



Linguagem para Web I

JavaScript

Parte 2 - Elementos da linguagem

Versão de 21.10.2012



Elementos da linguagem

1. A sintaxe do JavaScript;
2. Quais os tipos de dados;
3. O que são variáveis e como criá-las;
4. *Arrays*;
5. Quais os operadores disponíveis;
6. Comandos e palavras reservadas;



1. A sintaxe do JavaScript

✓ Case-sensitive.

escreveTexto() é diferente de **Escrevetexto()**

✓ Uso de ponto-e-vírgula ; ao final de cada comando.

Embora seja opcional, é considerado boa prática.

✓ Blocos de comandos são delimitados por chaves { }



1. A sintaxe do JavaScript

Comentários:

Única linha:

```
// Sou uma linha de comentário
```

Em bloco (várias linhas):

```
/* Sou um bloco  
de comentário  
com múltiplas linhas */
```



2. Tipos de dados



number

Não existe distinção entre números inteiros e reais.

Ex.: **2** e **3.659** tem o mesmo tipo *number*;



string

Sequência de caracteres. Ex.: “Olá Pessoal”, ‘Olá Pessoal’. Primeira forma mais indicada;



boolean

true / false;



null

Um valor especial que denota um valor nulo;



undefined

Denota um dado cujo valor ainda não foi definido.



2. Tipos de dados - number

Aritmética em JavaScript

<code>Math.pow(2,53)</code>	<code>// => 9007199254740992: 2 na potência de 53</code>
<code>Math.round(.6)</code>	<code>// => 1.0: arredonda para o inteiro mais próximo</code>
<code>Math.ceil(.6)</code>	<code>// => 1.0: arredonda pra cima</code>
<code>Math.floor(.6)</code>	<code>// => 0.0: arredonda pra baixo</code>
<code>Math.abs(-5)</code>	<code>// => 5: valor absoluto</code>
<code>Math.max(x,y,z)</code>	<code>// Máximo</code>
<code>Math.min(x,y,z)</code>	<code>// Mínimo</code>
<code>Math.random()</code>	<code>// number pseudo-aleatório x onde 0 <= x < 1.0</code>
<code>Math.PI</code>	<code>// π: Circunferência de um círculo / diâmetro</code>
<code>Math.E</code>	<code>// e: A base de um logaritmo</code>
<code>Math.sqrt(3)</code>	<code>// Raiz quadrada de 3</code>
<code>Math.pow(3, 1/3)</code>	<code>// Raiz cúbica de 3</code>
<code>Math.sin(0)</code>	<code>// Trigonometria: também Math.cos, Math.atan, etc.</code>
<code>Math.log(10)</code>	<code>// Logaritmo de 10</code>
<code>Math.log(100)/Math.LN10</code>	<code>// Logaritmo 100 na base 10</code>
<code>Math.log(512)/Math.LN2</code>	<code>// Logaritmo 512 na base 2</code>
<code>Math.exp(3)</code>	<code>// Math.E cúbica</code>



2. Tipos de dados - number

Datas e horas em JavaScript

JavaScript inclui um construtor **Date()** para criação de objetos que representam datas e horas. Estes objetos oferecem métodos para trabalhar datas e horas de forma simples.

```
// Declaro o objeto 'entao' com o 1º dia do primeiro mês de 2012
var entao = new Date(2012, 0, 1);

// Mesmo dia às 17:10:30, hora local
var depois = new Date(2012, 0, 1, 17, 10, 30);


// Data e hora atuais
var agora = new Date();

// Subtração de data: intervalo em milisegundos
var passado = agora - entao;
```



2. Tipos de dados - number

Datas e horas em JavaScript

```
depois.getFullYear()      // Retorna 2012  
  
depois.getMonth()         // Retorna mês baseado de 0 a 11   
  
depois.getDate()         // Retorna dia de 1 a 31  
  
depois.getDay()          // Dia da semana. 0 é domingo e 5 é sexta  
  
depois.getHours()         // Hora local (17 para 5pm)  
  
depois.toLocaleDateString() // 01/01/2012  
  
depois.toLocaleTimeString() // 05:10:30 PM
```




2. Tipos de dados - string

Texto em JavaScript

Para incluir uma string em JavaScript, simplesmente coloque caracteres entre aspas duplas ou aspas simples (' ou ").

```
" " // Uma string vazia: tem zero caracteres
```

```
'testando'
```

```
"3.14"
```

```
'name="formContato"'
```

```
"Você prefere 'IHC' ou 'LPW'?"
```

```
"Essa string tem\nduas linhas"
```

```
" $\pi$  é o raio de uma circunferência de um círculo"
```



3. Variáveis

Quando um programa precisa manter um valor para uso futuro, este programa “armazena” este valor a uma variável. Uma variável define um nome simbólico para um valor e permite que ele seja referenciado por este nome.

Antes de usar uma variável em JavaScript você deve declará-la. Variáveis são declaradas com a palavra-chave **var** assim:

```
var i; // inicia com o valor undefined
```

```
var soma; // inicia com o valor undefined
```

```
var nome = "Rodrigo"; // inicia com o valor Rodrigo
```



3. Variáveis

É possível dispensar o uso do **var** em alguns casos declarando e iniciando a variável assim:

```
nome    = "Rodrigo";      // variavel local  
end     = "Rua Amélia";  // variavel local
```

Você pode declarar múltiplas variáveis usando o mesmo **var**:

```
var i, soma; // iniciam com o valor undefined
```

E pode combinar declarações na inicialização das variáveis:

```
var nome = "Rodrigo";
```

```
var i = 0, j = 0, k = 100;
```



3. Variáveis

- ✓ O primeiro caractere do nome de uma variável **não** pode ser número;
- ✓ Os nomes das variáveis podem conter os seguintes caracteres:
 - ✓ letras de “A” a “Z”
 - ✓ letras de “a” a “z”
 - ✓ números
 - ✓ Caractere “_” (sublinhado ou underline)
- ✓ Lembrando: **Case-sensitive**



3. Variáveis - Escopo

O escopo é a abrangência ou a região do seu programa onde a variável é definida e pode ter seu valor acessado. O escopo do ser:

✓ **global**

variável criada fora de qualquer função e podem ser acessadas por qualquer parte do script

✓ **local**

variáveis criadas dentro de uma função e que só podem ser acessadas dentro daquela função.



3. Variáveis - Escopo

```
var escopo = "global" // declara variável global

function verificaEscopo() { // declara uma função
    var escopo = "local"; // declara variável local
    return escopo;
}

// ao chamar esta função, ela retorna o valor de escopo
verificaEscopo() // => "local"
```



3. Variáveis - Escopo

Apesar de poder declarar variáveis globais sem o uso do **var**, ele é sempre necessário para declarar uma variável local.

```
escopo = "global" // declara variável global sem var

function verificaEscopo() {
    escopo = "local"; // acabamos de mudar a variável global
    meuEscopo = "local"; // declara uma nova variável global
    return [escopo, meu escopo]; // retorna dois valores
}

verificaEscopo() // => ["local", "local"]

escopo // => "local": variável local foi alterada

meuEscopo // => "local": desorganização de código, pois
declarou uma nova variável global dentro de uma função
```



3. Variáveis - Escopo

O código abaixo apresentará erro no browser, pois as variáveis **fator** e **indice** são locais (só podem ser acessadas dentro do escopo da função):

```
<script>

function resultado() {
    var fator = 10; // var local
    var indice = 5; // var local
}

</script>

<h1>Variaveis locais e globais</h1>

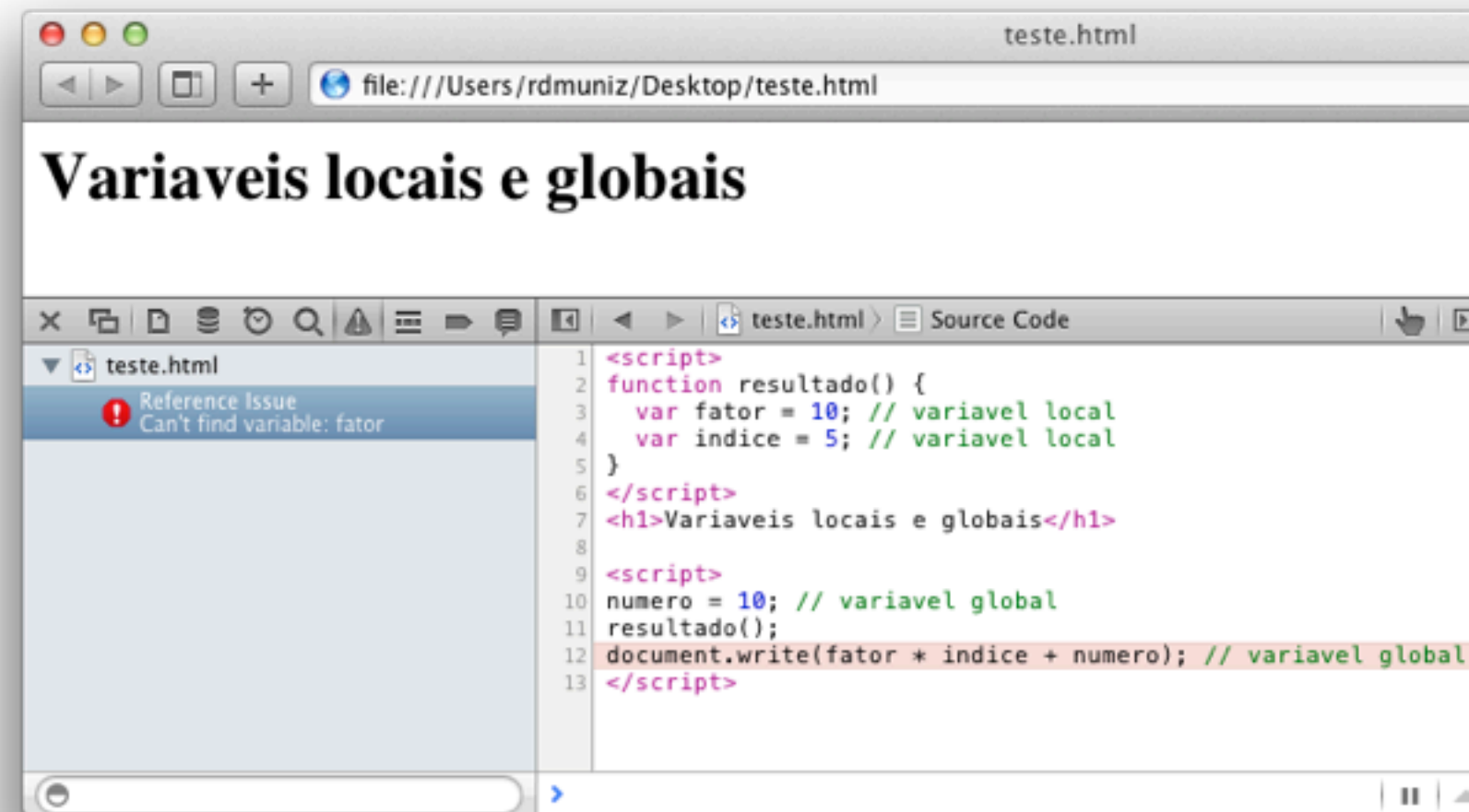
<script>

numero = 10; // var global

resultado();

document.write(fator * indice + numero); // var global

</script>
```





3. Variáveis - Escopo

O código abaixo agora estará acessando as variáveis locais **fator** e **indice** dentro do escopo da função:

```
<script>

function resultado() {

    var fator = 10; // var local

    var indice = 5; // var local

    document.write(fator * indice + numero); // var global

}

</script>

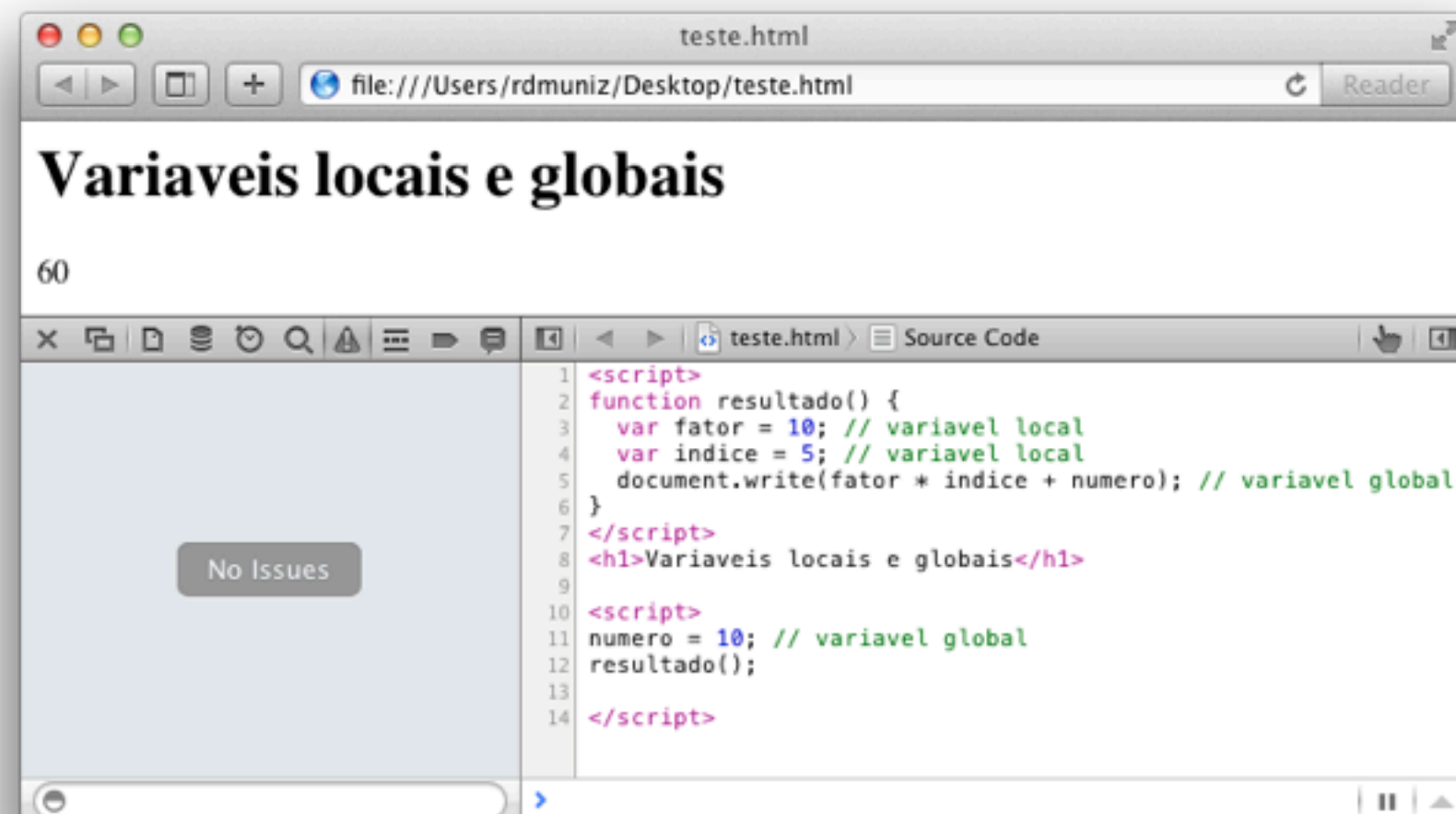
<h1>Variaveis locais e globais</h1>
```

```
<script>

numero = 10; // var global

resultado();

</script>
```





Elementos da linguagem - Revisando

1. A sintaxe do JavaScript;
2. Quais os tipos de dados;
3. O que são variáveis e como criá-las;
4. *Arrays*;
5. Quais os operadores disponíveis;
6. Comandos e palavras reservadas;



4. Arrays

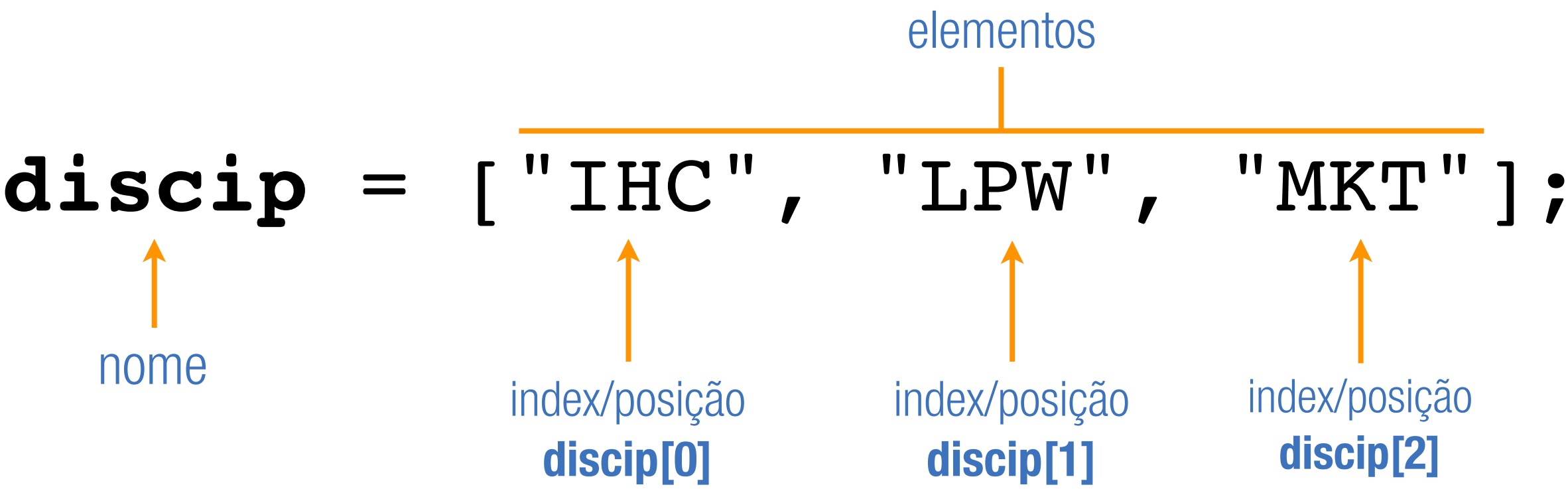
Também chamado de matriz e vetor, *array* é uma coleção de valores ordenados. Cada valor é chamado de elemento e cada elemento tem uma posição numérica (ordem) no *array*, chamada de *index*.

Um *array* pode conter elementos de vários tipos *number*, *string* e até outros *arrays*.



4. Arrays - Estrutura

Coleção de disciplinas



Podemos comparar
um *array* a uma tabela:

	discip
0	IHC
1	LPW
2	MKT



4. Arrays - Como criar

A forma mais simples de criar um *array* é com um *array* literal, que é uma lista de valores separados por vírgula dentro de colchetes. Exemplo:

```
// Um array sem elementos
```

```
var vazio = [];
```

```
// Array com 5 elementos do tipo number
```

```
var primos = [2, 3, 5, 7, 11];
```

```
// 3 elementos de vários tipos
```

```
var variado = [1.1, true, "Muniz"];
```

Se você omitir um valor num *array* literal, o elemento omitido terá o valor **undefined**:

```
// Um array com 4 elementos, onde um elemento é undefined.
```

```
var contagem = [1,,3,4];
```

Exercício

1. Pense em 3 conjuntos de coisas (objetos, pessoas...). Crie 3 arrays, um para cada conjunto. Sendo 8 itens em cada array (ex.: 8 pessoas).

2. Em seguida sublinhe:

O elemento na posição 5 do array 1

O elemento da posição 2 do array 2

Os elementos da posição 1 e 7 do array 3

Formato	Entrega
Individual	Na aula
No papel	Não esqueça nome, período e turno



4. Arrays - Como criar

Outra forma de criar um *array* é com o construtor `Array()`. Há 3 formas de usar este construtor:

```
var contagem    = new Array();    // Sem argumentos. Cria um array sem elementos.
```

```
// Um argumento numérico especificando o tamanho do array.
```

```
var contagem = new Array(10); // Neste caso, 10 elementos.
```

```
// Já especificando dois ou mais elementos ou um elemento não numérico
```

```
var numeros    = new Array(3, 2, 1);
```

```
var nomes      = new Array("Muniz");
```

```
var elementos  = new Array(3, 2, 1, "Muniz");
```



4. Arrays - Lendo e Escrevendo

Você pode acessar um elemento de um *array* usando o operador **[]**. Antes dos colchetes você referencia o nome do *array*. Tanto para ler quanto para escrever valores dos elementos.

```
// Lê o elemento 0 do array discip guardando na variável valor  
var valor = discip[0];
```

```
// Escreve 3.14 como sendo o valor do elemento 1 do array discip  
discip[1] = 3.14;
```

```
// Declara-se uma variável local i de valor 2  
var i = 2;
```

```
// Usa-se o valor da variável i para escrever no elemento 2  
discip[i] = 3;
```

```
// Usa-se o valor da variável i para escrever no elemento 3  
discip[i + 1] = "Muniz";
```

```
// Usa-se o valor da variável i para ler o elemento 2 e escrever no elemento 3  
discip[discip[i]] = discip[0];
```




4. Arrays - **length** (Tamanho)

Todo *array* tem uma propriedade *length*, que é a quantidade de elementos deste *array*.

```
[].length // => 0: Este array não tem elementos
```

```
['a', 'b', 'c'].length // => 3: O tamanho é 3 e o index maior é 2
```

Se você estabelecer o *length* de um *array* como sendo menor que a quantidade total de seus elementos, os elementos com o *index* maior que este número serão apagados:

```
a = [1,2,3,4,5]; // Iniciou o array a com 5 elementos.  
a.length = 3; // a agora é [1,2,3]. Indexes 3 e 4 foram apagados  
a.length = 0; // Excluiu todos. a agora é [].  
a.length = 5; // Length agora é 5, mas sem elementos. É um novos array  
como Array(5)
```



5. Operadores

São os mesmos operadores encontrados no C++ e Java.

Operadores aritméticos		Operadores lógicos		Operadores de bits	
- n	negação	!=	diferente de	&	and
++n, n++	incremento	==	igual a		or
--n, n--	decremento	>	menor que	^	xor
*	multiplicação	<	maior que	~	not
/	divisão	>=	maior ou igual a	<<	desloc. à esquerda
%	resto	<=	menor ou igual a	>>	desloc. à direita
+	adição e concatenação (string)		or	>>>	desloc. à direita sem sinal
-	subtração	&&	and	Operadores de objetos	
Operadores de atribuição		!	not	new	criação
=	atribuição	?:	condicional	delete	remoção
op=	atribuição com operação op	,	vírgula	typeof	tipo do objeto
				void	descarta o tipo



5. Operadores de comparação - Exemplo

```
<script>
  var valor = 10;

  if (valor == 10){          // igual a
    alert("Igual a 10!");
  }
  if (valor != 10){          // diferente de
    alert("Diferente de 10!");
  }
  if (valor > 9){             // maior que
    alert("Maior que 9!");
  }
  if (valor >= 10){           // maior igual a
    alert("Maior igual a 10!");
  }
  if (valor == true){         // igual a boolean
    alert("É verdadeiro!");
  }
</script>
```



5. Operadores aritméticos - Exemplo

```
<script>
```

```
var valor = 10;
```

```
var indice = 5.4;
```

```
var outro = 3;
```

```
var a = valor + indice; // atribui 15.4
```

```
var b = valor - outro; // atribui 7
```

```
var c = valor * indice; // atribui 54
```

```
var d = valor++; // atribui 10 a d e 11 a valor
```

```
var e = ++valor; // atribui 11 a e e 11 a valor
```

```
var f = -outro; // atribui -3
```

```
var g = 2 + 3 * outro; // atribui -11
```

```
var g = (2 + 3) * outro; // atribui -15
```

```
</script>
```



5. Operadores de atribuição - Exemplo

```
<script>
```

```
  var valor    = 10;
```

```
  var numero   = 10;
```

```
  var outro    = 3;
```

```
  var fav      = 1;
```

```
  // fav recebe 11 (10 mais o valor que já tinha)
```

```
  var fav      += valor;
```

```
  var numero   -= outro; // atribui 7 a numero
```

```
  var valor    /= outro; // atribui 3.33333333 a valor
```

```
  var numero   *= valor; // atribui 70 a numero
```

```
</script>
```



5. Operadores lógicos - Exemplo

```
<script>
```

```
  var numero = 10;
```

```
  var valor  = 5;
```

```
  var outro  = 5;
```

```
  var fator  = 7;
```

```
  var a = (numero > valor); // true
```

```
  var b = (valor == outro); // true
```

```
  var c = (fator <= outro); // false
```

```
  var d = (numero == 25); // false
```

```
  var cond1 = (a && b); // true
```

```
  var cond2 = (a && c); // false
```

```
  var cond3 = (c && d); // false
```

```
  var cond4 = (a || c); // a ou b são verdadeiras? => true
```

```
  var cond5 = (c || d); // c ou d são verdadeiras? => false
```

```
  var cond6 = !(a && b); // a ou b NÃO são verdadeiras? => false
```

```
  var cond7 = !d; // d NÃO é verdadeira? => true
```

```
</script>
```



6. Comandos e palavras reservadas

JavaScript reserva alguns termos para ela mesma, que não podem ser usados como identificadores, ou seja, não podem ser usados como nomes de suas variáveis ou funções.

break	delete	function	return	typeof		
case	do	if	switch	var		
catch	else	in	this	void		
continue	false	instanceof	throw	while		
debugger	finally	new	true	with		
default	for	null	try			
class	const	enum	export	extends	import	super
implements	let	private	public	yield		
interface	package	protected	static			
arguments	eval					
abstract	double	goto	native	static		
boolean	enum	implements	package	super		
byte	export	import	private	synchronized		
char	extends	int	protected	throws		
class	final	interface	public	transient		
const	float	long	short	volatile		



Referências

FLANAGAN, David. JavaScript O Guia Definitivo. Editora: Bookman, 2004.



:-)

Obrigado!

professor@rodrigomuniz.com