CENTRO UNIVERSITÁRIO
INSTITUTO DE EDUCAÇÃO SUPERIOR DE BRASÍLIA – IESB

Lógica de Programação

MICHEL EMERSON BARROS COSTA, MSC.

1. Tipos, valores e variáveis
   - Número, texto, booleanos, *null* e *undefined*, objeto global, conversões, declaração de variável e escopo.

➤ Tipos, valores e variáveis

- Os programas de computador funcionam manipulando valores, como o número 3,14 ou o texto “Olá Mundo”;
- Os tipos de valores que podem ser representados e manipulados em uma linguagem de programação são conhecidos como tipos e uma das características mais fundamentais de uma linguagem de programação é o conjunto de tipos que ela aceita;
- Quando um programa precisa manter um valor para uso futuro, ele atribui o valor (ou “armazena” o valor) a uma variável. Uma variável define um nome simbólico para um valor e permite que o valor seja referido pelo nome.
- Os tipos de JavaScript podem ser divididos em duas categorias: tipos primitivos e tipos de objeto;
- Os tipos primitivos de JavaScript incluem números, sequências de texto (conhecidas como strings) e valores de verdade (conhecidos como booleanos);

Tipos, valores e variáveis

- Qualquer valor em JavaScript que não seja número, string, booleano, null ou undefined é um objeto. Um objeto (isto é, um membro do tipo objeto) é um conjunto de propriedades, em que cada propriedade tem um nome e um valor;
- Um objeto normal em JavaScript é um conjunto não ordenado de valores nomeados;
- Um array é um tipo especial de objeto, que representa um conjunto ordenado de valores numerados;
- JavaScript define outro tipo especial de objeto, conhecido como função. Uma função é um objeto que tem código executável associado. Uma função pode ser chamada para executar esse código executável e retornar um valor calculado;
- As funções que são escritas para serem usadas (com o operador new) para inicializar um objeto criado recentemente são conhecidas como construtoras. Cada construtora define uma classe de objetos – o conjunto de objetos inicializados por essa construtora. As classes podem ser consideradas como subtipos do tipo de objeto.

➤ Tipos, valores e variáveis

- Além das classes Array e Function, JavaScript básica define outras três classes úteis.
  - ✓ A classe Date define objetos que representam datas;
  - ✓ A classe RegExp define objetos que representam expressões regulares (uma poderosa ferramenta de comparação de padrões);
  - ✓ A classe Error define objetos que representam erros de sintaxe e de execução que podem ocorrer em um programa JavaScript;
- O interpretador JavaScript realiza a coleta de lixo automática para gerenciamento de memória. Isso significa que um programa pode criar objetos conforme for necessário e o programador nunca precisa se preocupar com a destruição ou desalocação desses objetos;
- JavaScript é orientada a objetos;

➢ Tipos, valores e variáveis

- Exercício criando variáveis (Aula04_Exercicio01)

```
var a = 3
let b = 4


var a = 30
b = 40

console.log(a, b)

a = 300
b = 400

console.log(a, b)

const c = 5
// c = 50
console.log(c)
```

Var var têm escopo global ou escopo de função.

let permite que você declare variáveis limitando seu escopo no **bloco**, **instrução**, ou em uma **expressão** na qual ela é usada.

constante não pode ser alterado por meio de reatribuição (ou seja, usando o operador de atribuição) e não pode ser redeclarado (ou seja, por meio de uma declaração de variável).

Escopo de variáveis nada mais é do que onde uma variável existe.

Exemplo: Se declararmos uma variável dentro de um bloco de ***if*** por exemplo, esta variável deve existir fora deste bloco de ***if***?

SIM, você deve usar ***var*** para declará-la.

NÃO, ou seja, esta variável deve existir **apenas** dentro do ***if***, você deve usar ***let*** para declará-la neste caso.

➢ Tipos, valores e variáveis

▪ Números

- ✓ JavaScript não faz distinção entre valores inteiros e valores em ponto flutuante;
- ✓ Quando um número aparece diretamente em um programa JavaScript, ele é chamado de literal numérico;
- ✓ Literais inteiros

```
0
3
10000000
```

- ✓ Literais em ponto flutuante
  - Os literais em ponto flutuante podem ter um ponto decimal; eles usam a sintaxe tradicional dos números reais. Um valor real é representado como a parte inteira do número, seguida de um ponto decimal e a parte fracionária do número.

```
3.14
2345.789
.33333333333333333
6.02e23          // 6.02 × 10^23
1.4738223E-32    // 1.4738223 × 10^-32
```

➢ Tipos, valores e variáveis

▪ Números

✓ Aritmética em JavaScript

- Os programas JavaScript trabalham com números usando os operadores aritméticos fornecidos pela linguagem... + para adição, – para subtração, * para multiplicação, / para divisão e % para módulo (resto da divisão);
- JavaScript aceita operações matemáticas mais complexas por meio de um conjunto de funções e constantes definidas como propriedades do objeto Math, por exemplo:

```
Math.pow(2,53)          // => 9007199254740992: 2 elevado à potência 53
Math.round(.6)          // => 1.0: arredonda para o inteiro mais próximo
Math.ceil(.6)           // => 1.0: arredonda para cima para um inteiro
Math.floor(.6)          // => 0.0: arredonda para baixo para um inteiro
Math.abs(-5)            // => 5: valor absoluto
```

➢ Tipos, valores e variáveis

▪ Números

✓ Aritmética em JavaScript

- A aritmética em JavaScript não gera erros em casos de estouro, estouro negativo ou divisão por zero. Quando o resultado de uma operação numérica é maior do que o maior número representável (estouro), o resultado é um valor infinito especial, que JavaScript indica como Infinity.
- Divisão por zero não é erro em JavaScript: ela simplesmente retorna infinito ou infinito negativo... há uma exceção: zero dividido por zero não tem um valor bem definido e o resultado dessa operação é o valor especial not-a-number, impresso como NaN. O NaN não é comparado como igual a qualquer outro valor, incluindo ele mesmo.

➢ Tipos, valores e variáveis

▪ Números

✓ Datas e horas

- JavaScript básico inclui uma construtora Date() para criar objetos que representam datas e horas.
- Esses objetos Date têm métodos que fornecem uma API para cálculos simples de data.
- Os objetos Date não são um tipo fundamental como os números.

```
var then = new Date(2010, 0, 1);   // O 1º dia do 1º mês de 2010
var later = new Date(2010, 0, 1,   // O mesmo dia, às 5:10:30 da tarde, hora local
                     17, 10, 30);
var now = new Date();              // A data e hora atuais
var elapsed = now - then;          // Subtração de data: intervalo em milissegundos

later.getFullYear()                // => 2010
later.getMonth()                   // => 0: meses com base em zero
later.getDate()                    // => 1: dias com base em um
later.getDay()                     // => 5: dia da semana. 0 é domingo, 5 é sexta-feira.
later.getHours()                   // => 17: 5 da tarde, hora local
later.getUTCHours()                // Horas em UTC; depende do fuso horário
```

➤ Tipos, valores e variáveis

▪ Números

✓ Exercício (Aula04_Exercicio02)

```
const peso1 = 1.0
const peso2 = Number('2.0')

console.log(peso1, peso2)
console.log(Number.isInteger(peso1))
console.log(Number.isInteger(peso2))

const avaliacao1 = 9.871
const avaliacao2 = 6.871

const total = avaliacao1 * peso1 + avaliacao2 *
 peso2
const media = total / (peso1 + peso2)

console.log(media.toFixed(2))
console.log(media.toString(2)) // em binário
console.log(typeof media)
console.log(typeof Number)
```

➤ Tipos, valores e variáveis

▪ Números cuidados

✓ Exercício (Aula04_Exercicio03)

```
console.log(7 / 0) -> retorna infinity
console.log("10" / 2) -> o resultado será calculado, o programa converte automaticamente
console.log('3' + 2) -> a string ganha
console.log('3' - 2) -> o resultado será calculado
console.log("Show!" * 2) -> Not a Number
console.log(0.1 + 0.7) -> Não dá 0,8 em função do ponto flutuante que o programa suporta,
                          ou seja gera imprecisões
console.log(10.toString()) -> gera um erro, não consegue neste formato
console.log((10.345).toFixed(2))
```

➤ Tipos, valores e variáveis

▪ Números CONVERSÃO

✓ String para Number:

- parseInt(n) - para inteiro ou parseFloat() – para decimais
- Number
  - Ex.: `var n1 = Number.parseInt(window.prompt("Digite o PRIMEIRO número: "))`
  - Ex.: `var n1 = Number(window.prompt("Digite o PRIMEIRO número: "))`

✓ Number para String:

- String(**n**) ou **n**.toString()
  - Ex.: `window.alert("A soma do número " + n1 + " + " + n2 + " é: " + string(soma))`

✓ Colocando casas decimais

- toFixed()... Ex: `soma.toFixed(2)` – coloca duas casas decimais na variável soma

✓ Trocar o ponto por vírgula

- Ex: `soma.toFixed(2).replace('.',',')`

✓ Colocar o REAL

- Ex: `soma.toLocaleString('pt-BR', {style: 'currency', currency: 'BRL'}))`

➢ Tipos, valores e variáveis

▪ Texto

- ✓ Uma *string* é uma sequência ordenada imutável de valores de 16 bits, cada um dos quais normalmente representa um caractere Unicode – as *strings* são um tipo de JavaScript usado para representar texto;
- ✓ O comprimento de uma *string* é o número de valores de 16 bits que ela contém. As *strings* (e seus arrays) de JavaScript utilizam indexação com base em zero: o primeiro valor de 16 bits está na posição 0, o segundo na posição 1 e assim por diante;
- ✓ A *string* vazia é a *string* de comprimento 0.
- ✓ *Strings* literais
  - Para incluir uma string literalmente em um programa JavaScript, basta colocar os caracteres da *string* dentro de um par combinado de aspas simples ou duplas (' ou ").

```
""   // A string vazia: ela tem zero caracteres
'testing'
"3.14"
```

➢ Tipos, valores e variáveis

▪ Texto

✓ *Strings* literais

- pode-se dividir uma string literal em várias linhas, finalizando cada uma delas, menos a última, com uma barra invertida (\);

```
"two\nlines" // Uma string representando 2 linhas escritas em uma linha
"one\        // Uma string de uma linha escrita em 3 linhas. Somente ECMAScript 5.
long\
line"
```

- Sequências de escape em strings literais: O caractere de barra invertida (\) tem um propósito especial nas strings em JavaScript. Combinado com o caractere que vem a seguir, ele representa um caractere que não pode ser representado de outra forma dentro da string. Por exemplo, \n é uma sequência de escape que representa um caractere de nova linha. O escape \', que representa o caractere de aspas simples (ou apóstrofo).

Tipos, valores e variáveis

Texto

✓ Sequência de escape

Sequência	Caractere representado
\0	O caractere NUL (\u0000)
\b	Retrocesso (\u0008)
\t	Tabulação horizontal (\u0009)
\n	Nova linha (\u000A)
\v	Tabulação vertical (\u000B)
\f	Avanço de página (\u000C)
\r	Retorno de carro (\u000D)
\"	Aspas duplas (\u0022)
\'	Apóstrofo ou aspas simples (\u0027)
\\	Barra invertida (\u005C)
\x *XX*	O caractere Latin-1 especificado pelos dois dígitos hexadecimais *XX*
\u *XXXX*	O caractere Unicode especificado pelos quatro dígitos hexadecimais *XXXX*

➢ Tipos, valores e variáveis

▪ Texto

✓ Trabalhando com strings

- Se o operador + é utilizado com números, ele os soma (concatena). Mas se esse operador é usado em strings, ele as une, anexando a segunda na primeira. Por exemplo:

  msg = "Hello, " + "world"; // Produz a string "Hello, world"

  greeting = "Welcome to my blog," + " " + name;

- Para determinar o comprimento de uma string (16 bits), use sua propriedade length. Ex. s.lenght

```
var s = "hello, world"          // Começa com um texto.
s.charAt(0)                     // => "h": o primeiro caractere.
s.charAt(s.length-1)            // => "d": o último caractere.
s.substring(1,4)                // => "ell": o 2º, 3º e 4º caracteres.
s.slice(1,4)                    // => "ell": a mesma coisa
s.slice(-3)                     // => "rld": os últimos 3 caracteres
s.indexOf("l")                  // => 2: posição da primeira letra l.
s.lastIndexOf("l")              // => 10: posição da última letra l.
s.indexOf("l", 3)               // => 3: posição do primeiro "l" em ou após 3
```

➤ Tipos, valores e variáveis

▪ Texto

✓ Trabalhando com strings

- Em ECMAScript 5, as strings podem ser tratadas como arrays somente para leitura e é possível acessar caracteres individuais (valores de 16 bits) de uma string usando colchetes em lugar do método charAt():

```
s = "hello, world";
s[0] // => "h"
s[s.length-1] // => "d"
```

➤ Tipos, valores e variáveis

▪ Texto

✓ Exercício (Aula04_Exercicio04)

```
const escola = "Cod3r"

console.log(escola.charAt(4))
console.log(escola.charAt(5))
console.log(escola.charCodeAt(3))
console.log(escola.indexOf('3'))

console.log(escola.substring(1))
console.log(escola.substring(0, 3))

console.log('Escola '.concat(escola).concat("!"))
console.log('Escola ' + escola + "!")
console.log(escola.replace(3, 'e'))

console.log('Ana,Maria,Pedro'.split(','))
var cartao = '111122233334444';
console.log("Cartão ****.****.****."+ cartao.substring(12));
```

➤ Tipos, valores e variáveis

▪ Texto

- ✓ Formatador de string (template String)
  - ▪ Placeholder – `$ (utilizar crase no texto)`
    - Ex: `window.alert(`A soma do número ${n1} + ${n2} é: ${soma}`)`
  - ▪ length – conta quantos caracteres tem a string... Sintaxe: **variável**.length
  - ▪ toUpperCase()... Sintaxe: **variável**.toUpperCase
  - ▪ toLowerCase() ... Sintaxe: **variável**.toLowerCase

➢ Tipos, valores e variáveis

▪ Valores booleanos

- ✓ Um valor booleano representa verdadeiro ou falso, ligado ou desligado, sim ou não. Só existem dois valores possíveis desse tipo. As palavras reservadas true e false são avaliadas nesses dois valores;
- ✓ Geralmente, os valores booleanos resultam de comparações feitas nos programas JavaScript. Por exemplo: a == 4;
- ✓ Os valores booleanos são comumente usados em estruturas de controle em JavaScript. Por exemplo, a instrução if/else de JavaScript executa uma ação se um valor booleano é true e outra ação se o valor é false.

```
if (a == 4)
  b = b + 1;
else
  a = a + 1;
```

➢ Tipos, valores e variáveis

▪ Valores booleanos

✓ O operador && executa a operação booleana E. Ele é avaliado como um valor verdadeiro se, e somente se, seus dois operandos são verdadeiros; caso contrário, é avaliado como um valor falso. O operador || é a operação booleana OU: ele é avaliado como um valor verdadeiro se um ou outro (ou ambos) de seus operandos é verdadeiro e é avaliado como um valor falso se os dois operandos são falsos. Por fim, o operador unário ! executa a operação booleana NÃO: ele é avaliado como true se seu operando é falso e é avaliado como false se seu operando é verdadeiro.

```
if ((x == 0 && y == 0) || !(z == 0)) {
    // x e y são ambos zero ou z não é zero
}
```

Tipos, valores e variáveis

Valores booleanos

- ✓ Exercício (Aula04_Exercicio05)

```
let isAtivo = false
console.log(isAtivo)

isAtivo = true
console.log(isAtivo)

isAtivo = 1
console.log(!!isAtivo)

console.log('os verdadeiros...')
console.log(!!3)
console.log(!!-1)
console.log(!!' ')
console.log(!!'texto')
console.log(![])
console.log(!!{})
console.log(!!Infinity)
console.log(!!(isAtivo = true))

console.log('os falsos...')
console.log(!!0)
console.log(!!'')
console.log(!!null)
console.log(!!NaN)
console.log(!!undefined)
console.log(!!(isAtivo = false))

console.log('pra finalizar...')
console.log(!!('' || null || 0 || '
'))

let nome = 'Lucas'
console.log(nome || 'Desconhecido')
```

➤ Tipos, valores e variáveis

▪ Null e undefined

- ✓ null é uma palavra-chave da linguagem avaliada com um valor especial, normalmente utilizado para indicar a ausência de um valor;
- ✓ JavaScript também tem um segundo valor que indica ausência de valor. O valor indefinido representa uma ausência mais profunda. É o valor de variáveis que não foram inicializadas e o valor obtido quando se consulta o valor de uma propriedade de objeto ou elemento de array que não existe;
- ✓ O valor indefinido também é retornado por funções que não têm valor de retorno e o valor de parâmetros de função quando os quais nenhum argumento é fornecido;
- ✓ Tanto null quanto undefined indicam uma ausência de valor e muitas vezes podem ser usados indistintamente;

➤ Tipos, valores e variáveis

▪ Null e undefined

- ✓ Você pode pensar em usar undefined para representar uma ausência de valor em nível de sistema, inesperada ou como um erro e null para representar ausência de valor em nível de programa, normal ou esperada.
- ✓ Se precisar atribuir um desses valores a uma variável ou propriedade ou passar um desses valores para uma função, null quase sempre é a escolha certa.

Tipos, valores e variáveis

Null e undefined

- ✓ Exercício (Aula04_Exercicio06)

```
let valor // não inicializada
console.log(valor)

valor = null // ausência de valor
console.log(valor)
// console.log(valor.toString()) // Err
o!

const produto = {}
console.log(produto.preco)
console.log(produto)

produto.preco = 3.50
console.log(produto)

produto.preco = undefined // evite atri
buir undefined
console.log(!!produto.preco)
// delete produto.preco
console.log(produto)

produto.preco = null // sem preço
console.log(!!produto.preco)
console.log(produto)
```

➤ Tipos, valores e variáveis

▪ Array

- ✓ Um array é um conjunto ordenado de valores. Cada valor é chamado de elemento e cada elemento tem uma posição numérica no array, conhecida como índice.
- ✓ Os arrays em JavaScript não são tipados: um elemento do array pode ser de qualquer tipo e diferentes elementos do mesmo array podem ser de tipos diferentes.
- ✓ Os elementos podem ser até objetos ou outros arrays, o que permite a criação de estruturas de dados complexas, como arrays de objetos e arrays de arrays.
- ✓ Os arrays em JavaScript são baseados em zero e usam índices de 32 bits: o índice do primeiro elemento é 0 e o índice mais alto possível é 4294967294 (232−2), para um tamanho de array máximo de 4.294.967.295 elementos.
- ✓ Os arrays em JavaScript são dinâmicos: eles crescem ou diminuem conforme o necessário e não há necessidade de declarar um tamanho fixo para o array ao criá-lo ou realocá-lo quando o tamanho muda.

➤ Tipos, valores e variáveis

▪ Array

✓ Exercício (Aula04_Exercicio07)

```
const valores = [7.7, 8.9, 6.3, 9.2]
console.log(valores[0], valores[3])
console.log(valores[4])

valores[4] = 10
console.log(valores)
console.log(valores.length)

valores.push({id: 3}, false, null, 'teste')
console.log(valores)

console.log(valores.pop())
delete valores[0]
console.log(valores)

console.log(typeof valores)
```

➢ Tipos, valores e variáveis

▪ Objeto

- ✓ O tipo fundamental de dados de JavaScript é o objeto;
- ✓ Um objeto é um valor composto: ele agrega diversos valores (valores primitivos ou outros objetos) e permite armazenar e recuperar esses valores pelo nome;
- ✓ Um objeto é um conjunto não ordenado de propriedades, cada uma das quais tendo um nome e um valor;
- ✓ Os objetos JavaScript são dinâmicos – normalmente propriedades podem ser adicionadas e excluídas –, mas podem ser usados para simular os objetos e as “estruturas” estáticas das linguagens estaticamente tipadas;
- ✓ Qualquer valor em JavaScript que não seja uma string, um número, true, false, null ou undefined, é um objeto;
- ✓ E mesmo que strings, números e valores booleanos não sejam objetos, eles se comportam como objetos imutáveis;

➤ Tipos, valores e variáveis

- **Objeto**
  - ✓ Os objetos são mutáveis e são manipulados por referência;
  - ✓ As coisas mais comuns feitas com objetos são: criá-los e configurar, consultar, excluir, testar e enumerar suas propriedades;
  - ✓ Uma propriedade tem um nome e um valor. Um nome de propriedade pode ser qualquer string, incluindo a string vazia, mas nenhum objeto pode ter duas propriedades com o mesmo nome;
  - ✓ O objeto é uma coleção de chave e valor, o identificador para o qual passamos um valor.

➤ Tipos, valores e variáveis

▪ Objeto

✓ Exercício (Aula04_Exercicio08)

```
const prod1 = {}
prod1.nome = 'Celular Ultra Mega'
prod1.preco = 4998.90
prod1['Desconto Legal'] = 0.40 // evitar atribu
tos com espaço

console.log(prod1)

const prod2 = {
    nome: 'Camisa Polo',
    preco: 79.90
}

console.log(prod2)
```

➢ Tipos, valores e variáveis

▪ Objeto Global

- ✓ O objeto global é um objeto normal de JavaScript que tem um objetivo muito importante: as propriedades desse objeto são os símbolos definidos globalmente que estão disponíveis para um programa JavaScript.
- ✓ Quando o interpretador JavaScript começa (ou quando um navegador Web carrega uma nova página), ele cria um novo objeto global e dá a ele um conjunto inicial de propriedades que define:
  - propriedades globais, como undefined, Infinity e NaN
  - funções globais, como isNaN(), parseInt() e eval()
  - funções construtoras, como Date(), RegExp(), String(), Object() e Array()
  - objetos globais, como Math e JSON

➢ Tipos, valores e variáveis

▪ Objeto Global

- ✓ Em JavaScript do lado do cliente, o objeto Window serve como objeto global para todo código JavaScript contido na janela do navegador que ele representa. Esse objeto global Window tem uma propriedade de autoreferência window que pode ser usada no lugar de this para se referir ao objeto global. O objeto Window define as propriedades globais básicas, mas também define muitos outros globais que são específicos para navegadores Web e para JavaScript do lado do cliente;

```
var global = this; // Define uma variável global para se referir ao objeto global
```

- ✓ Os objetos JavaScript são valores compostos: eles são um conjunto de propriedades ou valores nomeados. Ao usarmos a notação . fazemos referência ao valor de uma propriedade. Quando o valor de uma propriedade é uma função, a chamamos de método. Para chamar o método m de um objeto o, escrevemos o.m().

Tipos, valores e variáveis

Valores primitivos imutáveis e referências de objeto mutáveis

- Em JavaScript existe uma diferença fundamental entre valores primitivos (undefined, null, booleanos, números e strings) e objetos (incluindo arrays e funções). Os valores primitivos são imutáveis: não há como alterar (ou “mudar”) um valor primitivo. Isso é óbvio para números e booleanos – nem mesmo faz sentido mudar o valor de um número. No entanto, não é tão óbvio para strings.
- Como as strings são como arrays de caracteres, você poderia pensar que é possível alterar o caractere em qualquer índice especificado. Na verdade, JavaScript não permite isso e todos os métodos de string que parecem retornar uma string modificada estão na verdade retornando um novo valor de string. Por exemplo:

```
var s = "hello"; // Começa com um texto em letras minúsculas
s.toUpperCase(); // Retorna "HELLO", mas não altera s
s // => "hello": a string original não mudou
```

➢ Tipos, valores e variáveis

▪ Valores primitivos imutáveis e referências de objeto mutáveis

- ✓ Os valores primitivos também são comparados por valor: dois valores são iguais somente se têm o mesmo valor;
- ✓ Os objetos são diferentes dos valores primitivos. Primeiramente, eles são mutáveis – seus valores podem mudar:

```
var o = { x:1 };        // Começa com um objeto
o.x = 2;                // Muda-o, alterando o valor de uma propriedade
o.y = 3;                // Muda-o novamente, adicionando uma nova propriedade

var a = [1,2,3]         // Os arrays também são mutáveis
a[0] = 0;               // Muda o valor de um elemento do array
a[3] = 4;               // Adiciona um novo elemento no array
```

- ✓ Objetos não são comparados por valor: dois objetos não são iguais mesmo que tenham as mesmas propriedades e valores. E dois arrays não são iguais mesmo que tenham os mesmos elementos na mesma ordem:

```
var o = {x:1}, p = {x:1};       // Dois objetos com as mesmas propriedades
o === p                         // => falso: objetos distintos nunca são iguais
var a = [], b = [];             // Dois arrays vazios diferentes
a === b                         // => falso: arrays diferentes nunca são iguais
```

Tipos, valores e variáveis

Conversão de tipo

- ✓ A JavaScript é muito flexível quanto aos tipos de valores que exige;
- ✓ se a JavaScript quer uma string, ela converte qualquer valor fornecido em uma string. Se a JavaScript quer um número, ela tenta converter o valor fornecido para um número (ou para NaN, caso não consiga fazer uma conversão significativa). Alguns exemplos:

```
10 + " objects" // => "10 objects". O número 10 é convertido em uma string
"7" * "4" // => 28: as duas strings são convertidas em números
var n = 1 – "x"; // => NaN: a string "x" não pode ser convertida em um número
n + " objects " // => "NaN objects": NaN é convertido na string "NaN"
```

➢ Tipos, valores e variáveis

▪ Declaração de variável

✓ Antes de utilizar uma variável em um programa JavaScript, você deve declará-la. As variáveis são declaradas com a palavra-chave var, como segue:

```
var i;
var sum;
```

✓ Também é possível declarar várias variáveis com a mesma palavra-chave var:

```
var i, sum;
```

✓ Uma variável em JavaScript pode conter um valor de qualquer tipo. Por exemplo, em JavaScript é perfeitamente válido atribuir um número a uma variável e posteriormente atribuir uma string a essa variável:

```
var i = 10;
i = "ten";
```

3.9.1

➤ Tipos, valores e variáveis

▪ Escopo de variável

- ✓ O escopo de uma variável é a região do código-fonte de seu programa em que ela está definida. Uma variável global tem escopo global; ela está definida em toda parte de seu código JavaScript;
- ✓ As variáveis declaradas dentro de uma função estão definidas somente dentro do corpo da função. Elas são variáveis locais e têm escopo local. Os parâmetros de função também contam como variáveis locais e estão definidos somente dentro do corpo da função;

```
var scope = "global";           // Declara uma variável global
function checkscope() {
    var scope = "local";        // Declara uma variável local com o mesmo nome
    return scope;               // Retorna o valor local, não o global
}
checkscope()                    // => "local"
```

➢ Tipos, valores e variáveis

▪ Escopo de variável

✓ As definições de função podem ser aninhadas. Cada função tem seu próprio escopo local; portanto, é possível ter várias camadas de escopo local aninhadas. Por exemplo:

```
var scope = "global scope";           // Uma variável global
function checkscope() {
    var scope = "local scope";        // Uma variável local
    function nested() {
        var scope = "nested scope";   // Um escopo aninhado de variáveis locais
        return scope;                 // Retorna o valor em scope aqui
    }
    return nested();
}
checkscope()                          // => "nested scope"
```

➤ Tipos, valores e variáveis

▪ Escopo de variável

✓ Exercício (Aula04_Exercicio09)

```
{ { { {  var sera = 'Será???'
         console.log(sera) } } } }

console.log(sera)

function teste() {
    var local = 123
    console.log(local)
}

teste()
console.log(local)
```

```
var numero = 1
{
    var numero = 2
    console.log('dentro =', numero)
}
console.log('fora =', numero)
```

Tipos, valores e variáveis

Escopo de variável

✓ Exercício (Aula04_Exercicio10)

```
var numero = 1
{
    var numero = 2
    console.log('dentro =', numero)
}
console.log('fora =', numero)
```


THANKS
FOLKS !!
AND
HOPE YOU
ENJOYED


That's all Folks!


Thank You!


Obrigado
Thank You
Welalin
Vinaka
Kia Ora
Matondo
Dankon
Mochchakkeram
Arigato
Spasibo
Multumesc
Danke
Merci
Grazie
Kiitos
Chokrane
Maake

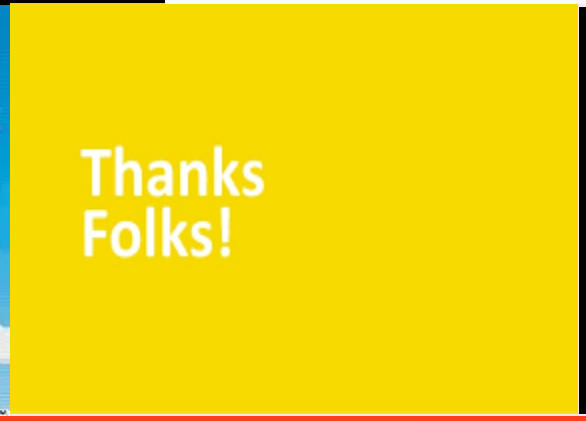

By the way.....


OBRIGADO!


THANK
YOU
GRACIAS
ARIGATO
SHUKURIA
BOLZÏN
MERCI
BÏYAN


OBRIGADO


OBRIGADO
PELA ATENÇÃO


Thanks
Folks!


Thank You!


Thank You !