Módulo 1: Fundamentos do JavaScript

Aula 01: Introdução ao JavaScript

1. Introdução ao JavaScript

JavaScript é uma das linguagens de programação mais populares do mundo. Criado em 1995 por **Brendan Eich** enquanto trabalhava na Netscape, inicialmente chamado de LiveScript, foi posteriormente renomeado para JavaScript. Seu principal objetivo era tornar as páginas da web mais dinâmicas e interativas. Com o passar do tempo, o JavaScript evoluiu significativamente, tornando-se essencial no desenvolvimento web. Hoje, é amplamente utilizado tanto no frontend (executado no navegador) quanto no backend (com Node.js).

1.1 Importância do JavaScript

- Presente em praticamente todos os sites modernos.
- Linguagem essencial para o desenvolvimento web.
- Suportado por todos os navegadores.
- Possui uma vasta comunidade e muitas bibliotecas/frameworks.

2. Configuração do Ambiente de Desenvolvimento

Para iniciar o desenvolvimento com JavaScript, precisamos configurar algumas ferramentas fundamentais.

2.1 Instalando o Visual Studio Code (VS Code)

O **VS Code** é um editor de código poderoso e leve, muito utilizado por desenvolvedores.

a. Acesse o site https://code.visualstudio.com/.
b. Baixe e instale a versão correspondente ao seu sistema operacional.
c. Abra o VS Code e familiarize-se com a interface.

2.2 Instalando o Node.js e npm

O **Node.js** permite executar JavaScript fora do navegador e vem acompanhado do **npm (Node Package Manager)**, que gerencia pacotes e bibliotecas.

1. Acesse o site https://nodejs.org/.
2. Baixe a versão LTS (Long Term Support) recomendada.
3. Instale o Node.js e verifique a instalação:
    - No terminal (ou prompt de comando), digite:
    - node -v
      Isso deve exibir a versão do Node.js instalada.

- Para verificar o npm:
- npm -v
  Isso deve exibir a versão do npm instalada.

3. Executando Código JavaScript

3.1 No Navegador

Podemos executar JavaScript diretamente no console do navegador.

1. Abra o Google Chrome (ou outro navegador moderno).
2. Pressione F12 ou Ctrl + Shift + I para abrir as ferramentas de desenvolvedor.
3. Acesse a aba **Console**.
4. Digite o seguinte código e pressione Enter:
5. console.log("Hello, JavaScript!");

   Isso deve exibir a mensagem "Hello, JavaScript!" no console.

3.2 No Node.js

Podemos executar JavaScript diretamente no terminal utilizando o Node.js.

1. No VS Code, crie um novo arquivo chamado **script.js**.
2. Adicione o seguinte código ao arquivo:
3. console.log("Executando JavaScript no Node.js!");
4. Salve o arquivo e execute no terminal com o comando:
5. node script.js

   Isso deve imprimir "Executando JavaScript no Node.js!" no terminal.

Aula 02: Variáveis e Tipos de Dados

4. Introdução às Variáveis

Variáveis são espaços na memória usados para armazenar valores que podem ser manipulados pelo código. No JavaScript, existem três formas principais de declarar variáveis:

- **var** (forma antiga, escopo global ou de função)
- **let** (escopo de bloco, pode ser reatribuído)
- **const** (escopo de bloco, não pode ser reatribuído)

```javascript
var nome = "João";  // Pode ser redeclarada e reatribuída
let idade = 25;      // Pode ser reatribuída, mas não redeclarada
const PI = 3.14159;  // Não pode ser reatribuída
```

4.1 Tipos Primitivos no JavaScript

Os tipos primitivos representam valores imutáveis e são os blocos fundamentais de dados em JavaScript.

- `string` → Texto entre aspas

```javascript
let nome = "Maria";
```

`number` → Números inteiros e decimais

```javascript
let preco = 19.99;
```

`boolean` → Verdadeiro (`true`) ou Falso (`false`)

```javascript
let ativo = true;
```

`undefined` → Variável declarada, mas sem valor atribuído

javascript

```javascript
let semValor;
```

`symbol` → Cria um identificador único

javascript

```javascript
let id = Symbol("id único");
```

`bigint` → Para números muito grandes

javascript

```javascript
let numeroGrande = 1234567890123456789012345678890n;
```

4.2OPERADORES ARITMÉTICOS E DE ATRIBUIÇÃO

Os operadores permitem manipular os valores armazenados nas variáveis.

Operadores Aritméticos

```
let a = 10;
let b = 5;

console.log(a + b);  // Soma: 15
console.log(a - b);  // Subtração: 5
console.log(a * b);  // Multiplicação: 50
console.log(a / b);  // Divisão: 2
console.log(a % b);  // Resto da divisão: 0
console.log(a ** b); // Exponenciação: 100000
```

Operadores de Atribuição

```
let x = 10;
x += 5;  // Equivalente a x = x + 5
x -= 2;  // Equivalente a x = x - 2
x *= 3;  // Equivalente a x = x * 3
x /= 2;  // Equivalente a x = x / 2
x %= 4;  // Equivalente a x = x % 4
x **= 2; // Equivalente a x = x ** 2
```

Exercícios Práticos

1. Declare três variáveis: uma usando var, outra let e outra const, e tente reatribuir seus valores.
2. Crie variáveis para cada tipo primitivo e exiba seus valores no console.
3. Use operadores aritméticos para calcular a média de três números.
4. Teste os operadores de atribuição para modificar o valor de uma variável.

Aula 03: Estruturas de Controle de Fluxo

Nesta aula, vamos explorar como controlar o fluxo de execução do código usando **condicionais** e **loops** no JavaScript.

5. CONDICIONAIS (IF, ELSE, SWITCH)

As condicionais são usadas para tomar decisões dentro do código, executando diferentes blocos dependendo de uma condição.

5.1ESTRUTURA IF E ELSE

O if executa um bloco de código se a condição for verdadeira. O else pode ser usado para tratar o caso contrário.

```javascript
let idade = 18;

if (idade >= 18) {
    console.log("Você é maior de idade.");
} else {
    console.log("Você é menor de idade.");
}
```

5.2ESTRUTURA IF, ELSE IF, ELSE

Caso existam várias condições, usamos else if.

```
let nota = 85;

if (nota >= 90) {
    console.log("Aprovado com excelência!");
} else if (nota >= 70) {
    console.log("Aprovado.");
} else {
    console.log("Reprovado.");
}
```

5.3ESTRUTURA SWITCH

O `switch` é útil quando há múltiplos casos a serem avaliados.

```
let dia = 3;

switch (dia) {
    case 1:
        console.log("Domingo");
        break;
    case 2:
        console.log("Segunda-feira");
        break;
    case 3:
        console.log("Terça-feira");
        break;
    default:
        console.log("Dia inválido.");
}
```

5.4Operadores Lógicos

Os operadores lógicos são usados para combinar condições.

Lógicos

Operadores	Significado
!	Negação (NOT)
&&	E (AND)
||	OU (OR)

Exemplo prático:

javascript

```javascript
let idade = 25;
let temCarteira = true;

if (idade >= 18 && temCarteira) {
    console.log("Pode dirigir.");
} else {
    console.log("Não pode dirigir.");
}
```

5.5Estruturas de Repetição

5.5.1 Estrutura for

O for é usado quando sabemos quantas vezes queremos repetir um bloco de código.

```
for (let i = 1; i <= 5; i++) {
    console.log("Número: " + i);
}
```

5.5.2 Estrutura while

O **while** repete um bloco enquanto uma condição for verdadeira.

```javascript
let contador = 1;

while (contador <= 5) {
    console.log("Contagem: " + contador);
    contador++;
}
```

5.5.3 Estrutura do while

A diferença do **do while** é que ele executa o bloco pelo menos uma vez, mesmo que a condição seja falsa.

```javascript
let numero = 0;

do {
    console.log("Número: " + numero);
    numero++;
} while (numero < 3);
```

Exercícios Práticos

1. **Condicional:** Crie um programa que verifica se um número é positivo, negativo ou zero.
2. **Operadores Lógicos:** Faça um código que verifica se uma pessoa pode votar (idade ≥ 16 **OU** é emancipada).
3. **Repetição:** Use um for para exibir os números de 1 a 10 no console.
4. **Desafio:** Crie um programa que recebe um número e usa um while para exibir a tabuada desse número.

Aula 04: Funções e Escopos

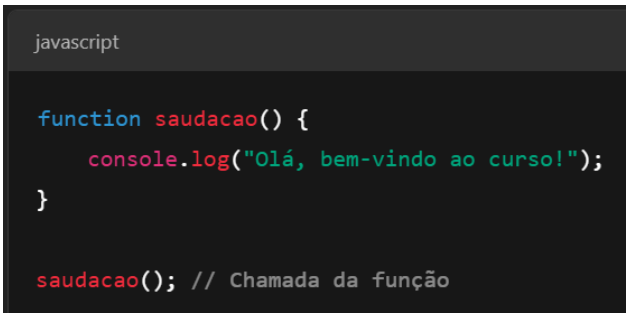
Nesta aula, vamos aprender sobre **funções**, que são blocos de código reutilizáveis, e **escopo de variáveis**, que determina onde as variáveis podem ser acessadas no código.

6. Declaração de Funções

Funções permitem organizar o código em blocos reutilizáveis, tornando-o mais modular e eficiente.

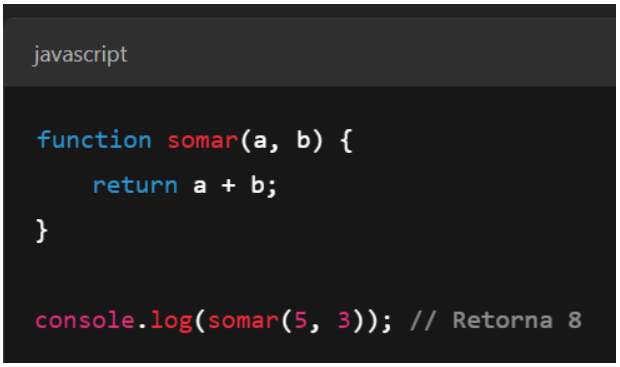
6.1Funções Tradicionais (function)

Uma função é definida usando a palavra-chave function, seguida por um nome, parênteses () e um bloco de código {}.



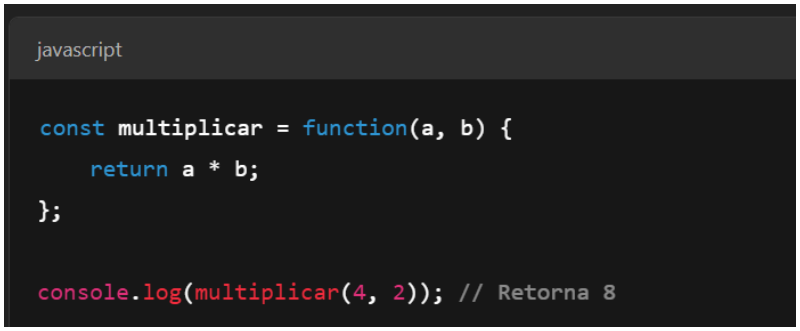
6.2Funções com Parâmetros

Os parâmetros permitem passar valores para dentro da função.



6.3Funções Anônimas

São funções que não possuem um nome e geralmente são atribuídas a uma variável.



6.4Arrow Functions (=>)

Arrow functions são uma forma mais curta e moderna de escrever funções.

```javascript
const dividir = (a, b) => a / b;

console.log(dividir(10, 2)); // Retorna 5
```

Se a função tiver apenas um parâmetro, os parênteses podem ser omitidos:

```javascript
const dobro = numero => numero * 2;

console.log(dobro(7)); // Retorna 14
```

Se o código tiver várias linhas, usamos {} e return:

```
const saudacaoPersonalizada = (nome) => {
    return `Olá, ${nome}!`;
};

console.log(saudacaoPersonalizada("Carlos")); // Retorna "Olá, Carlos!"
```

6.5 Escopo de Variáveis

O escopo define onde uma variável pode ser acessada dentro do código.

6.5.1 Escopo Global

Variáveis declaradas fora de qualquer função são **globais** e podem ser acessadas em qualquer parte do código.

```
let mensagem = "Olá, mundo!"; // Variável global

function exibirMensagem() {
    console.log(mensagem);
}

exibirMensagem(); // Funciona porque "mensagem" é global
```

6.5.2 Escopo Local (Função)

Variáveis declaradas dentro de uma função só podem ser acessadas dentro dessa função.

```
function exemploLocal() {
    let local = "Sou local!";
    console.log(local);
}

exemploLocal(); // Funciona
// console.log(local); // Erro! "local" não existe fora da função
```

6.5.3 Escopo de Bloco (let e const)

Com let e const, as variáveis têm escopo de **bloco**, ou seja, só existem dentro do {} onde foram criadas.

```
if (true) {
    let bloco = "Estou dentro de um bloco!";
    console.log(bloco);
}

// console.log(bloco); // Erro! "bloco" não existe fora do if
```

Se usarmos var, a variável **não** respeita o escopo de bloco:

```
if (true) {
    var teste = "Com var, eu saio do bloco!";
}

console.log(teste); // Funciona! Mas pode gerar problemas.
```

Exercícios Práticos

1. Crie uma função que recebe um nome como parâmetro e retorna "Olá, [nome]!".
2. Faça uma função que recebe dois números e retorna o maior deles.
3. Declare uma variável global e tente acessá-la dentro de uma função. Depois, crie uma variável dentro da função e tente acessá-la fora dela.
4. Escreva uma arrow function que recebe um número e retorna se ele é par ou ímpar.
5. Escreva uma função para descobrir o valor de Delta ($\Delta = b^2 4ac$).