Eventos são ações que acontecem na página (cliques, passagem do mouse, digitação, etc.). O JavaScript espera por eles para executar uma função.

Método	Descrição	Exemplo
onClick	Dispara ao **clicar** no elemento. (Geralmente usado diretamente no HTML ou com getElementById)	botao.onclick = function() { ... };
onChange	Dispara quando o valor de um formulário (input, select) é **alterado**.	input.onchange = function() { ... };
addEventListener	O **método moderno e mais recomendado**. Dispara quando uma certa condição é atendida (Ex: quando elemento for clicado)	botao.addEventListener("click", minhaFuncao);

- Exemplo:

```javascript
// Exemplo usando addEventListener (RECOMENDADO)
const meuBotao = document.getElementById("meuBotao");

meuBotao.addEventListener("click", function () {
    // Código que será executado SOMENTE quando o botão for clicado
    alert("Você clicou no botão!");
});
```

🚀 Capítulo 4: Colocando a Mão na Massa

Agora, vamos aplicar os conhecimentos de DOM e Eventos para adicionar interatividade aos projetos de **E-commerce** e **Portfólio** que vocês já criaram!
Os códigos em javascript devem ser adicionados no arquivo **script.js**

🛒 1. Alterações Possíveis no Projeto E-commerce

Essas ideias exigem pouca lógica, mas demonstram o poder do JS!

🔥 Alterar Texto e Estilos Usando DOM

1. **Mudar o texto do banner ao clicar no botão:**
    - **Conceito:** Seleciona um botão e, ao ser clicado, altera a propriedade `textContent` de outro elemento.

```javascript
document.querySelector(".banner button").addEventListener("click", function () {
  document.querySelector(".banner h2").textContent = "Veja nossas ofertas!";
});
```

2. **Mudar o fundo do card ao passar o mouse:**

- **Conceito:** Usa `querySelectorAll` para pegar vários cards e faz um *loop* (`forEach`) para adicionar dois *listeners*: `mouseenter` e `mouseleave`, que mudam a propriedade `style.background`.

```
cards.forEach(card => {
  card.addEventListener("mouseenter", () => card.style.background = "red");
  card.addEventListener("mouseleave", () => card.style.background = "aqua");
});
```

🔥 Criar um Sistema Simples de "Favoritar"

- **Conceito:** Usa `querySelectorAll` para adicionar um *listener* de `click` a vários botões de "Favoritar". Ao ser clicado, ele altera o próprio conteúdo do botão (`btn.textContent`).

```
const botoes = document.querySelectorAll(".card button:nth-child(5)");

botoes.forEach(btn => {
  btn.addEventListener("click", function () {
    btn.textContent = "Favoritado ❤️";
  });
});
```

🔥 Simular Carrinho de Compras na Tela

- **Conceito:** Cria uma variável (`let contador = 0;`) para armazenar o estado global. Cada clique em um botão de "Comprar" incrementa o contador e atualiza o `textContent` de um elemento na tela (`<p id="carrinho">`).

```
let contador = 0;
document.querySelectorAll(".card button:first-of-type").forEach(btn => {
  btn.addEventListener("click", () => {
    contador++;
    document.getElementById("carrinho").textContent = "Carrinho: " + contador;
  });
});
```

(Lembre-se de adicionar a tag `<p id="carrinho">Carrinho: 0</p>` no seu HTML!)

✅ 2. Alterações Possíveis no Projeto Portfólio

🔥 Trocar a Foto do Portfólio ao Clicar em um Botão

- **Conceito:** botão que altera foto de perfil.

```
document.getElementById("trocarFoto").onclick = function () {
  document.getElementById("fotoPerfil").src = "outra-foto.png";
};
```

(Lembre-se de adicionar um botão `<button id="trocarFoto" type="button">Mudar foto</button>` *no seu HTML!)*

🔥 Mudar o tema do site (Dark/Light Mode)

- **Conceito:** botão que alterna entre claro ↔ escuro.

```
const botao = document.getElementById("btn-dark-mode");

botao.addEventListener("click", () => {
    document.body.classList.toggle("dark-mode");
});
```

(Lembre-se de adicionar um botão `<button id="btn-dark-mode">Dark Mode</button>` *no seu HTML!)*

(E no seu arquivo style.css criar classe .dark-mode e selecionar as cores pro Dark Mode)

```
/* Tema escuro */
.dark-mode {
    background-color: #121212;
    color: #f1f1f1;
}

/* Para links no dark mode */
.dark-mode a {
    color: #f1f1f1;
}

/* Botão do dark mode */
#btn-dark-mode {
    padding: 8px 12px;
    border-radius: 8px;
    background-color: #333;
    color: white;
    border: none;
    cursor: pointer;
    margin-left: 16px;
}
```