**VARIÁVEIS**

**4 MANEIRAS DE DECLARAR UMA VARIÁVEL JAVASCRIPT**

* Usando **var**
* Usando **let**
* Usando **const**
* Usando nada

**O QUE SÃO VARIÁVEIS**

As variáveis ​​são contêineres para armazenar dados (**armazenar valores de dados**).

Neste exemplo, x, y e z, são variáveis, declaradas com a **VAR** palavra-chave:

var x = 5;  
var y = 6;  
var z = x + y;

Neste exemplo, x, y e z, são variáveis, declaradas com a **LET** palavra-chave:

let x = 5;  
let y = 6;  
let z = x + y;

Neste exemplo, x, y, e z, são **VARIÁVEIS ​​NÃO DECLARADAS**:

x = 5;  
y = 6;  
z = x + y;

**VARIÁVEL LET**

A let palavra-chave foi introduzida no [ES6 (2015)](https://www.w3schools.com/js/js_es6.asp) .

Variáveis ​​definidas com let não podem ser Redeclaradas.

As variáveis ​​definidas com let devem ser declaradas antes do uso.

Variáveis ​​definidas com let têm escopo de bloco.

**NÃO PODEM SER REDECLARADAS**

Variáveis ​​definidas com let não podem ser **redeclaradas** .

Você não pode redeclarar acidentalmente uma variável.

Com let você não pode fazer isso:

Exemplo:

let x = "John Doe";  
  
let x = 0;

**ESCOPO DO BLOCO**

Antes do ES6 (2015), o JavaScript tinha apenas **escopo global** e **escopo de função** .

O ES6 introduziu duas novas palavras-chave importantes do JavaScript: lete const.

Essas duas palavras-chave fornecem **escopo de bloco** em JavaScript.

Variáveis ​​declaradas dentro de um bloco { } não podem ser acessadas de fora do bloco:

{  
  let x = 2;  
}

Variáveis ​​declaradas com a palavra- varchave NÃO podem ter escopo de bloco.

As variáveis ​​declaradas dentro de um bloco { } podem ser acessadas de fora do bloco.

{  
  var x = 2;  
}

**VARIÁVEL CONST**

A const palavra-chave foi introduzida no [ES6 (2015)](https://www.w3schools.com/js/js_es6.asp) .

Variáveis ​​definidas com const não podem ser Redeclaradas.

As variáveis ​​definidas com const não podem ser reatribuídas.

Variáveis ​​definidas com const têm escopo de bloco.

**NÃO PODEM SER REATRIBUÍDAS**

const PI = 3.141592653589793;  
PI = 3.14;         
PI = PI + 10;

**DEVE SER ATRIBUÍDO**

As variáveis ​​JavaScript constdevem receber um valor quando são declaradas:

const PI = 3.14159265359;

**QUANDO USAR JAVASCRIPT CONST**

**Sempre declare uma variável com const quando souber que o valor não deve ser alterado.**

Use const quando você declarar:

* **Uma nova matriz**
* **um novo objeto**
* **Uma nova Função**
* **Um novo RegExp**

**OBJETOS CONSTANTES E ARRAY**

A palavra-chave consté um pouco enganadora.

Não define um valor constante. Ele define uma referência constante a um valor.

Por isso você NÃO pode:

* Reatribuir um valor constante
* Reatribuir uma matriz constante
* Reatribuir um objeto constante

Mas você pode:

* Alterar os elementos da matriz constante
* Alterar as propriedades do objeto constante

**MATRIZES CONSTANTES**

Você pode alterar os elementos de uma matriz constante:

// Você pode criar um array constante  
const cars = ["Saab", "Volvo", "BMW"];  
  
// Você pode alterar um elemento  
cars[0] = "Toyota";  
  
// Você pode adicionar um elemento  
cars.push("Audi");

Mas você NÃO pode reatribuir o array:

const cars = ["Saab", "Volvo", "BMW"];  
  
cars = ["Toyota", "Volvo", "Audi"];    // ERROR

**OBJETOS CONSTANTES**

Você pode alterar as propriedades de um objeto constante:

// Você pode criar um objeto const  
const car = {type:"Fiat", model:"500", color:"white"};  
  
// Você pode alterar uma propriedade  
car.color = "red";  
  
// Você pode adicionar uma propriedade  
car.owner = "Johnson";

**Mas você NÃO pode reatribuir o objeto:**

const car = {type:"Fiat", model:"500", color:"white"};  
  
car = {type:"Volvo", model:"EX60", color:"red"};    // ERROR

**TIPOS DE DADOS**

**JavaScript tem 8 tipos de dados**

* 1. **String**
  2. **Número**
  3. **Bigint**
  4. **Booleano**
  5. **Indefinido**
  6. **Nulo**
  7. **Símbolo**
  8. **Objeto**

**STRING**

**String são para armazenar e manipular textos.**

Uma string JavaScript é zero ou mais caracteres escritos entre aspas.

let nome = "Guilherme";

Você pode usar aspas simples ou duplas:

let nome1 = "Volvo XC60";  // aspas duplas  
let nome2 = 'Volvo XC60';  // aspas simples

Você pode usar aspas dentro de uma string, desde que não coincidam com as aspas ao redor da string:

let answer1 = "It's alright";  
let answer2 = "He is called 'Johnny'";  
let answer3 = 'He is called "Johnny"';

**COMPRIMENTO DE UMA STRING**

Para encontrar o comprimento de uma string, use a propriedade interna length:

let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
let length = text.length;

**QUEBRANDO LINHAS DE CÓDIGO LONGAS**

Para melhor legibilidade, os programadores costumam evitar linhas de código com mais de 80 caracteres.

Se uma instrução JavaScript não couber em uma linha, o melhor lugar para interrompê-la é após um operador:

document.getElementById("demo").innerHTML =  
"Hello Dolly!";

Você também pode quebrar uma linha de código **dentro de uma string de texto** com uma única barra invertida:

document.getElementById("demo").innerHTML = "Hello \  
Dolly!";

**STRINGS JAVASCRIPT COMO OBJETOS**

Normalmente, strings JavaScript são valores primitivos, criados a partir de literais:

let x = "John";

Mas strings também podem ser definidas como objetos com a palavra-chave new:

let x = "John";  
let y = new String("John");

**NUMBER**

JavaScript tem apenas um tipo de número. Os números podem ser escritos com ou sem casas decimais.

let x = 3.14;    // Numeros com ponto flutuante  
let y = 3;       // Numeros Inteiros

Números extragrandes ou extrapequenos podem ser escritos com notação científica (expoente):

let x = 123e5;    // 12300000  
let y = 123e-5;   // 0.00123

**ADICIONANDO NUMEROS E STRING**

O JavaScript usa o operador + para adição e concatenação

Numero + numero == soma

String + string == concatenação

**NaN – Não é um Número**

NaN é uma palavra reservada do JavaScript que indica que um número não é um número válido.

Tentar fazer aritmética com uma string não numérica resultará em NaN (não é um número):

let x = 100 / "Apple";

**MÉTODOS NUMÉRICOS JAVASCRIPT**

Esses **métodos numéricos** podem ser usados ​​em todos os números JavaScript:

O método **toString()**

O toString() método retorna um número como uma string.

Todos os métodos numéricos podem ser usados ​​em qualquer tipo de número (literais, variáveis ​​ou expressões):

let x = 123;  
x.toString();  
(123).toString();  
(100 + 23).toString();

O método **toExponential()**

toExponential() retorna uma string, com um número arredondado e escrito usando a notação exponencial.

Um parâmetro define o número de caracteres atrás do ponto decimal:

let x = 9.656;  
x.toExponential(2);  
x.toExponential(4);  
x.toExponential(6);

O método **toFixed()**

toFixed() retorna uma string, com o número escrito com um número especificado de casas decimais:

let x = 9.656;  
x.toFixed(0);  
x.toFixed(2);  
x.toFixed(4);  
x.toFixed(6);

O método **toPrecision()**

toPrecision() retorna uma string, com um número escrito com um comprimento especificado:

let x = 9.656;  
x.toPrecision();  
x.toPrecision(2);  
x.toPrecision(4);  
x.toPrecision(6);

O método **valueOf()**

valueOf()retorna um número como um número.

let x = 123;  
x.valueOf();  
(123).valueOf();  
(100 + 23).valueOf();

**CONVERTENDO VARIÁVEIS EM NÚMEROS**

Existem 3 métodos JavaScript que podem ser usados ​​para converter uma variável em um número:

Númber() -> converte números inteiros ou com ponto flutuante

parseInt() -> converte números inteiros

parseFloat() -> converte números com ponto flutuante

Os métodos acima não são métodos **numéricos** . Eles são métodos JavaScript **globais .**

**O Método Number()**

O Number() método pode ser usado para converter variáveis ​​JavaScript em números:

Number(true);  
Number(false);  
Number("10");  
Number("  10");  
Number("10  ");  
Number(" 10  ");  
Number("10.33");  
Number("10,33");  
Number("10 33");  
Number("John");

Number() também pode converter uma data em um número.

Number(new Date("1970-01-01"))

**O Método parseInt()**

parseInt() analisa uma string e retorna um número inteiro. Os espaços são permitidos. Apenas o primeiro número é retornado:

parseInt("-10");  
parseInt("-10.33");  
parseInt("10");  
parseInt("10.33");  
parseInt("10 20 30");  
parseInt("10 years");  
parseInt("years 10");

**O Método parseFloat()**

parseFloat() analisa uma string e retorna um número. Os espaços são permitidos. Apenas o primeiro número é retornado:

parseFloat("10");  
parseFloat("10.33");  
parseFloat("10 20 30");  
parseFloat("10 years");  
parseFloat("years 10");

**MÉTODOS DE OBJETO DE NÚMERO**

Esses **métodos de objeto** pertencem ao objeto **Number** :

## **Number.isInteger()**

O Number.isInteger() método retorna truese o argumento for um número inteiro.

Number.isInteger(10);  
Number.isInteger(10.5);

## **O método Number.isSafeInteger()**

Um inteiro seguro é um inteiro que pode ser representado exatamente como um número de precisão dupla.

O Number.isSafeInteger()método retorna truese o argumento for um inteiro seguro.

Number.isSafeInteger(10);  
Number.isSafeInteger(12345678901234567890);

## **O método Number.parseFloat()**

Number.parseFloat("10");  
Number.parseFloat("10.33");  
Number.parseFloat("10 20 30");  
Number.parseFloat("10 years");  
Number.parseFloat("years 10");

## **O Método Number.parseInt()**

Number.parseInt()analisa uma string e retorna um número inteiro.

Os espaços são permitidos. Apenas o primeiro número é retornado:

Number.parseInt("-10");  
Number.parseInt("-10.33");  
Number.parseInt("10");  
Number.parseInt("10.33");  
Number.parseInt("10 20 30");  
Number.parseInt("10 years");  
Number.parseInt("years 10");

**BIGINT**

As variáveis JavaScript BigInt são usadas para armazenar valores inteiros grandes que são muito grandes para serem representados por um JavaScript normal Number.

## **Precisão de inteiro JavaScript**

Números inteiros JavaScript são precisos apenas até 15 dígitos:

let x = 999999999999999;  
let y = 9999999999999999;

Em JavaScript, todos os números são armazenados em um formato de ponto flutuante de 64 bits (padrão IEEE 754).

Com esse padrão, números inteiros grandes não podem ser representados com exatidão e serão arredondados.

Por causa disso, o JavaScript só pode representar números inteiros com segurança:

Até **9007199254740991** +(2 53 -1)

e

Até **-9007199254740991** -(2 53 -1).

Valores inteiros fora desse intervalo perdem a precisão.

**COMO CRIAR UM BIGINT**

Para criar um BigInt, anexe n ao final de um inteiro ou chame BigInt():

let x = 9999999999999999;  
let y = 9999999999999999n;

ou

let x = 1234567890123456789012345n;  
let y = BigInt(1234567890123456789012345)

## **Operadores BigInt**

Os operadores que podem ser usados ​​em um JavaScript Number também podem ser usados ​​em um arquivo BigInt.

let x = 9007199254740995n;  
let y = 9007199254740995n;  
let z = x \* y;

## **BigInt decimais**

A BigIntnão pode ter decimais.

let x = 5n;  
let y = x / 2;  
// Error: Cannot mix BigInt and other types, use explicit conversion.

let x = 5n;  
let y = Number(x) / 2;

## **BigInt Hex, Octal e Binário**

let hex = 0x20000000000003n;  
let oct = 0o400000000000000003n  
let bin = 0b100000000000000000000000000000000000000000000000000011n;

**BOOLEANOS**

Um Bollean representa um dos dois valores: **true** ou **false**

## **Valores Booleanos**

Muitas vezes, na programação, você precisará de um tipo de dado que só pode ter um de dois valores, como

* SIM NÃO
* LIGADO DESLIGADO
* VERDADEIRO FALSO

Para isso, o JavaScript possui um tipo de dados **booleano .**Ele só pode assumir os valores **true** ou **false** .

## **A Função Booleana()**

Você pode usar a Boolean()função para descobrir se uma expressão (ou uma variável) é verdadeira:

Exemplo:

Boolean(10 > 9)

Ou ainda mais fácil:

(10 > 9)  
10 > 9

## **Comparações e condições**

O capítulo Comparações JS fornece uma visão geral completa dos operadores de comparação.

O capítulo JS Conditions fornece uma visão geral completa das declarações condicionais.

aqui estão alguns exemplos:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | equal to | if (day == "Monday") |
| > | greater than | if (salary > 9000) |
| < | less than | if (age < 18) |

O valor booleano de uma expressão é a base para todas as comparações e condições de JavaScript.

## **Booleanos de JavaScript como objetos**

Normalmente, os booleanos do JavaScript são valores primitivos criados a partir de literais:

let x = false;

Mas os booleanos também podem ser definidos como objetos com a palavra-chave new:

let y = new Boolean(false);

let x = false;  
let y = new Boolean(false);  
  
// typeof x returns boolean  
// typeof y returns object

FALTA

* 1. **Indefinido**
  2. **Nulo**
  3. **Símbolo**
  4. **Objeto**