

Introdução ao Javascript

Variáveis: em JS

Em JavaScript, as variáveis são usadas para armazenar e manipular valores. Elas podem ser declaradas usando as palavras-chave **var**, **let** ou **const**. Aqui estão algumas informações sobre essas variáveis:

- 1 **var**: Antes do ES6 (ECMAScript 2015), a palavra-chave **var** era a forma mais comum de declarar variáveis em JavaScript. No entanto, o escopo das variáveis declaradas com **var** é diferente do escopo das variáveis declaradas com **let** e **const**. As variáveis declaradas com **var** têm escopo de função ou escopo global, o que significa que elas são acessíveis dentro da função em que são declaradas ou em todo o escopo global. Além disso, as variáveis declaradas com **var** sofrem hoisting, ou seja, são movidas para o topo do escopo em que foram declaradas durante a fase de compilação.

Exemplo de uso de **var**:

```
var x = 10;
function example() {
  var y = 20;
  console.log(x); // 10
  console.log(y); // 20
}
console.log(x); // 10
console.log(y); // ReferenceError: y is not defined
```

- 1 **let**: A palavra-chave **let** foi introduzida no ES6 e é usada para declarar variáveis com escopo de bloco. As variáveis declaradas com **let** são acessíveis apenas dentro do bloco em que são declaradas, seja ele um bloco de função, um bloco de instrução ou um bloco de loop. Ao contrário de **var**, as variáveis declaradas com **let** não são movidas para o topo do escopo e não sofrem hoisting.

Exemplo de uso de **let**:

```
let x = 10;
function example() {
  let y = 20;
  if (true) {
    let z = 30;
    console.log(x); // 10
    console.log(y); // 20
    console.log(z); // 30
  }
  console.log(x); // 10
  console.log(y); // 20
  console.log(z); // ReferenceError: z is not defined
}
console.log(x); // 10
console.log(y); // ReferenceError: y is not defined
console.log(z); // ReferenceError: z is not defined
```

- 1 **const**: A palavra-chave **const** também foi introduzida no ES6 e é usada para declarar constantes. Uma vez atribuído um valor a uma constante, esse valor não pode ser alterado posteriormente. As constantes têm escopo de bloco, assim como

as variáveis declaradas com **let**. Elas não sofrem hoisting e não podem ser reatribuídas.

Exemplo de uso de **const**:

```
const x = 10;
x = 20; // TypeError: Assignment to constant variable.

const y; // SyntaxError: Missing initializer in const declaration.

if (true) {
  const z = 30;
  console.log(x); // 10
  console.log(z); // 30
}
console.log(x); // 10
console.log(z); // ReferenceError: z is not defined
```

Em resumo, **var** tem escopo de função ou escopo global, **let** tem escopo de bloco e **const** também tem escopo de bloco, mas é usado para declarar constantes que não podem ser reatribuídas. Recomenda-se o uso de **let** e **const** em vez de **var** na maioria dos casos, pois isso ajuda a evitar problemas com escopo e hoisting.

Convenções de nomenclatura de variáveis.
Escopo de variáveis (global e local).

As convenções de nomenclatura de variáveis em JavaScript são amplamente adotadas para tornar o código mais legível e compreensível. Aqui estão algumas das convenções comuns:

- 1 **Camel Case:** É a prática de escrever palavras compostas ou frases onde as palavras são iniciadas com letras minúsculas, exceto a primeira palavra. Por exemplo: **nomeCompleto**, **idadeUsuario**, **saldoConta**.
- 2 **Snake Case:** É a prática de escrever palavras compostas ou frases onde as palavras são separadas por underscore (underline) "_". Por exemplo: **nome_completo**, **idade_usuario**, **saldo_conta**.
- 3 **Pascal Case:** É a prática de escrever palavras compostas ou frases onde todas as palavras são iniciadas com letras maiúsculas, sem espaços ou caracteres especiais. Por exemplo: **NomeCompleto**, **IdadeUsuario**, **SaldoConta**.
- 4 **Uppercase:** É a prática de escrever todas as letras em maiúsculas. Geralmente, é usada para constantes. Por exemplo: **TAXA_JUROS**, **VALOR_PI**.

Além disso, é importante seguir algumas regras e práticas ao nomear variáveis:

- Use nomes descritivos: Dê nomes que sejam significativos e representem o propósito ou conteúdo da variável. Isso torna o código mais legível e compreensível.
- Evite abreviações obscuras: Prefira nomes claros e descritivos em vez de abreviações que possam causar confusão para outras pessoas que leiam o código.
- Seja consistente: Mantenha uma convenção de nomenclatura consistente em todo o código para facilitar a leitura e a manutenção.

Agora, em relação ao escopo de variáveis:

- 1 **Escopo Global:** Variáveis declaradas fora de qualquer função têm escopo global e são acessíveis em todo o código. Elas podem ser acessadas e modificadas em qualquer parte do programa, incluindo dentro de funções.

Exemplo:

```
var globalVariable = 10;

function example() {
  console.log(globalVariable); // 10
}

console.log(globalVariable); // 10
```

- 1 **Escopo Local:** Variáveis declaradas dentro de uma função têm escopo local e são acessíveis apenas dentro dessa função. Elas não podem ser acessadas fora da função, a menos que sejam retornadas por meio de valores de retorno ou sejam utilizadas em closures.

Exemplo:

```
function example() {
  var localVariable = 20;
  console.log(localVariable); // 20
}

console.log(localVariable); // ReferenceError: localVariable is not defined
```

É importante observar que, a partir do ES6, as variáveis declaradas com **let** e **const** têm escopo de bloco, o que significa que elas são limitadas a um bloco de código específico (por exemplo, um bloco **if**, um loop **for**, um bloco de função) em vez de toda a função.

Tipos de dados em JS

JavaScript possui vários tipos de dados que podem ser usados para representar diferentes tipos de valores. Aqui estão os principais tipos de dados em JavaScript:

- 1 **Number:** O tipo **Number** representa valores numéricos, como inteiros e números de ponto flutuante.

Exemplos:

```
let age = 25;
let price = 9.99;
```

- 1 **String:** O tipo **String** representa sequências de caracteres. As strings devem ser colocadas entre aspas simples ("), aspas duplas (") ou crases (`).

Exemplos:

```
let name = 'John';
let message = "Hello, World!";
```

- 1 **Boolean:** O tipo **Boolean** representa um valor lógico que pode ser verdadeiro (**true**) ou falso (**false**).

Exemplos:

```
let isLogged = true;
let isActive = false;
```

- 1 **Null:** O tipo **Null** representa a ausência intencional de qualquer valor ou objeto.

Exemplo:

```
let data = null;
```

- 1 **Undefined:** O tipo **Undefined** é atribuído automaticamente a variáveis que foram declaradas, mas ainda não receberam um valor.

Exemplo:

```
let username;
console.log(username); // undefined
```

- 1 **Object:** O tipo **Object** é usado para armazenar coleções de pares de chave-valor. Os objetos em JavaScript são estruturas de dados complexas compostas por propriedades e métodos.

Exemplo:

```
let person = {
  name: 'John',
  age: 25,
  city: 'New York'
};
```

- 1 **Array:** O tipo **Array** é usado para armazenar uma coleção ordenada de valores. Os elementos em um array são acessados por meio de um índice numérico.

Exemplo:

```
let numbers = [1, 2, 3, 4, 5];
let fruits = ['apple', 'banana', 'orange'];
```

- 1 **Symbol:** O tipo **Symbol** é usado para criar valores únicos e imutáveis que podem ser usados como chaves de propriedades de objetos.

Exemplo:

```
let id = Symbol('id');
```

Além desses tipos de dados, JavaScript também oferece alguns tipos especiais, como **BigInt** para representar números inteiros arbitrariamente grandes e **Date** para representar datas e horários.

É importante observar que JavaScript é uma linguagem de tipagem dinâmica, o que significa que as variáveis podem mudar de tipo durante a execução do programa.

Operadores aritméticos:

JavaScript possui vários operadores aritméticos que permitem realizar operações matemáticas em valores numéricos. Aqui estão os principais operadores aritméticos em JavaScript:

1 Adição (+): Realiza a adição de dois valores.

Exemplo:

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

1 Subtração (-): Realiza a subtração de um valor pelo outro.

Exemplo:

```
let x = 5 - 3; // 2
```

1 Multiplicação (*): Realiza a multiplicação de dois valores.

Exemplo:

```
let x = 5 * 3; // 15
```

1 Divisão (/): Realiza a divisão de um valor pelo outro.

Exemplo:

```
let x = 6 / 3; // 2
```

1 Módulo (%): Retorna o resto da divisão de um valor pelo outro.

Exemplo:

```
let x = 7 % 3; // 1
```

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

Exemplo:

```
let x = 5;  
x++; // x agora é 6
```

1 Decremento (--): Diminui o valor de uma variável em 1.

Exemplo:

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

É importante lembrar que esses operadores podem ser combinados com atribuição para atualizar o valor de uma variável. Por exemplo:

```
let x = 5;  
x += 3; // x agora é 8 (equivalente a x = x + 3)  
x *= 2; // x agora é 16 (equivalente a x = x * 2)
```

Além dos operadores aritméticos básicos, JavaScript também possui operadores de incremento/decremento pré-fixados (++x, --x) e operadores de exponenciação (**) para cálculos com potências.

Operadores de atribuição:

* Operador de atribuição simples (=).

* Operadores de atribuição compostos: +=, -=, *=, /=.

* Operadores de atribuição de resto (%=), incremento (++) e decremento (--).

Os operadores de atribuição em JavaScript são usados para atribuir valores a variáveis. Aqui estão os principais operadores de atribuição:

1 Operador de atribuição simples (=): Atribui o valor à variável.

Exemplo:

```
let x = 5;
```

1 Operadores de atribuição compostos: São usados para realizar uma operação e atribuir o resultado à variável. Esses operadores combinam a operação com a atribuição.

● Operador de atribuição de adição (+=): Adiciona um valor à variável e atribui o resultado.

Exemplo:

```
let x = 5;  
x += 3; // x = x + 3 (x agora é 8)
```

● Operador de atribuição de subtração (-=): Subtrai um valor da variável e atribui o resultado.

Exemplo:

```
let x = 5;  
x -= 3; // x = x - 3 (x agora é 2)
```

- **Operador de atribuição de multiplicação (*=):** Multiplica a variável por um valor e atribui o resultado.

Exemplo:

```
let x = 5;  
x *= 3; // x = x * 3 (x agora é 15)
```

- **Operador de atribuição de divisão (/=):** Divide a variável por um valor e atribui o resultado.

Exemplo:

```
let x = 6;  
x /= 3; // x = x / 3 (x agora é 2)
```

- ❶ **Operadores de atribuição de resto (%=), incremento (++=) e decremento (--=):** Realizam a operação correspondente e atribuem o resultado à variável.

- **Operador de atribuição de resto (%=):** Calcula o resto da divisão da variável por um valor e atribui o resultado.

Exemplo:

```
let x = 7;  
x %= 3; // x = x % 3 (x agora é 1)
```

- **Operador de atribuição de incremento (++=):** Incrementa o valor da variável em 1 e atribui o resultado.

Exemplo:

```
let x = 5;  
x++; // x = x + 1 (x agora é 6)
```

- **Operador de atribuição de decremento (--=):** Decrementa o valor da variável em 1 e atribui o resultado.

Exemplo:

```
let x = 5;  
x--; // x = x - 1 (x agora é 4)
```

Esses operadores de atribuição são úteis para simplificar a escrita de código e combinar uma operação com a atribuição do resultado em uma única instrução.

Operadores de comparação:

* Igual (==) e estritamente igual (===).

* Diferente (`!=`) e estritamente diferente (`!==`).
* Maior (`>`), menor (`<`), maior ou igual (`>=`), menor ou igual (`<=`).

Os operadores de comparação em JavaScript são usados para comparar valores e retornar um resultado booleano (verdadeiro ou falso) com base na comparação. Aqui estão os principais operadores de comparação:

1 Operador igual (`==`): Verifica se dois valores são iguais, convertendo os operandos se necessário.

Exemplo:

```
let x = 5;  
let y = '5';  
console.log(x == y); // true (os valores são considerados iguais após conversão)
```

1 Operador estritamente igual (`===`): Verifica se dois valores são iguais, sem converter os operandos. Os valores e os tipos devem ser idênticos.

Exemplo:

```
let x = 5;  
let y = '5';  
console.log(x === y); // false (os tipos são diferentes)
```

1 Operador diferente (`!=`): Verifica se dois valores não são iguais, convertendo os operandos se necessário.

Exemplo:

```
let x = 5;  
let y = '5';  
console.log(x != y); // false (os valores são considerados iguais após conversão)
```

1 Operador estritamente diferente (`!==`): Verifica se dois valores não são iguais, sem converter os operandos. Os valores e os tipos devem ser diferentes.

Exemplo:

```
let x = 5;  
let y = '5';  
console.log(x !== y); // true (os tipos são diferentes)
```

1 Operador maior (`>`): Verifica se um valor é maior que outro.

Exemplo:

```
let x = 5;  
let y = 3;  
console.log(x > y); // true
```

1 Operador menor (`<`): Verifica se um valor é menor que outro.

Exemplo:

```
let x = 5;  
let y = 3;  
console.log(x < y); // false
```

1 Operador maior ou igual (>=): Verifica se um valor é maior ou igual a outro.

Exemplo:

```
let x = 5;  
let y = 3;  
console.log(x >= y); // true
```

1 Operador menor ou igual (<=): Verifica se um valor é menor ou igual a outro.

Exemplo:

```
let x = 5;  
let y = 3;  
console.log(x <= y); // false
```

Os operadores de comparação são amplamente usados em condições de controle de fluxo, como declarações `if`, `while`, `for`, entre outras, para executar diferentes blocos de código com base no resultado da comparação.

Operadores lógicos:

* E lógico ('&&').

* Ou lógico ('||').

* Negação lógica ('!').

Os operadores lógicos em JavaScript são usados para combinar ou inverter valores booleanos. Eles retornam um resultado booleano com base nas operações lógicas aplicadas aos operandos. Aqui estão os principais operadores lógicos:

1 Operador E lógico (&&): Retorna `true` se ambos os operandos forem `true`, caso contrário, retorna `false`.

Exemplo:

```
let x = 5;  
let y = 3;  
console.log(x > 0 && y > 0); // true (ambos os operandos são verdadeiros)  
console.log(x > 0 && y > 5); // false (um dos operandos é falso)
```

1 Operador OU lógico (||): Retorna `true` se pelo menos um dos operandos for `true`, caso contrário, retorna `false`.

Exemplo:

```
let x = 5;
let y = 3;
console.log(x > 0 || y > 0); // true (pelo menos um dos operandos é verdadeiro)
console.log(x < 0 || y > 5); // false (ambos os operandos são falsos)
```

- ❶ **Operador de Negação lógica (!):** Inverte o valor booleano do operando. Se o operando for **true**, retorna **false**. Se o operando for **false**, retorna **true**.

Exemplo:

```
let x = 5;
let y = 3;
console.log(!(x > 0)); // false (o operando é verdadeiro, a negação é falsa)
console.log(!(x < 0)); // true (o operando é falso, a negação é verdadeira)
```

Os operadores lógicos são frequentemente usados em expressões condicionais, como declarações **if** e loops, para avaliar múltiplas condições e controlar o fluxo do programa com base nos resultados das operações lógicas.

Operadores de concatenação:

* Concatenação de strings usando o operador de adição (+).

O operador de concatenação em JavaScript é usado para combinar strings em uma única string. O operador de adição (+) é utilizado para realizar a concatenação de strings. Aqui está um exemplo:

```
let firstName = "John";
let lastName = "Doe";

let fullName = firstName + " " + lastName;
console.log(fullName); // "John Doe"
```

Nesse exemplo, as variáveis **firstName** e **lastName** contêm strings. Usando o operador de adição (+), podemos concatenar essas strings e atribuir o resultado à variável **fullName**.

Além disso, também é possível concatenar strings com outros tipos de dados. Nesse caso, o JavaScript irá converter automaticamente os outros tipos de dados em strings antes de realizar a concatenação.

```
let number = 42;
let message = "The answer is: " + number;
console.log(message); // "The answer is: 42"
```

Nesse exemplo, o número **42** é convertido em uma string antes de ser concatenado com a mensagem.

É importante observar que o operador de adição (+) também pode ser usado para realizar a adição de números e para concatenar strings. O JavaScript executa a concatenação de strings quando encontra pelo menos um operando como uma string. Caso contrário, ele executa a adição numérica.

```
let x = 5;
let y = 10;
console.log(x + y); // 15

let str = "Hello" + " " + "World";
console.log(str); // "Hello World"
```

No segundo exemplo, como todos os operandos são strings, o JavaScript realiza a concatenação de strings.

Monte 10 exercícios com enunciado e respostas sobre o que foi visto até agora

Claro! Aqui estão 10 exercícios com enunciado e respostas sobre os conceitos discutidos até agora:

Exercício 1: Enunciado: Declare uma variável chamada **idade** e atribua a ela o valor 25. Em seguida, imprima o valor da variável no console. Resposta:

```
let idade = 25;
console.log(idade); // 25
```

Exercício 2: Enunciado: Declare duas variáveis, **x** e **y**, e atribua a elas os valores 10 e 5, respectivamente. Calcule e imprima a soma das duas variáveis. Resposta:

```
let x = 10;
let y = 5;
let soma = x + y;
console.log(soma); // 15
```

Exercício 3: Enunciado: Declare uma variável chamada **nome** e atribua a ela uma string com o seu nome. Em seguida, imprima uma mensagem de saudação no console utilizando o seu nome. Resposta:

```
let nome = "João";
console.log("Olá, " + nome + "! Bem-vindo!");
```

Exercício 4: Enunciado: Declare uma variável chamada **temperatura** e atribua a ela o valor 25. Verifique se a temperatura é maior que 30 e imprima no console o resultado como um valor booleano. Resposta:

```
let temperatura = 25;
let resultado = temperatura > 30;
console.log(resultado); // false
```

Exercício 5: Enunciado: Declare uma variável chamada **saldo** e atribua a ela o valor 1000. Reduza o saldo em 200 e imprima o novo valor no console. Resposta:

```
let saldo = 1000;
saldo -= 200;
console.log(saldo); // 800
```

Exercício 6: Enunciado: Declare duas variáveis, **altura** e **largura**, e atribua a elas os valores 10 e 5, respectivamente. Calcule a área de um retângulo utilizando essas variáveis e imprima o resultado no console. Resposta:

```
let altura = 10;
let largura = 5;
let area = altura * largura;
console.log(area); // 50
```

Exercício 7: Enunciado: Declare uma variável chamada **login** e atribua a ela o valor "admin". Verifique se o valor do login é igual a "admin" e imprima o resultado no console como um valor booleano. Resposta:

```
let login = "admin";
let resultado = login === "admin";
console.log(resultado); // true
```

Exercício 8: Enunciado: Declare uma variável chamada **ano** e atribua a ela o valor 2022. Verifique se o ano é maior ou igual a 2021 e imprima o resultado no console como um valor booleano. Resposta:

```
let ano = 2022;
let resultado = ano >= 2021;
console.log(resultado); // true
```

Exercício 9: Enunciado: Declare uma variável chamada **preco** e atribua a ela o valor 10.50. Aumente o preço em 20% e imprima o novo valor no console. Resposta:

```
let preco = 10.50;
preco *= 1.2;
console.log(preco); // 12.60
```

Exercício 10: Enunciado: Declare uma variável chamada **nomeCompleto** e atribua a ela uma string vazia. Em seguida, concatene o seu primeiro nome e sobrenome na variável e imprima o resultado no console. Resposta:

```
let nomeCompleto = "";
let nome = "Maria";
let sobrenome = "Silva";
nomeCompleto = nome + " " + sobrenome;
console.log(nomeCompleto); // "Maria Silva"
```

Esses exercícios abrangem alguns dos conceitos básicos discutidos até agora, incluindo declaração de variáveis, operações aritméticas, operadores de comparação, concatenação de strings e uso de operadores de atribuição. Espero que sejam úteis para praticar e reforçar o aprendizado!

Monte mais 10 exercícios diferentes dos já citados com enunciaado e respostas sobre o que foi visto até agora

Com certeza! Aqui estão mais 10 exercícios com enunciado e respostas sobre os conceitos que abordamos até agora:

Exercício 1: Enunciado: Declare uma variável chamada `num1` e atribua a ela o valor 10. Declare outra variável chamada `num2` e atribua a ela o valor 3. Calcule e imprima o resultado da multiplicação entre `num1` e `num2`. Resposta:

```
let num1 = 10;
let num2 = 3;
let resultado = num1 * num2;
console.log(resultado); // 30
```

Exercício 2: Enunciado: Declare uma variável chamada `nome` e atribua a ela uma string com o seu nome. Em seguida, verifique se o nome tem mais de 5 caracteres e imprima o resultado como um valor booleano. Resposta:

```
let nome = "Ana";
let resultado = nome.length > 5;
console.log(resultado); // false
```

Exercício 3: Enunciado: Declare uma variável chamada `precoProduto` e atribua a ela o valor 19.99. Declare outra variável chamada `desconto` e atribua a ela o valor 0.15. Calcule e imprima o valor do desconto aplicado ao preço do produto. Resposta:

```
let precoProduto = 19.99;
let desconto = 0.15;
let valorDesconto = precoProduto * desconto;
console.log(valorDesconto); // 2.9985
```

Exercício 4: Enunciado: Declare uma variável chamada `estudante` e atribua a ela o valor `true`. Declare outra variável chamada `trabalhador` e atribua a ela o valor `false`. Verifique se pelo menos uma das variáveis é `true` e imprima o resultado como um valor booleano. Resposta:

```
let estudante = true;
let trabalhador = false;
let resultado = estudante || trabalhador;
console.log(resultado); // true
```

Exercício 5: Enunciado: Declare uma variável chamada `idadeUsuario` e atribua a ela um valor numérico. Verifique se a idade do usuário está entre 18 e 30 anos (inclusive) e imprima o resultado como um valor booleano. Resposta:

```
let idadeUsuario = 25;
let resultado = idadeUsuario >= 18 && idadeUsuario <= 30;
console.log(resultado); // true
```

Exercício 6: Enunciado: Declare uma variável chamada `nota1` e atribua a ela o valor 7.5. Declare outra variável chamada `nota2` e atribua a ela o valor 8.2. Calcule a média das duas notas e imprima o resultado. Resposta:

```
let nota1 = 7.5;
let nota2 = 8.2;
let media = (nota1 + nota2) / 2;
console.log(media); // 7.85
```

Exercício 7: Enunciado: Declare uma variável chamada `saldo` e atribua a ela o valor 1000. Verifique se o saldo é menor que 500 ou maior que 1500 e imprima o resultado como um valor booleano. Resposta:

```
let saldo = 1000;
let resultado = saldo < 500 || saldo > 1500;
console.log(resultado); // false
```

Exercício 8: Enunciado: Declare uma variável chamada **numero** e atribua a ela um valor numérico. Verifique se o número é par e imprima o resultado como um valor booleano. Resposta:

```
let numero = 7;
let resultado = numero % 2 === 0;
console.log(resultado); // false
```

Exercício 9: Enunciado: Declare uma variável chamada **frase** e atribua a ela uma string com uma frase de sua escolha. Imprima o comprimento da frase (quantidade de caracteres) no console. Resposta:

```
let frase = "JavaScript é incrível!";
console.log(frase.length); // 20
```

Exercício 10: Enunciado: Declare uma variável chamada **valor** e atribua a ela o valor **null**. Verifique se o valor é igual a **null** e imprima o resultado como um valor booleano. Resposta:

```
let valor = null;
let resultado = valor === null;
console.log(resultado); // true
```

Esses exercícios adicionais abrangem mais aspectos dos conceitos discutidos até agora, como operações matemáticas, operadores lógicos, manipulação de strings e verificação de valores nulos. Espero que eles sejam úteis para você praticar e solidificar o conhecimento adquirido!

Mais 10 exercícios diferentes e bem elaborados

Com certeza! Aqui estão mais 10 exercícios com enunciado e respostas mais elaboradas:

Exercício 1: Enunciado: Escreva um programa que solicite ao usuário que digite um número e verifique se o número é positivo, negativo ou zero. Imprima o resultado no console. Resposta:

```
let numero = prompt("Digite um número:");
numero = Number(numero);

if (numero > 0) {
  console.log("O número é positivo.");
} else if (numero < 0) {
  console.log("O número é negativo.");
} else {
  console.log("O número é zero.");
}
```

Exercício 2: Enunciado: Escreva um programa que solicite ao usuário que digite dois números e verifique se o primeiro número é divisível pelo segundo número. Imprima o resultado no console. Resposta:

```
let num1 = prompt("Digite o primeiro número:");
let num2 = prompt("Digite o segundo número:");

num1 = Number(num1);
num2 = Number(num2);
```

```
if (num1 % num2 === 0) {  
  console.log("O primeiro número é divisível pelo segundo número.");  
} else {  
  console.log("O primeiro número não é divisível pelo segundo número.");  
}
```

Exercício 3: Enunciado: Escreva um programa que solicite ao usuário que digite um caractere. Verifique se o caractere é uma vogal ou consoante. Imprima o resultado no console. Resposta:

```
let caractere = prompt("Digite um caractere:").toLowerCase();  
  
if (caractere === "a" || caractere === "e" || caractere === "i" || caractere === "o" || caractere === "u") {  
  console.log("O caractere é uma vogal.");  
} else {  
  console.log("O caractere é uma consoante.");  
}
```

Exercício 4: Enunciado: Escreva um programa que solicite ao usuário que digite três números e imprima o maior número no console. Resposta:

```
let num1 = prompt("Digite o primeiro número:");  
let num2 = prompt("Digite o segundo número:");  
let num3 = prompt("Digite o terceiro número:");  
  
num1 = Number(num1);  
num2 = Number(num2);  
num3 = Number(num3);  
  
let maiorNumero = num1;  
  
if (num2 > maiorNumero) {  
  maiorNumero = num2;  
}  
  
if (num3 > maiorNumero) {  
  maiorNumero = num3;  
}  
  
console.log("O maior número é:", maiorNumero);
```

Exercício 5: Enunciado: Escreva um programa que solicite ao usuário que digite um ano. Verifique se o ano é bissexto e imprima o resultado no console. Resposta:

```
let ano = prompt("Digite um ano:");  
  
ano = Number(ano);  
  
if ((ano % 4 === 0 && ano % 100 !== 0) || ano % 400 === 0) {  
  console.log("O ano é bissexto.");  
} else {  
  console.log("O ano não é bissexto.");  
}
```

Exercício 6: Enunciado: Escreva um programa que solicite ao usuário que digite uma frase. Verifique se a frase é um palíndromo (ou seja, pode ser lida da mesma forma de trás para frente) e imprima o resultado no console. Resposta:

```
let frase = prompt("Digite uma frase:").toLowerCase();  
let fraseInvertida = "";  
  
for (let i = frase.length - 1; i >= 0; i--) {  
  fraseInvertida += frase[i];  
}
```

```
}

if (frase === fraseInvertida) {
  console.log("A frase é um palíndromo.");
} else {
  console.log("A frase não é um palíndromo.");
}
```

Exercício 7: Enunciado: Escreva um programa que solicite ao usuário que digite um número. Calcule e imprima a tabuada desse número de 1 a 10. Resposta:

```
let numero = prompt("Digite um número:");
numero = Number(numero);

for (let i = 1; i <= 10; i++) {
  let resultado = numero * i;
  console.log(numero + " x " + i + " = " + resultado);
}
```

Exercício 8: Enunciado: Escreva um programa que solicite ao usuário que digite um número. Verifique se o número é primo (divisível apenas por 1 e por ele mesmo) e imprima o resultado no console. Resposta:

```
let numero = prompt("Digite um número:");
numero = Number(numero);
let ehPrimo = true;

if (numero === 1) {
  ehPrimo = false;
} else {
  for (let i = 2; i < numero; i++) {
    if (numero % i === 0) {
      ehPrimo = false;
      break;
    }
  }
}

if (ehPrimo) {
  console.log("O número é primo.");
} else {
  console.log("O número não é primo.");
}
```

Exercício 9: Enunciado: Escreva um programa que solicite ao usuário que digite uma string. Verifique se a string é um palíndromo, ignorando espaços em branco e diferenças de maiúsculas e minúsculas. Imprima o resultado no console. Resposta:

```
let frase = prompt("Digite uma frase:");
frase = frase.toLowerCase().replace(/\s/g, "");

let fraseInvertida = frase.split("").reverse().join("");

if (frase === fraseInvertida) {
  console.log("A frase é um palíndromo.");
} else {
  console.log("A frase não é um palíndromo.");
}
```

Exercício 10: Enunciado: Escreva um programa que solicite ao usuário que digite um número e calcule o fatorial desse número. Imprima o resultado no console. Resposta:



```
let numero = prompt("Digite um número:");
numero = Number(numero);

let fatorial = 1;

for (let i = 2; i <= numero; i++) {
  fatorial *= i;
}

console.log("O fatorial de", numero, "é", fatorial);
```

Esses exercícios adicionais são um pouco mais desafiadores e exploram conceitos como estruturas de decisão, estruturas de repetição, manipulação de strings e cálculos matemáticos. Espero que eles sejam úteis para aprimorar suas habilidades de programação em JavaScript!

From ChatGPT, designed with  FancyGPT