

IOS – Instituto de
Oportunidade Social

Aula JS 02 - Console, variáveis e operadores



- > Variáveis
- > Valores e tipos de dados
- > Operadores
- > Objeto console
- > Primeiros comandos em JS

IOS – Instituto de
Oportunidade Social

Variáveis



Os dados de um programa são armazenados em variáveis. 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.

Podemos declarar variáveis de três maneiras:

A palavra reservada **var**

A palavra reservada **let**

A palavra reservada **const**

> 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
```

> 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** está cada vez mais comum.

O let não permite isso

```
let x = 'John Doe';  
let x = 0;
```

O var permite isso

```
var x = "John Doe";  
var x = 0;
```

> 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
```

IOS – Instituto de
Oportunidade Social

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 a quantidade de bits que representam uma informação.

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

> Números

Valores podem ser do 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. Você deve separar a parte fracionária de um número utilizando o ponto:

9.81

> 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).**

> Strings

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"
```

Concatenando strings

```
"Instituto" + ' ' + 'da' + " " + "Oportunidade" + ' ' + 'Social'
```

Equivalente

```
Instituto da Oportunidade Social
```

O caractere de escape (`\n`) indica um nova linha e retorno do curso para o início do parágrafo. A barra invertida (`\`) é chamada de caractere de escape.

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.

Código de escape comuns no JavaScript.

> **Valores booleanos**

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).

TRUE OU FALSE

VERDADEIRO OU FALSO

> **Valores vazios**

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.

IOS – Instituto de
Oportunidade Social

Operadores



> Conversão automática

JavaScript 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 diferentes de dados:

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

Os operadores especificam uma avaliação/ação a ser executada em um ou mais operandos e podem ser aritméticos, relacionais e lógicos.

> Operadores aritméticos

A principal coisa a fazer com os números é realizar operações aritméticas. Operações aritméticas como adição ou multiplicação tomam dois valores numéricos e produzem um novo número a partir deles. Por exemplo:

Os símbolos **(+)** e **(*)** são chamados de operadores aritméticos. A tabela abaixo mostra os operadores aritméticos do JavaScript.

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

Operadores aritméticos do JavaScript

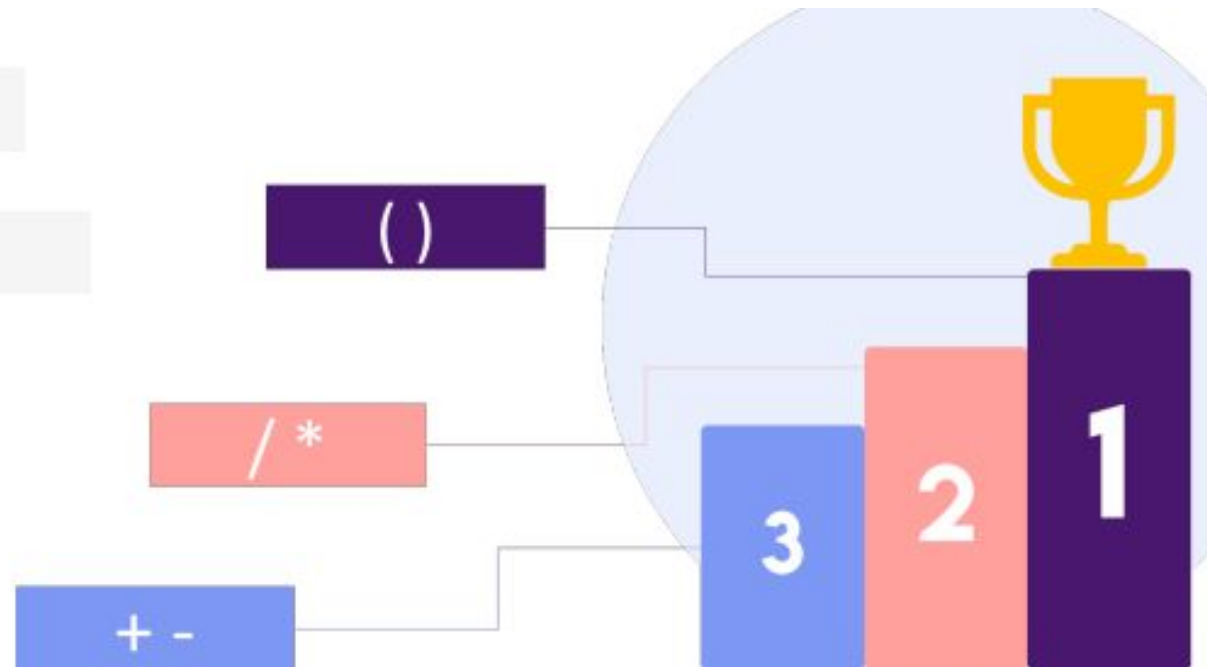
Operadores

Nesse exemplo, primeiro é realizada a multiplicação e não a adição por causa da precedência dos operadores.

$100 + 4 * 11$

$3 * 5 + 2$

$3 * (5 + 2)$



> Operadores de comparação (relacionais)

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

Operação	Resultado
3 > 2	true
3 < 2	false

> Operadores de comparação (relacionais)

Categoria	Operador	Descrição
Operadores de comparação	==	Valores iguais
	===	Valores e tipos iguais
	!=	Diferente
	<	Menor que
	<=	Menor ou igual
	>	Maior que
	>=	Maior ou igual

Operadores de comparação do JavaScript.

> Operadores de comparação (relacionais)

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
2 == "2"	True
2 === "2"	False

> Operadores lógicos

Operadores lógicos são usados em programação para concatenar expressões que estabelecem uma relação de comparação entre valores. Os principais operadores lógicos são mostrados abaixo:

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.

Operadores lógicos do JavaScript.

> Operador de 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**. A tabela a seguir, mostra diversos exemplos do **operador de atribuição** com os operadores, por exemplo: a instrução `x += 2` é equivalente a `x = x + 2`, ou seja, não precisamos repetir o `x` novamente

Operador	Exemplo	Equivalente
=	$x = y$	$x = y$
+=	$x += y$	$x = x + y$
-=	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
%=	$x \% = y$	$x = x \% y$
**=	$x ** = y$	$x = x ** y$
<<=	$x << = y$	$x = x << y$
>>=	$x >> = y$	$x = x >> y$
>>>=	$x >>> = y$	$x = x >>> y$

Continuação na página 25

> Operadores unários

Nem todos os operadores são símbolos. Alguns são escritos como palavras. Um exemplo é o operador `typeof`, que produz um valor de string nomeando o tipo do valor que você forneceu. Por exemplo:

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

IOS – Instituto de
Oportunidade Social

Objeto console



> 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.

> Vamos Praticar

IOS – Instituto de
Oportunidade Social

Exercício



Crie um arquivo main.js e crie um programa que leia quatro notas de um aluno e tire a média das notas no console.log apresente a seguinte mensagem “**A média do aluno com as 4 notas é:**”