Introdução ao JavaScript para Iniciantes

Um guia completo para dar as primeiras passos na programação com JavaScript.

1. Introdução ao JavaScript

JavaScript é uma linguagem de programação de alto nível, interpretada e multiparadigma. É uma das três tecnologias fundamentais da World Wide Web, junto com HTML e CSS. Originalmente criada para tornar as páginas web interativas, hoje o JavaScript é usado em uma vasta gama de aplicações, desde desenvolvimento web frontend e backend (com Node.js) até aplicativos móveis (com React Native) e jogos.

Para que serve?

- **Interatividade em páginas web:** Adicionar dinamismo a sites, como formulários interativos, galerias de imagens e animações.
- **Desenvolvimento *backend*:** Com Node.js, JavaScript pode ser usado para construir servidores e APIs.
- **Aplicativos móveis:** Frameworks como React Native permitem criar apps para *iOS* e *Android* usando JavaScript.
- **Jogos:** Desenvolvimento de jogos web e até mesmo jogos mais complexos.

Breve História

Criado em 1995 por Brendan Eich na Netscape, o JavaScript (inicialmente chamado LiveScript) foi desenvolvido para ser uma linguagem de *scripting* leve para navegadores. Rapidamente ganhou popularidade e foi padronizado como ECMAScript. Desde então, evoluiu significativamente, com novas versões e recursos sendo adicionados regularmente.

Onde o JavaScript é executado?

Principalmente em:

- **Navegadores *web*:** Onde ele manipula o DOM (Document Object Model) para criar interfaces de usuário dinâmicas.
- **Servidores:** Com ambientes de execução como Node.js, permitindo que JavaScript seja usado para lógica de *backend*.
- **Dispositivos móveis e desktops:** Através de *frameworks* específicos.

2. O Console do Navegador

O console do navegador é uma ferramenta fundamental para desenvolvedores JavaScript. Ele permite que você:

- Visualize mensagens de *log* (saída de informações do seu código).
- Teste pequenos trechos de código JavaScript em tempo real.
- Inspecione e manipule o DOM da página.
- Depure seu código, identificando erros e problemas.

Como abrir o console?

Na maioria dos navegadores modernos (Chrome, Firefox, Edge):

- Clique com o botão direito em qualquer lugar da página e selecione "Inspecionar" ou "Inspecionar Elemento".
- Vá para a aba "Console".

O comando `console.log()`

Este é o comando mais básico e frequentemente usado no console. Ele imprime o valor de uma variável ou uma mensagem no console.

```
console.log("Olá, mundo!"); // Imprime uma string
let nome = "Alice";
console.log(nome); // Imprime o valor da variável nome
let idade = 30;
console.log("Minha idade é:", idade); // Imprime múltiplos valores
```

Saída esperada no console:

```
Olá, mundo!
Alice
Minha idade é: 30
```

3. Comentários

Comentários são trechos de texto no seu código que são ignorados pelo interpretador JavaScript. Eles são usados para:

- Explicar o que o código faz.
- Tornar o código mais legível para você e para outros desenvolvedores.
- Desativar temporariamente partes do código para testes.

Comentários de uma linha (`//`)

Começam com duas barras e continuam até o final da linha.

```
// Este é um comentário de uma linha
let saudacao = "Olá!"; // Você pode colocar comentários no final da linha
```

Comentários de múltiplas linhas (`/* */`)

Começam com `/*` e terminam com `*/`. Podem abranger várias linhas.

```
/*
Este é um comentário
de múltiplas linhas.
Ele pode ser usado para blocos maiores de explicação.
*/
let numero = 10;
```

A importância de comentar o código

Um bom uso de comentários melhora a manutenibilidade do código e facilita a comunicação entre desenvolvedores.

4. Tipos de Dados

Em JavaScript, todo valor tem um tipo de dado. Os tipos de dados primitivos mais comuns são:

`string`

Representa texto. Strings são delimitadas por aspas simples (`' '`), aspas duplas (`" "`) ou *backticks* (`` ` ` ``).

```
let nomeCompleto = "João Silva";
let mensagem = 'Bem-vindo!';
let frase = `Olá, ${nomeCompleto}!`; // Exemplo de template literal
```

`number`

Representa números inteiros ou de ponto flutuante.

```
let idade = 25;
let preco = 99.99;
let pi = 3.14159;
```

`boolean`

Representa um valor lógico: `true` (verdadeiro) ou `false` (falso).

```
let isAtivo = true;
let temPermissao = false;
```

`null`

Representa a ausência intencional de qualquer valor de objeto. É um valor "vazio" ou "desconhecido" atribuído explicitamente.

```
let usuarioLogado = null; // Indica que nenhum usuário está logado
```

`undefined`

Representa uma variável que foi declarada, mas ainda não teve um valor atribuído. Também é o valor retornado por funções que não retornam explicitamente nada.

```
let sobrenome; // sobrenome é undefined
console.log(sobrenome); // Saída: undefined
```

`symbol` (breve menção)

Introduzido no ES6, `Symbol` é um tipo de dado primitivo cujos valores são únicos e imutáveis.
Usado principalmente para chaves de propriedades de objetos que se deseja que sejam únicas.

`object` (breve menção)

Um tipo de dado complexo que permite armazenar coleções de dados mais complexas e entidades mais complexas.
Arrays e funções são tipos de objetos.

5. Operadores Aritméticos

Operadores aritméticos são usados para realizar cálculos matemáticos.

- **Adição (`+`):** Soma dois números.

```
let resultadoSoma = 5 + 3; // 8
```

- **Subtração (`-`):** Subtrai um número do outro.

```
let resultadoSubtracao = 10 - 4; // 6
```

- **Multiplicação (`*`):** Multiplica dois números.

```
let resultadoMultiplicacao = 6 * 7; // 42
```

- **Divisão (`/`):** Divide um número por outro.

```
let resultadoDivisao = 20 / 5; // 4
```

- **Módulo (`%`):** Retorna o resto da divisão.

```
let resultadoModulo = 10 % 3; // 1 (10 dividido por 3 é 3 com resto 1)
```

- **Incremento (`++`):** Aumenta o valor de uma variável em 1.

```
let contador = 0;
contador++; // contador agora é 1
```

- **Decremento (`--`): Decreases the value of a variable by 1.**

```
let decrescer = 5;
decrescer--; // decrescer agora é 4
```

6. Concatenação de *Strings*

Concatenação é o processo de unir duas ou mais *strings* para formar uma nova *string*.

Usando o operador `+`

O operador `+` pode ser usado para concatenar *strings*.

```
let saudacao = "Olá, ";
let nome = "Mundo";
let mensagemCompleta = saudacao + nome + "!"; // "Olá, Mundo!"
```

Template Literals (``` ` ```) e Interpolação (`${}`)

Introduzidos no ES6, os *template literals* (delimitados por acentos graves) oferecem uma maneira mais flexível e legível de trabalhar com *strings*, incluindo a interpolação de variáveis.

```
let produto = "Caneta";
let preco = 2.50;
let detalhes = `O produto ${produto} custa R$ ${preco}.`;
console.log(detalhes); // Saída: O produto Caneta custa R$ 2.00.
```

7. Propriedades

Propriedades são valores associados a um objeto. Elas descrevem características ou estados do objeto. Em JavaScript, muitos tipos de dados, incluindo *strings*, se comportam como objetos e possuem propriedades.

A propriedade `.length` de *strings*

A propriedade `length` retorna o número de caracteres em uma *string*.

```
let texto = "JavaScript";
let tamanho = texto.length; // tamanho será 10
console.log(tamanho); // Saída: 10
```

Exemplos práticos:

```
let nomeUsuario = "Ana";
// Saída: O nome tem 3 caracteres.
console.log("O nome tem ".concat(nomeUsuario.length, " caracteres."));

let fraseLonga = "Aprender JavaScript é divertido!";
// Saída: A frase tem 31 caracteres.
console.log("A frase tem ".concat(fraseLonga.length, " caracteres."));
```

8. Métodos

Métodos são funções associadas a um objeto. Eles realizam ações ou operações com base nos dados do objeto. Você chama um método usando a sintaxe de ponto (`.`) após o nome do objeto, seguido pelo nome do método e parênteses `()`, que podem conter argumentos.

Métodos de *string*

- **`.toUpperCase()`:** Converte todos os caracteres de uma *string* para maiúsculas.

```
let saudacaoOriginal = "Olá Mundo";
let saudacaoMaiuscula = saudacaoOriginal.toUpperCase(); // "OLÁ MUNDO"
console.log(saudacaoMaiuscula); // Saída: OLÁ MUNDO
```

- **.toLowerCase():** Converte todos os caracteres de uma *string*.

```
let textoOriginal = "EXEMPLO";
let textoMinusculo = textoOriginal.toLowerCase(); // "exemplo"
console.log(textoMinusculo); // Saída: exemplo
```

- **`.trim()`:** Remove espaços em branco do início e do fim de uma *string*.

```
let textoComEspacos = "   Olá, JavaScript!   ";
let textoSemEspacos = textoComEspacos.trim(); // "Olá, JavaScript!"
console.log(textoSemEspacos); // Saída: Olá, JavaScript!
```

Métodos de *number*

Embora números sejam primitivos, JavaScript fornece objetos invólucro (*wrapper objects*) que permitem que eles tenham métodos.

- **`.toFixed()`:** Formata um número usando notação de ponto fixo.

```
let precoItem = 19.998;
// "20.00" (arredonda para 2 casas decimais)
let precoFormatado = precoItem.toFixed(2);
console.log(precoFormatado); // Saída: 20.00
```

- **`.toString()`:** Converte um número para sua representação em *string*.

```
let numeroInteiro = 123;
let numeroString = numeroInteiro.toString(); // "123"
console.log(typeof numeroString); // Saída: string
```

9. Revisão

Nesta aula, exploramos os fundamentos do JavaScript, desde sua história e onde ele é executado até conceitos práticos como o uso do console, comentários, tipos de dados, operadores aritmáticos, concatenação de *strings*, propriedades e métodos.

- O **Console** é seu melhor amigo para depurar e testar código.
- **Comentários** tornam seu código compreensível.
- Dominar os **Tipos de Dados** é crucial para manipular informações.
- **Operadores Aritméticos** permitem realizar cálculos.
- A **Concatenação de *Strings*** e os *Template Literals* são essenciais para trabalhar com texto.
- **Propriedades** descrevem características de objetos.
- **Métodos** permitem que objetos realizam ações.

Próximos passos:

Continue praticando! A melhor forma de aprender é. codificando. Experimente os conceitos aprendidos, crie pequenos projetos e explore a vasta documentação do JavaScript.