

IOS – Instituto de  
Oportunidade Social

## JS 02 - Console, variáveis e operadores



- Tipos de Dados;
- Operadores;
- Variáveis;
- Objeto do Console.

IOS – Instituto de  
Oportunidade Social

## Tipos de Dados



## Valores e tipos de dados

Dentro do mundo do computador, existem apenas **dados**. Você pode **ler** dados, **modificar** dados, **criar** dados, mas o que não são dados não pode ser mencionado. Todos esses dados são armazenados como longas **sequências de bits** e, portanto, são fundamentalmente semelhantes.

## Valores

Os valores em um computador são quantidade de bits que representam uma informação.

```
1           // Valor do tipo numérico  
"Homer"    // Valor do tipo string (sequência de caracteres)
```

## Numérico

Valores podem ser o tipo numérico, que representam valores constantes. Como foi dito, o JavaScript tem um único tipo de número. Internamente, é representado como **ponto flutuante de 64 bits**. A parte fracionária de um número é separada utilizando o ponto: **9.81**  
Para números muito grandes ou pequenos, pode utilizar a notação científica:

Notação científica	Equivalente
2.998e8	$2.998 \times 10^8 = 299,800,000$
6.5e-3	$6.5 \times 10^{-3} = 0,0065$

## Números especiais

Existem três valores especiais em JavaScript que são considerados números, mas não se comportam como números normais. Os dois primeiros são **Infinity** e **-Infinity**, que representam os infinitos positivos e negativos. A expressão **Infinity - 1** ainda é infinito e assim por diante. O terceiro é o **NaN** significa “**Not a Number**” (“não é um número”), embora seja um valor do **tipo numérico**. Você obterá este resultado quando, por exemplo, tentar calcular **0/0** (zero dividido por zero).

## String

String é um outro tipo de dado, que é usado para representar texto, e devem ser envolvidas utilizando aspas simples ou aspas duplas.

**"Presentemente eu posso me considerar um sujeito de sorte"**

**"Porque apesar de muito moço, me sinto são e salvo e forte"**

**"E tenho comigo pensado: Deus é brasileiro e anda do meu lado"**

**"E assim já não posso sofrer no ano passado"**

Strings não podem ser divididas, multiplicadas ou subtraídas, mas o operador **+** pode ser usado para **concatenar**. Por exemplo:

Concatenando strings
"Instituto" + ' ' + 'da' + " " + "Oportunidade" + ' ' + 'Social'
Equivalente
Instituto da Oportunidade Social

## Comandos para Strings em Console

É possível inserir caracteres especiais de escape na string, Exemplo:  
**"Essa é a primeira linha\nE essa é a segunda linha"**

A sua string será interpretada como:

**Essa é a primeira linha**

**E essa é a segunda linha**

O caractere de escape `\n` indica um nova linha e retorno para o início do parágrafo.

Código escape	Resultado
<code>\n</code>	Nova linha e posiciona o cursor no início da nova linha do console.
<code>\t</code>	Tabulação horizontal. Move o cursor do console horizontalmente um espaço de tabulação.
<code>\v</code>	Tabulação vertical. Move o cursor do console verticalmente um espaço de tabulação.
<code>\\</code>	Barra invertida. Insere uma barra invertida na string.
<code>\'</code>	Aspa simples. Insere uma aspa simples na string.
<code>\"</code>	Aspas duplas. Insere uma aspas dupla na string.
<code>\b</code>	Backspace. Retorna o cursor uma posição para trás.



## Booleano

Muitas vezes é útil ter um valor que distingue apenas duas possibilidades, como **verdadeiro** e **falso**. Para isso, o JavaScript possui um tipo booleano, que possui apenas dois valores, **true** (verdadeiro) e **false** (falso). Existem dois tipos de operadores que retornam um resultado do tipo booleano: operadores de **comparação** (relacionais) e operadores **lógicos**.

## Vazio

Existem dois valores especiais, escritos como **null** (nulos) e **undefined** (indefinidos), que são usados para denotar a **ausência** de um **valor significativo**. Eles próprios são valores, mas não contêm nenhuma informação. Muitas operações na linguagem que não produzem um valor significativo resultam em indefinidas simplesmente porque têm que produzir algum valor.

A diferença de significado entre **undefined** e **null** é um acidente do design do JavaScript, e isso não é relevante na maioria dos casos, uma vez que não se consegue realizar algo objetivo com esses valores.

## Conversão automática

Quando um operador é aplicado ao tipo **"errado"** de valor, o JavaScript silenciosamente converte esse valor para o tipo de que precisa, usando um conjunto de regras que muitas vezes não são o que você deseja ou espera. Isso é chamado de **coerção de tipo**.

O JS faz de tudo para aceitar quase qualquer programa que você forneça, até mesmo programas que fazem coisas estranhas. Por exemplo, você pode fazer operações com tipos deferentes de dados:

Operação	Resultado
8 * null	0
"5" - 1	4
"5" + 1	51
"five" * 2	NaN

IOS – Instituto de  
Oportunidade Social

Operadores



## Aritméticos

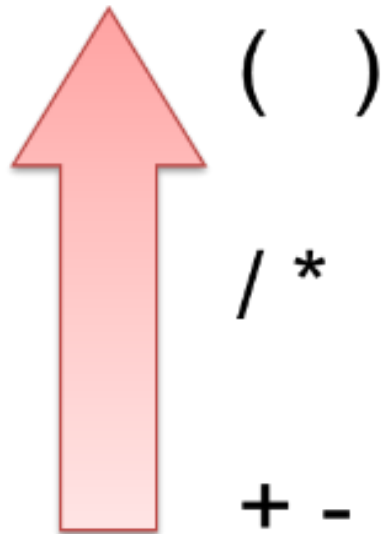
Operações aritméticas como adição ou multiplicação tomam dois valores numéricos e produzem um novo número a partir deles. Por exemplo:  **$100 + 4 * 11$**

Categoria	Operador	Descrição
Operadores aritméticos	+	Adição
	-	Subtração
	*	Multiplicação
	**	Exponenciação
	/	Divisão
	%	Módulo (Resto da divisão inteira)
	++	Incremento
	--	Decremento

## Precedência dos operadores

Nesse exemplo abaixo, primeiro é realizada a multiplicação e não a adição por causa da precedência dos operadores. A ordem de precedência dos operandos aritméticos é:

**Maior**



Desse modo a operação:  $3 * 5 + 2$

O programa primeiro faz a multiplicação e com o resultado dessa multiplicação realiza a soma. Os parênteses têm a maior precedência, portanto a operação:  $3 * (5 + 2)$

Primeiro é realizado a soma dentro dos parênteses e com o resultado dessa soma realiza a multiplicação.

## Relacionais

Os operadores relacionais são utilizados na realização de comparação entre valores. Exemplo:

Operação	Resultado
3 > 2	true
3 < 2	false

Categoria	Operador	Descrição
Operadores de comparação	==	Igual
	!=	Diferente
	<	Menor que
	<=	Menor ou igual
	>	Maior que
	>=	Maior ou igual

## Comparação

O JavaScript utiliza o operador triplo de igualdade, para garantir a comparação dos valores executando a conversão de tipos. Sendo assim no JavaScript:

Operação	Resultado
<code>2 == "2"</code>	True
<code>2 === "2"</code>	False



## Lógicos

Operadores lógicos são usados em programação para concatenar expressões que estabelecem uma relação de comparação entre valores.

Categoria	Operador	Descrição
Operadores lógicos	&&	Lógica “and” ou “e”, que retorna verdadeiro se todos os operandos forem verdadeiros.
		Lógica “or” ou “ou”, que retorna verdadeiro se pelo menos um operando for verdadeiro.
	!	Lógica “not” ou “não”, que inverte o valor lógico se é verdadeiro, retorna falso e se é falso retorna verdadeiro.

## Atribuição

O operador de atribuição ( = ) permite atribuir um valor a uma variável, por exemplo. O JavaScript permite utilizar uma forma contraída do operador de atribuição.

Operador	Exemplo	Equivalente
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y
<<=	x <<= y	x = x << y
>>=	x >>= y	x = x >> y
>>>=	x >>>= y	x = x >>> y
&=	x &= y	x = x & y
^=	x ^= y	x = x ^ y
=	x  = y	x = x   y
**=	x **= y	x = x ** y

## Unários

Nem todos operadores são símbolos. Alguns são escritos com palavras. Um exemplo é o operador **typeof**, que produz um valor de string nomeando o tipo do valor fornecido. Ex:

```
console.log(typeof 4.5)  
// → number  
console.log(typeof "x")  
// → string
```

O `typeof` de 4.5 retornou que ele é um tipo **numérico** e de “x” retornou uma **string**.

Outro operador unário é o símbolo de – (**menos**), que quando usado em um número transforma o valor de positivo para negativo. Por exemplo: **-8**

IOS – Instituto de  
Oportunidade Social

Variáveis



## Nomes de variável

Os **dados** de um programa são **armazenados** em variáveis. As variáveis são ferramentas indispensáveis na programação, são nelas que colocamos valores para podermos trabalhar com esses dados posteriormente.



**Importante!** O nome de variáveis ou constantes deve começar por letras ou underline ( \_ ) e pode conter números e não pode conter caracteres especiais. O nome também não pode ser uma palavra-chave ou palavra reservada.

## Exemplos de Nomes de variável

Nomes válidos:

- x, y, ola\_01, \_teste

Nomes inválidos:

- 12\_teste, nome pessoa, 1xx3

Programadores geralmente usam somente **letras minúsculas** para os **nomes de variáveis**, mas isso é uma convenção utilizada e não uma regra.

## Tipos de variável

No JavaScript, podemos declarar variáveis de três maneiras:

Usando a palavra reservada **var**

Usando a palavra reservada **let**

Usando a palavra reservada **const**

## Tipo var

A instrução **var** declara uma variável no escopo de uma função ou no escopo global e é opcional inicializar o seu valor.

```
var x = 1;           // Variável numerica
var nome = "Homer"; // Variável string
var teste;           // Variável não inicializada
```



## Tipo let

A palavra-chave **let** foi introduzida na ES6 em 2015. Variáveis definidas com **let** não podem ser redeclaradas, ou seja, você não pode declarar novamente uma variável com o mesmo nome. Por isso, nos programas mais recentes declarar variáveis utilizando **let** esta cada vez mais comum.

O let não permite isso	O var permite isso
<pre>let x = "John Doe"; let x = 0; //      SyntaxError:      'x'      has already been declared</pre>	<pre>var x = "John Doe"; var x = 0; // Sem problemas, 😊</pre>

## Escopo de Bloco

A instrução `let` também permite que você declare uma variável no **escopo de bloco**, ou seja, em uma **região delimitada** pela abertura e fechamento de chaves `{ }`. Por exemplo:

```
{  
    let z = 10;  
    // Comandos  
}
```

## Tipo const

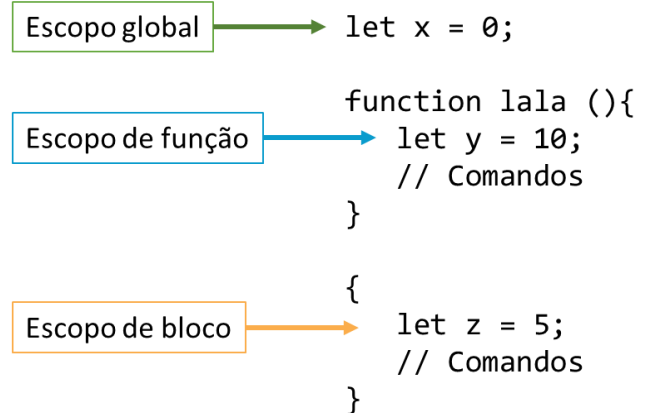
A palavra-chave **const** foi também introduzida na ES6 em 2015. Variáveis definidas com **const** não podem ser redeclaradas e não podem ter seu valor alterado, ou seja, você não pode declarar novamente uma variável com o mesmo nome e uma vez inicializada o seu **valor será o mesmo** até o **fim do programa**.

```
const PI = 3.141592653589793;  
PI = 3.14;           // Isso produzirá um erro  
PI = PI + 10;        // Isso também produzirá um erro
```

## Para Saber Mais

**Escopo de Variáveis:** O escopo é o conjunto de regras que determinam o uso e a validade de variáveis nas diversas partes do programa. Um escopo define uma região do programa definida pela abertura e fechamento de chaves `{ }`. JavaScript permite criar variáveis em três escopos: global, de função e de bloco. O escopo de função e de bloco são também chamados de escopo local. No caso, o escopo de uma variável define uma região do código onde a variável é visível, ou seja, ela pode ser acessada.

Por exemplo: a variável **x** foi declarada **globalmente** e pode ser acessada ou modificada em qualquer lugar do código, pois não há um escopo a delimitando. Então ela pode ser acessada ou modificada pela função ou pelo bloco normalmente. Já a variável **y** está definida **localmente** no escopo da função **lala** e só vai existir dentro do escopo dessa função, não podendo ser acessada fora da função. O mesmo acontece com a variável **z** só pode ser acessada no **escopo do bloco**.



```
Escopo global → let x = 0;

function lala () {
  Escopo de função → let y = 10;
                      // Comandos
}

{
  Escopo de bloco → let z = 5;
                   // Comandos
}
```

## Strict Mode

Você pode “falar” para o interpretador do JavaScript que você quer usar o **strict mode**. O **strict mode** indica que você não pode usar nenhuma variável sem a devida declaração. Para ativar esse modo, você deve colocar a string "use strict" no início do arquivo que você irá colocar seu código JavaScript. Vejamos o exemplo:

```
"use strict";  
x = 3.14;      // Isso gerará erro, pois a variável x não foi declarada
```

IOS – Instituto de  
Oportunidade Social

Objeto Console



O objeto console fornece acesso ao console (terminal) de *debugging* do navegador. O console possui diversos métodos, mas vamos aprender com calma quatro métodos do objeto console:

Método	Descrição
<code>clear()</code>	Limpa o console.
<code>error()</code>	Envia uma mensagem de erro no console.
<code>log()</code>	Envia uma mensagem no console.
<code>warn()</code>	Envia uma mensagem de aviso no console.

Utilizar o objeto console para fazer debug no seu código e visualizar as coisas acontecendo é melhor e mais prático do que usar a mensagem de alerta. Por isso, usaremos esse objeto constantemente nos nossos códigos.

IOS – Instituto de  
Oportunidade Social

Vamos Praticar





Apostila de JS

04.Javascript

Páginas 29 a 35

OBS: Acompanhar o passo a passo com o instrutor

IOS – Instituto de  
Oportunidade Social

## Exercícios



Receber os valores: **nome** (string), **altura** (number) e *peso* (number) em variáveis, calcular o **IMC** ( $\text{peso} / \text{altura}^2$ ) e mostrar no Console do navegador através do Javascript uma frase concatenada em **Template String**. Ex: “Fulano possui altura X e altura Y, seu IMC eh ...”

Subir arquivo no GitHub e enviar o link através do Moodle.