



# Introdução a JavaScript



## que é JavaScript

JavaScript é uma linguagem de programação de alto nível, interpretada e orientada a objetos, amplamente utilizada no desenvolvimento de páginas web dinâmicas e interativas. Diferente do CSS e do HTML, os quais são linguagens de marcação, o JavaScript é uma linguagem de programação completa, que permite a criação de funcionalidades avançadas e interativas nas páginas web. Isso significa que, ao utilizar JavaScript, podemos adicionar comportamentos dinâmicos às páginas, como animações, transições, respostas a eventos de usuário, e atualização de conteúdo sem a necessidade de recarregar a página.

Por exemplo, quando um usuário clica em um botão ou passa o mouse sobre um elemento, o JavaScript pode ser utilizado para responder a essas ações de maneira imediata, proporcionando uma experiência mais fluida e interativa. Além disso, o JavaScript possibilita a atualização dinâmica do conteúdo de uma página sem que seja necessário recarregar a página inteira, eliminando a necessidade de loadings constantes e tornando a navegação mais agradável e eficiente.

Outra característica importante do JavaScript é sua capacidade de interagir com serviços web e APIs, permitindo que informações externas sejam integradas e exibidas em tempo real nas páginas web. Essa flexibilidade torna o JavaScript uma ferramenta essencial para desenvolvedores que desejam criar aplicações web modernas e interativas.

Ao longo desta aula, iremos explorar em mais detalhes os fundamentos do JavaScript, entender a importância de boas práticas no desenvolvimento com essa linguagem, e aprender como utilizá-la de maneira eficiente. Veremos também

como trabalhar com variáveis, que são um dos pilares das linguagens de programação, incluindo o JavaScript.



Espero que, ao final deste módulo, você tenha uma compreensão clara sobre o que é JavaScript e por que ele é uma peça fundamental no desenvolvimento de páginas web modernas.

## Fundamentos

Nesta segunda parte da nossa aula, falaremos sobre os fundamentos dessa linguagem, abordando conceitos essenciais cruciais para o desenvolvimento e para o uso eficiente do JavaScript. Esses fundamentos incluem variáveis, tipos de dados, operadores, estruturas de controle, funções, manipulação de arrays e objetos, e manipulação do DOM.

Vamos começar falando sobre variáveis e tipos de dados. Em qualquer linguagem de programação, as variáveis são fundamentais, pois armazenam os valores com os quais trabalhamos. No JavaScript, especificamente, temos um conceito conhecido como “tipagem fraca.” Isso significa que o JavaScript não exige que uma variável seja de um único tipo de dado. Por exemplo, você pode declarar uma variável inicialmente como texto e, mais tarde, atribuir um valor numérico a ela. Isso proporciona grande flexibilidade, mas também exige cuidado do programador para evitar erros inesperados.

Em seguida, temos os operadores, que são símbolos utilizados para realizar operações sobre os valores e controlar o fluxo do código. Em JavaScript, assim como em outras linguagens, temos operadores aritméticos (como `+`, `-`, `*`, `/`), operadores de comparação (como `>`, `<`, `==`) e operadores lógicos (como `&&`, `||`). Esses operadores são essenciais para manipular os dados e direcionar o comportamento do código.



As estruturas de controle também desempenham um papel fundamental no JavaScript. Elas permitem que o código siga diferentes caminhos com base em condições específicas. Por exemplo, você pode utilizar estruturas como “if”, “else”, “switch”, para decidir quais ações o código deve executar. Essas estruturas são vitais para a tomada de decisões durante a execução de um programa.


Outro ponto crucial são as funções. Em JavaScript, as funções são blocos de código que podem ser reutilizados em diferentes partes de um programa. Isso evita a repetição de código e torna a manutenção mais fácil. As funções no JavaScript são extremamente versáteis e podem ser manipuladas de diversas maneiras, tornando-as uma ferramenta poderosa no desenvolvimento.

Além das funções, a manipulação de arrays e objetos é outro aspecto importante. Arrays em JavaScript permitem armazenar múltiplos valores em uma única variável, facilitando o trabalho com listas de dados, como números, strings ou objetos. Por outro lado, os objetos são utilizados para armazenar coleções de dados relacionados, como as propriedades de um produto, que podem incluir código, descrição, preço e quantidade. Essa organização dos dados é fundamental para o desenvolvimento de aplicações mais complexas.

Por fim, temos a manipulação do DOM (Document Object Model). O DOM representa a estrutura de um documento HTML como uma árvore de nós, onde cada elemento é um nó que pode ser manipulado por meio do JavaScript. Isso permite que você adicione, remova ou edite elementos na página de forma dinâmica, criando uma interatividade direta com o usuário.

Compreender esses fundamentos é essencial para o desenvolvimento eficiente com JavaScript. Eles formam a base sobre a qual você poderá construir aplicações mais complexas e interativas.

## **Anotações**


Nesta terceira parte da nossa aula, exploraremos o conceito de anotações no código, que são extremamente importantes para a documentação e manutenção de programas. Anotações, ou comentários, são linhas de texto no código que não são interpretadas pelo JavaScript como comandos executáveis. Em vez disso, servem como uma ferramenta de comunicação dentro do código, explicando o que certas partes do código fazem, facilitando o entendimento por parte de outros desenvolvedores ou até mesmo pelo próprio autor, em revisões futuras. 

A importância das anotações no desenvolvimento de software não pode ser subestimada. Quando trabalhamos em equipes, é comum que múltiplos desenvolvedores trabalhem no mesmo projeto. Nesse contexto, comentários bem elaborados no código são essenciais para garantir que todos compreendam o funcionamento e o propósito das diferentes partes do código. Isso melhora a legibilidade e facilita a manutenção do código, especialmente em projetos de longo prazo, onde o código pode ser revisitado muito tempo depois de sua criação.

No JavaScript, as anotações podem ser feitas de duas formas principais: comentários em linha e comentários em bloco. Comentários em linha são iniciados com duas barras “//” e tudo que for escrito após essas barras, na mesma linha, é tratado como comentário. Este tipo de comentário é útil para anotações curtas e diretas, que explicam uma linha ou uma pequena parte do código.

Por outro lado, os comentários em bloco são utilizados para anotações mais longas, que podem abranger várias linhas. Eles são iniciados com “/” e finalizados com “/”. Essa forma de anotação é ideal para documentar funções, explicando seus parâmetros de entrada e saída, ou para fornecer descrições detalhadas de partes mais complexas do código.

Agora, vamos ver um exemplo prático. Imagine que estamos trabalhando em uma página HTML e queremos adicionar um código JavaScript. Após a tag `<body>`, inserimos a tag `<script>`, que informa ao navegador que o código JavaScript será escrito ali. Dentro dessa tag, podemos adicionar um comentário em linha usando “//”, por exemplo, para indicar que um log será

*enviado ao console: // Log do console. Isso deixa claro para qualquer pessoa que leia o código que o comando a seguir estará relacionado ao console  navegador.*

*Podemos também utilizar um comentário em bloco para documentar uma função ou uma seção maior do código. Esse comentário pode começar com /\* e ser expandido por várias linhas, até ser finalizado com \*/. Por exemplo:*

*/*

Este é um exemplo de comentário em bloco.

Ele pode ser usado para documentar funções

ou explicar o propósito de um bloco de código específico.

*\*/*

Esses comentários ajudam a manter o código organizado e compreensível, o que é crucial para o trabalho em equipe e a manutenção a longo prazo.

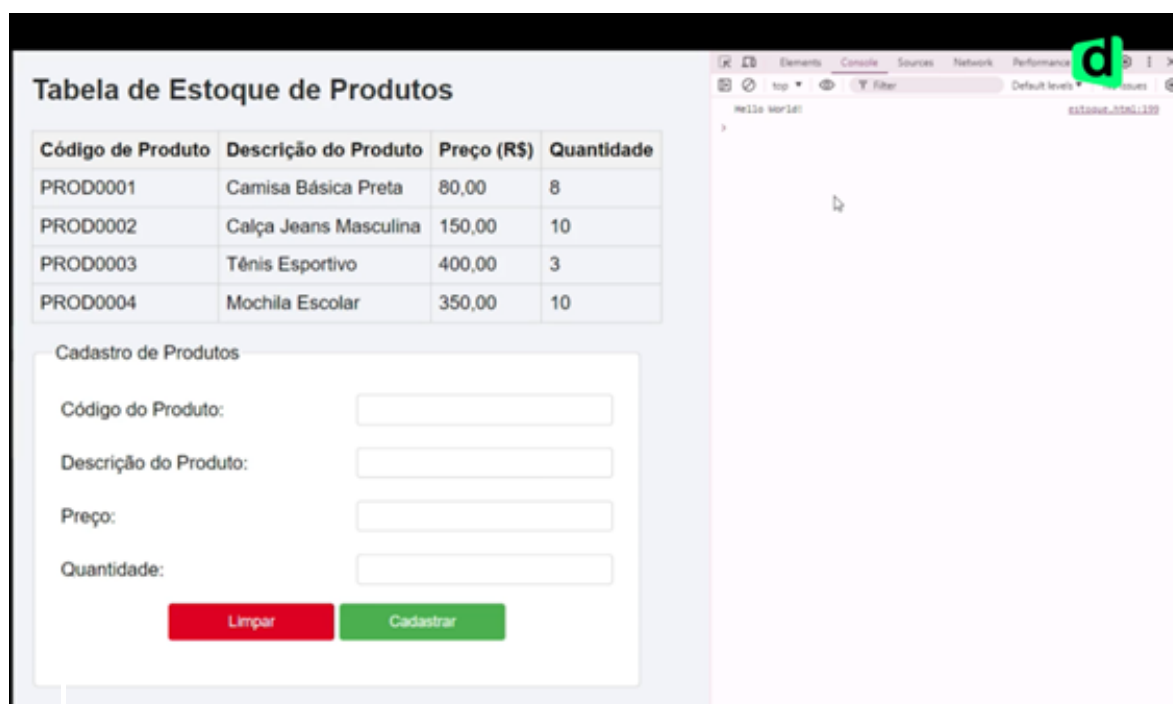
Para verificar a execução do código JavaScript e ver o resultado de um `console.log`, por exemplo, podemos usar a ferramenta “Inspecionar” do navegador. Ao clicar com o botão direito do mouse em qualquer parte da página e selecionar “Inspecionar”, podemos acessar o console do navegador, onde os logs e outros registros são exibidos. Isso é especialmente útil durante o desenvolvimento para depurar o código e entender como ele está funcionando em tempo real.

No exemplo que vimos na videoaula, quando adicionamos `console.log("Hello World");`, ao carregar a página, essa mensagem é exibida no console do navegador. Podemos acessar essa informação diretamente na aba “Console” dentro

da ferramenta de desenvolvedor. Relembre abaixo:



**Figura 1** - Visualização de um console.log




Fonte: elaborado pelo autor

Fonte: elaborado pelo autor

Em resumo, as anotações são uma prática fundamental no desenvolvimento com JavaScript, pois elas não apenas facilitam o entendimento e a manutenção do código, mas também são uma forma de comunicação eficaz entre os membros da equipe. Com isso, encerramos esta parte da aula, e espero que agora você tenha uma compreensão clara sobre a importância e o uso de anotações no JavaScript.

## Variáveis

Nesta última parte da nossa aula, vamos comentar sobre o conceito de variáveis, que desempenham um papel fundamental em qualquer linguagem de programação, incluindo o JavaScript. As variáveis são utilizadas para armazenar e manipular dados na memória durante a execução de um programa. Em JavaScript, as variáveis não interagem diretamente com a memória do computador, pois o código é executado dentro do navegador. Isso significa que as variáveis utilizam o ambiente do navegador para armazenar e processar informações. 

No JavaScript, existem três palavras reservadas principais que utilizamos para declarar variáveis: `var`, `let`, e `const`. Cada uma dessas palavras tem características específicas e é importante entender suas diferenças para utilizá-las de forma adequada no desenvolvimento de aplicações.

A primeira palavra reservada é `var`. Variáveis declaradas com `var` podem ter escopo global ou local. Isso significa que, se declaradas fora de uma função, elas estarão disponíveis em todo o código; se declaradas dentro de uma função, estarão disponíveis apenas dentro dessa função. Essa flexibilidade torna `var` uma escolha útil, mas também exige cuidado, pois pode levar a comportamentos inesperados se não for usada corretamente.

A segunda palavra reservada é `let`. Diferente de `var`, as variáveis declaradas com `let` têm escopo limitado ao bloco de código onde foram definidas. Um bloco de código pode ser, por exemplo, o interior de uma função ou de uma estrutura de controle, como um `if` ou `for`. Essa característica torna `let` ideal para situações em que você deseja garantir que uma variável seja utilizada apenas em uma parte específica do código, evitando possíveis conflitos ou redefinições acidentais.

Por fim, temos `const`, que, como o nome sugere, é utilizada para declarar variáveis cujo valor não deve mudar após a inicialização. Uma vez atribuído um valor a uma variável `const`, ele não pode ser alterado, o que pode prevenir erros, especialmente em situações onde a imutabilidade é desejada. No entanto, é importante lembrar que, se a variável `const` contiver um objeto ou array, as propriedades ou elementos

desse objeto, ou array podem ser modificados, embora a referência ao objeto em si não possa ser alterada.



Agora, vamos ver esses conceitos na prática com um exemplo simples. Suponha que estamos trabalhando em uma aplicação de estoque, onde precisamos manipular diferentes variáveis. Primeiro, declaramos uma variável utilizando `var`, como `var produto = "Monitor";`. Este comando cria uma variável chamada `produto` com o valor “Monitor”. Podemos então usar o comando `console.log("O meu produto é um " + produto);` para exibir uma mensagem concatenada com o valor da variável no console do navegador. Quando o código é executado, veremos a frase “O meu produto é um Monitor” no console, confirmando que a variável foi armazenada e utilizada corretamente.

Podemos também utilizar `let` para declarar uma variável com escopo limitado a um bloco de código específico. Veja o exemplo abaixo:

```
var a = "a";
```

```
function bloco() {
```

```
    let b = "b";
```

```
    console.log("O valor de a é: " + a);
```

```
    console.log("O valor de b é: " + b);
```

```
}
```

```
bloco();
```

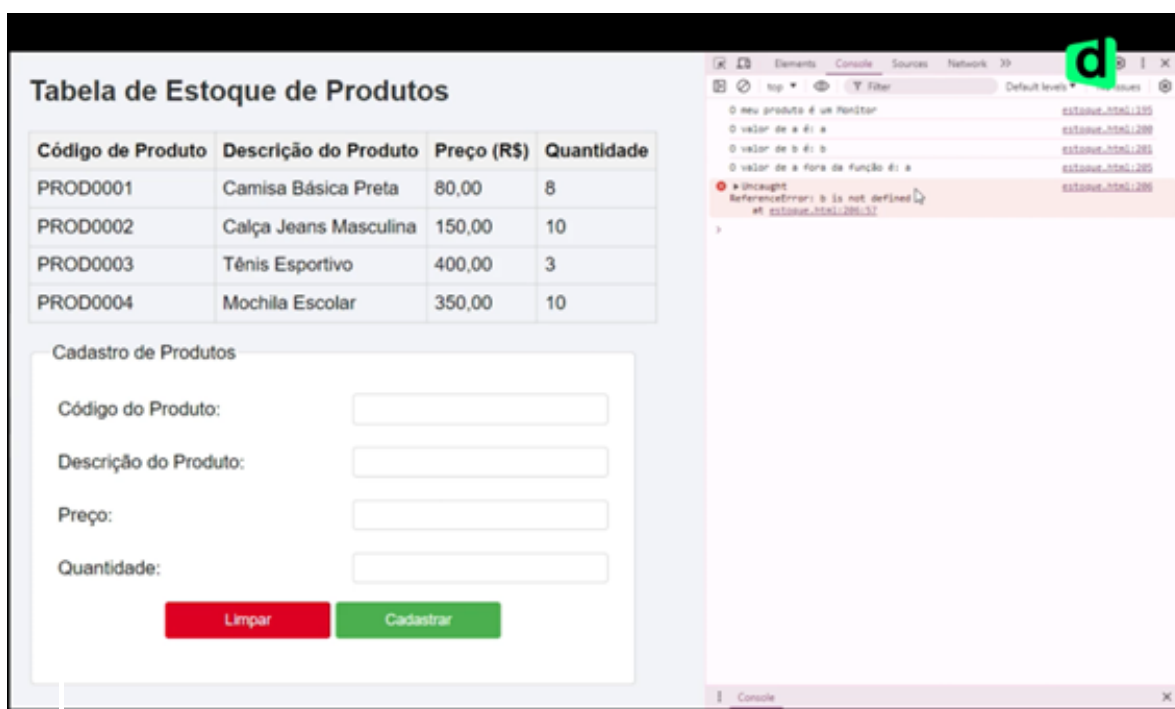


Neste código, a variável `b` é declarada utilizando `let` dentro da função `bloco()`. Isso significa que `b` está disponível apenas dentro desse bloco de código. Quando chamamos a função `bloco()`, o console exibirá “O valor de `a` é: `a`” e “O valor de `b` é: `b`”, mostrando que ambas as variáveis `a` e `b` estão acessíveis dentro da função. Entretanto, se tentarmos acessar `b` fora da função, como mostrado abaixo:

```
console.log("O valor de b fora da função é: " + b);
```

O JavaScript gerará um erro, indicando que a variável `b` não está definida naquele escopo fora da função `bloco()`. Isso ilustra a importância do escopo ao trabalhar com `let` em JavaScript. Veja o resultado:

**Figura 2** - Visualização do erro no console



Fonte: elaborado pelo autor



Com isso, encerramos nossa aula sobre o uso de variáveis e escopo em JavaScript. Discutimos como declarar variáveis usando `var` e `let`, como o escopo afeta seu comportamento e por que é importante entender essas diferenças para evitar erros e escrever código mais eficiente. Compreender esses conceitos é fundamental para o desenvolvimento em JavaScript, especialmente em projetos que exigem organização e manutenção de código. Espero que esta aula tenha sido esclarecedora e que você se sinta mais confiante ao utilizar variáveis em seus projetos. Até a próxima aula!

## Conteúdo Bônus

Para aprofundar os conhecimentos adquiridos na nossa aula, recomendo assistir ao vídeo **“Revisando Fundamentos do JavaScript (em menos de 30 minutos)”**. Esse vídeo, disponível no canal do Mayk Brito no YouTube, é ideal para quem deseja revisar ou reforçar rapidamente os conceitos fundamentais do JavaScript.

## Referência Bibliográfica

ASCENCIO, A. F. G.; CAMPOS, E. A. V. de. **Fundamentos da programação de computadores**. 3.ed. Pearson: 2012.

BRAGA, P. H. **Teste de software**. Pearson: 2016.

GALLOTTI, G. M. A. (Org.). **Arquitetura de software**. Pearson: 2017.



GALLOTTI, G. M. A. **Qualidade de software**. Pearson: 2015.

MEDEIROS, E. **Desenvolvendo software com UML 2.0 definitivo**. Pearson: 2004.

PFLEEGER, S. L. **Engenharia de software: teoria e prática**. 2.ed. Pearson: 2003.

SOMMERVILLE, I. **Engenharia de software**. 10.ed. Pearson: 2019.

## **Prática Integradora Desenvolvimento de Software**

### **Atividade Prática 7 - Explorando Fundamentos do JavaScript**

#### **Objetivos:**

- Compreender os conceitos básicos de JavaScript, incluindo variáveis, tipos de dados, operadores, e estruturas de controle.
- Aplicar os conceitos aprendidos para criar funcionalidades interativas simples em uma página web.
- Utilizar anotações para documentar e explicar o código JavaScript, facilitando a compreensão e manutenção.

#### **Materiais, Métodos e Ferramentas:**

- Editor de código (como Visual Studio Code, Sublime Text, ou qualquer outro editor de sua preferência)
- Navegador web (para testar o código e visualizar os resultados)

- Acesso à ferramenta de desenvolvedor do navegador (para verificar o console e depurar o código)



- Documentação online sobre JavaScript (para referência adicional, se necessário)

## Atividade Prática

Primeiramente, leia atentamente o texto a seguir:

Nesta atividade prática, você irá aplicar os conceitos básicos de JavaScript para criar uma página web interativa. O objetivo é utilizar variáveis, operadores, estruturas de controle e funções para implementar funcionalidades simples e documentar seu código com comentários apropriados.

Agora, vamos praticar!

## PASSO A PASSO DETALHADO DA ATIVIDADE

- **Criação da Página HTML e Inclusão do JavaScript**
  - Crie um novo arquivo HTML chamado index.html e adicione a estrutura básica de um documento HTML.
  - No corpo do documento HTML, adicione um botão e um parágrafo com um ID específico. O botão será usado para acionar uma função JavaScript, e o parágrafo servirá para exibir o resultado.

```
<!DOCTYPE html>
```

```
<html lang="pt-BR">
```

```
<head>
```

<meta charset="UTF-8">



<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Atividade JavaScript</title>

</head>

<body>

<button id="meuBotao">Clique aqui</button>

<p id="resultado"></p>

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

</body>

</html>

- **Criação do Arquivo JavaScript**

- Crie um novo arquivo JavaScript chamado script.js e inclua-o no documento HTML conforme o exemplo acima.
- No arquivo script.js, declare duas variáveis: uma variável de texto e uma variável numérica.
- Crie uma função chamada `exibirMensagem` que concatene o valor da variável de texto com o valor da variável numérica e atualize o conteúdo do parágrafo com o resultado.

// Declaração das variáveis

var texto = "O resultado é:";

```
let numero = 42;
```



```
// Função para exibir a mensagem
```

```
function exibirMensagem() {
```

```
    // Concatena as variáveis e atualiza o conteúdo do parágrafo
```

```
    document.getElementById("resultado").innerText = texto + " " + numero;
```

```
}
```

```
// Adiciona um evento ao botão para chamar a função quando clicado
```

```
document.getElementById("meuBotao").addEventListener("click",  
exibirMensagem);
```

- **Comentários e Documentação**

- Adicione comentários ao código JavaScript para explicar o propósito de cada parte do código. Utilize comentários em linha para explicações curtas e comentários em bloco para seções mais longas.

- **Testes e Validação**

- Abra o arquivo index.html em um navegador web e teste a funcionalidade do botão. Certifique-se de que, ao clicar no botão, o parágrafo é atualizado com a mensagem concatenada.
- Utilize a ferramenta de desenvolvedor do navegador para verificar o console e garantir que não há erros no código.

## **Gabarito Esperado**



- **Estrutura HTML:** O arquivo HTML deve conter um botão e um parágrafo com IDs específicos. O arquivo JavaScript deve estar vinculado corretamente e incluído no final do corpo do documento.
- **Código JavaScript:**
  - **Declaração de Variáveis:** Deve haver uma variável de texto (`var texto`) e uma variável numérica (`let numero`).
  - **Função `exibirMensagem`:** Deve concatenar o texto e o número e atualizar o conteúdo do parágrafo com o ID resultado.
  - **Comentários:** O código deve ser comentado adequadamente para explicar o propósito das variáveis, funções e ações realizadas.
- **Funcionamento:** O botão deve, ao ser clicado, chamar a função `exibirMensagem` e atualizar o parágrafo com a mensagem concatenada sem gerar erros no console.

**Ir para exercício**