MÓDULO

JAVASCRIPT/AJAX

UNIDADE

VARIÁVEIS JAVASCRIPT

Instituto Tecnológico

ÍNDICE

OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Definir os tipos de variáveis e compreender as suas diferenças.
- Criar variáveis e ser capaz de lhes dar o melhor uso possível.

INTRODUÇÃO

O JavaScript é baseado em funções e operadores, e, como todas as linguagens de programação, utiliza um sistema de variáveis para controlar valores e gerir operações.

Nesta unidade, irá aprender o básico da programação, não apenas JavaScript, mas também outras linguagens, já que a maioria das linguagens é baseada num sistema de variáveis para controlar funções.

1. DEFINIÇÃO DE VARIÁVEL

Uma variável é um dado utilizado na programação que fica armazenado na memória e que pode ser usado para apagar ou modificar, graças ao facto de ser identificado por um nome único. Portanto, não é possível ter duas variáveis com o mesmo nome. Para nomear variáveis, podemos usar letras e números (desde que a variável comece com uma letra) bem como um "_".

Como em todas as linguagens, há várias palavras reservadas que não podem ser usadas para nomes de variáveis. Essas palavras são as seguintes:

abstract	do	if	package
throw	boolean	double	implements
private	throws	break	else
import	protected	transient	byte
extendsin	public	true	case
false	instanceof	return	try
catch	final	int	short
var	char	finally	interface
static	void	class	float
super	long	while	const
with	for	native	switch
continue	functionnew	synchronized	default
goto	null	this	eval

Estas palavras estão reservadas para a linguagem de programação.

2. USO DE VARIÁVEIS

O JavaScript não possui uma secção predefinida para criar variáveis como outras linguagens de programação, logo, podem ser criadas em qualquer lugar no código.

Também não é necessário declarar as variáveis, ou seja, para criar uma variável que se chamará de contador e conterá números, basta atribuir um valor a essa variável:

```
contador = 0;
```

No caso de querer definir a variável, fá-lo-á com a palavra reservada var, o nome da variável e um valor, se desejar; ao definir a variável com var, não é necessário dar-lhe um valor inicial:

```
var contador;
var contador = 1;
```

Também pode definir várias variáveis numa única linha, separando-as por vírgulas:

```
var contador, cor, orientacao;
```

Neste caso, não foi definido o tipo de valor que a variável irá conter. Ele será atribuído na primeira vez que preencher a variável, ou seja, se o primeiro valor que der ao contador for numérico, essa variável será numérica.

Os tipos de dados são mais complexos de controlar, pois é possível criar variáveis sem lhes atribuir um tipo. Poderia dar-se o caso de querer multiplicar um dado numérico por outro que está em formato de string; neste caso, o JavaScript não falhará, mas dará um valor errado. Para resolver este problema, é possível alterar os tipos de dados das variáveis ou referir-se ao valor com o tipo desejado.

Estes são os métodos mais usados:

- Para converter um número numa string:
  - Adicione-o a um espaço em branco.

    número = 34;

    string = número + ' ';
  - Use o método string.

    número = 34;

    string = String (número);
- Para converter uma string num número:
  - parseInt, procura um número inteiro no início da string, ignorando as strings de texto.

    parseInt ("23 anos") = 23
  - parseFloat, funciona como parseInt, mas preserva as casas decimais e o sinal (positivo ou negativo).

    parseInt ("-23,2 quilos") = -23,2
- Evalé é a função mais complexa e procura qualquer expressão para convertê-la num número:

  x = 10;

  y = 20;

  z = 30;

  eval ("x + y + z + 900") = 960

Para os eventuais casos em que tem de saber que valor tem uma variável, existe uma função no JavaScript, designada typeof, que devolve o tipo de dados de uma variável específica. Segue-se um exemplo de como testar esta função:

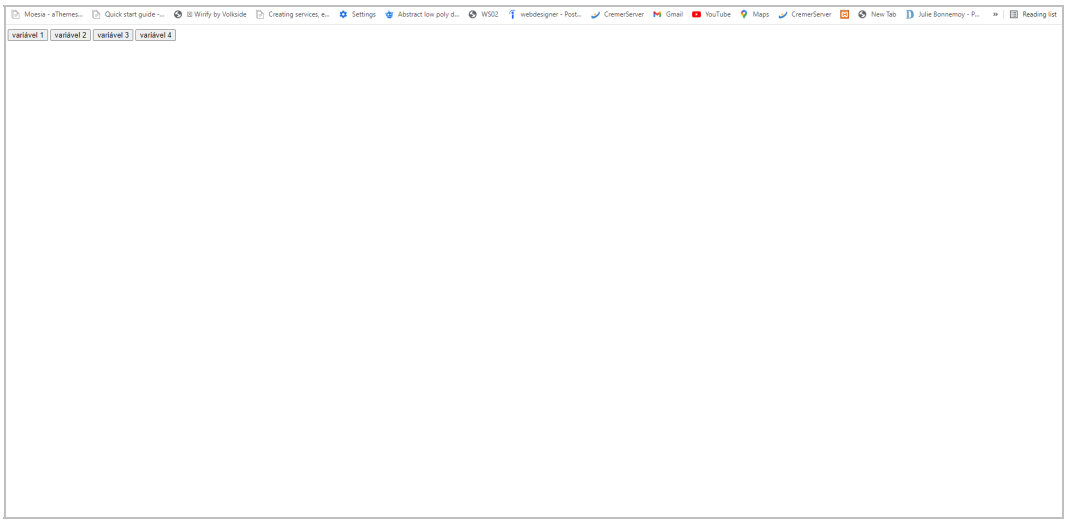
Exemplo 01

No exemplo a seguir existem quatro botões que, quando pressionados, mostram uma mensagem com o tipo de variável contida nas quatro variáveis diferentes.

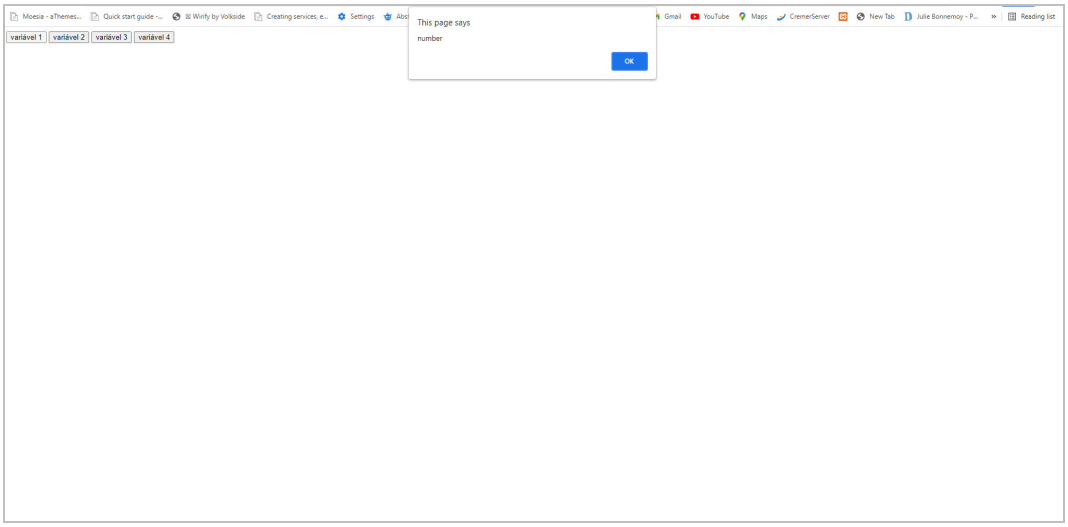
Exemplo:

```
<html>
<head></head>
<body>
<script type="text/javascript">
 var variable1 = 1;
 var variable2 = 'texto';
 var variable3 = true;
 var variable4 = newFunction();
</script>
<input type="button" onclick="alert(typeof variable1);"
    value="variável 1"/>
<input type="button" onclick="alert(typeof variable2);"
    value="variável 2"/>
<input type="button" onclick="alert(typeof variable3);"
    value="variável 3"/>
<input type="button" onclick="alert(typeof variable4);"
    value="variável 4"/>
</body>
</html>
```

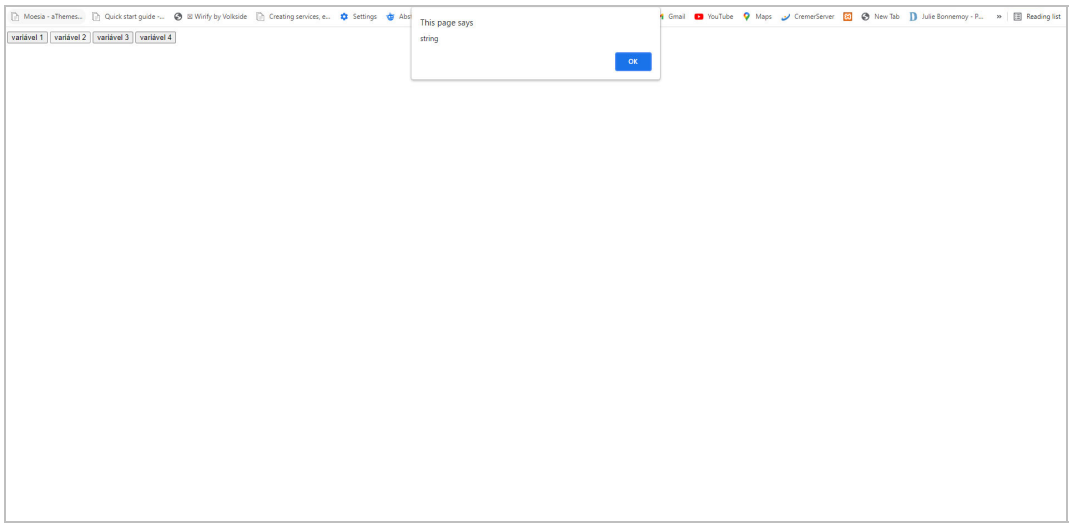
Se abrir a página recém-criada no navegador, verá os quatro botões e, ao clicar em cada um, irá aparecer um alert com o tipo de variável que contém.



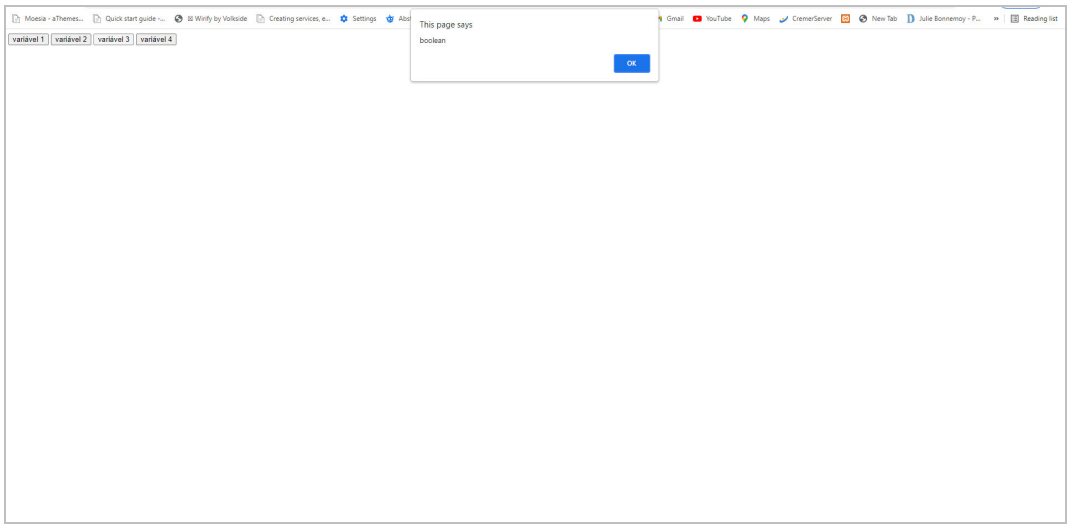
Resultado da visualização inicial do exemplo.



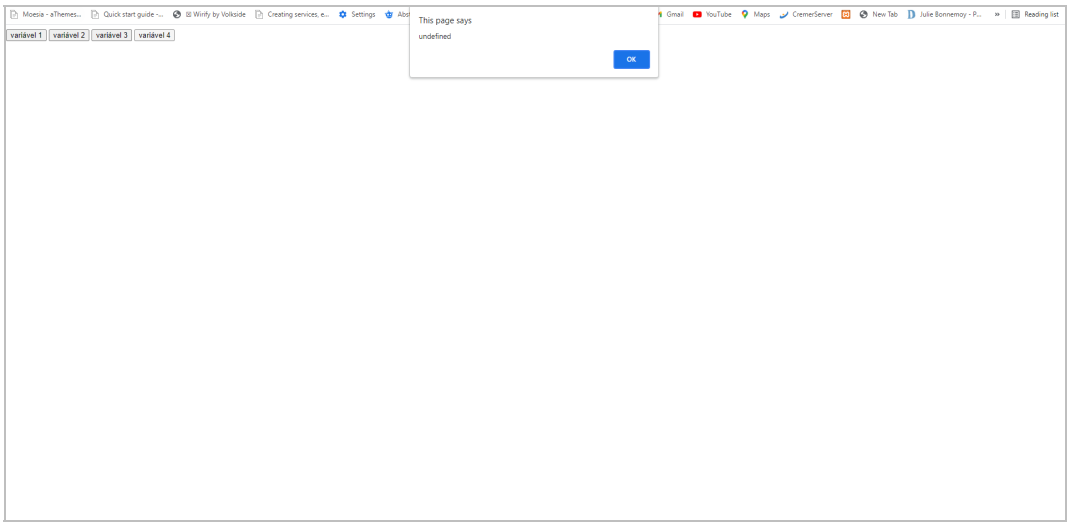
Resultado da visualização do exemplo ao clicar no botão “variável 1”.



Resultado da visualização do exemplo ao clicar no botão “variável 2”.



Resultado da visualização do exemplo ao clicar no botão “variável 3”.



Resultado da visualização do exemplo ao clicar no botão “variável 4”.

Uma característica importante das variáveis na zona onde podem ser usadas é a seguinte: por defeito, se criar uma variável em qualquer parte do código JavaScript, sem a definir como var, essa variável será global, porque o seu valor pode ser utilizado em qualquer parte ou função do código.

No entanto, ao definir uma variável como var, ela será uma variável local para a zona onde for definida, isto significa que, se a definir fora das funções que criou, a variável estará acessível (global); mas se a criar dentro de uma função, será local dentro da função e o seu valor não será acessível fora dessa função.

Exemplo 02

O exemplo a seguir mostra as diferenças entre uma variável global chamada “generica” e uma variável local chamada “interna”.

Exemplo:

```
<html>
<head></head>
<body >
<script type="text/javascript">
generica = 10;
```

```
function exemplo(){
var interna = 20;
}
alert('valor da generica = ' + generica);
alert('valor da interna = ' + interna);
</script>
</body>
</html>
```

Ao abrir este exemplo no navegador, este irá exibir o alert e mostrar o valor da variável "generica"; ao passar pela variável "interna" não vai lançar o alert, mas gerará, em vez disso, um erro, por estar a tentar aceder a uma variável não definida.

Resultado do exemplo.

3. TIPOS DE DADOS

A seguir, são apresentados os tipos de dados que pode definir:

- **Números inteiros**, dados numéricos, todos aqueles que se enquadram na definição de número; no caso de números decimais o separador é um ponto (.).

```
var variable = 33;
```

- **Strings**, dados de texto delimitados por aspas ou plicas.

```
var variable = ‘olá’;
```

- **Boolean**, dados cujo valor é verdadeiro (true) ou falso (false) e são escritos como palavras reservadas, sem aspas.

```
var varible = true;
```

- **Objetos**, dados mais complexos; são conjuntos de variáveis e funções definidas na linguagem ou criadas pelo utilizador, por exemplo, Array ou Image...

```
var variable = new Array(3);
```

- **Indefinidos**, aqueles dados que são declarados sem atribuir um tipo e que serão atribuídos quando receberem um valor.

```
var variable;>
```

4. ARRAYS E MATRIZES

Um array é um conjunto de elementos ordenados. Cada um desses elementos é uma variável que contém um valor.

A definição é simples:

```
var novoArray = new Array();
```

Entre os dois parênteses poderia indicar a quantidade de elementos que o array vai ter, mas tal não é necessário, pois se quiser um array com três elementos, basta preencher os três primeiros elementos do referido array para que este funcione (a partir do 0):

```
novoArray[0] = ‘amarelo’
novoArray[1] = ‘verde’
novoArray[2] = ‘azul’
```

Se quiser adicionar um quarto a este mesmo array de três elementos, será suficiente referir-se a esse elemento e preenchê-lo:

```
novoArray[3] = ‘vermelho’
```

Também pode criar um array diretamente, indicando os elementos que estarão contidos nele no construtor:

```
var novoArray = [‘carro’, ‘mota’, ‘barco’];
```

Para criar matrizes de dados em JavaScript não existe um tipo de dados ou uma estrutura pré-criada; basta fazer um loop para guardar uma variável do tipo array em cada elemento de um array normal. Por exemplo:

```
var matriz= new Array();
var elemento1 = new Array(3);
var elemento2 = new Array(3);
var elemento3 = new Array(3);

elemento1[0] = 4;
elemento1[1] = 7;
elemento1[2] = 2;

elemento2[0] = 1;
elemento2[1] = 4;
elemento2[2] = 3;

elemento3[0] = 8;
elemento3[1] = 9;
elemento3[2] = 9;

matriz[0] = elemento1;
matriz[1] = elemento2;
matriz[2] = elemento3;
```

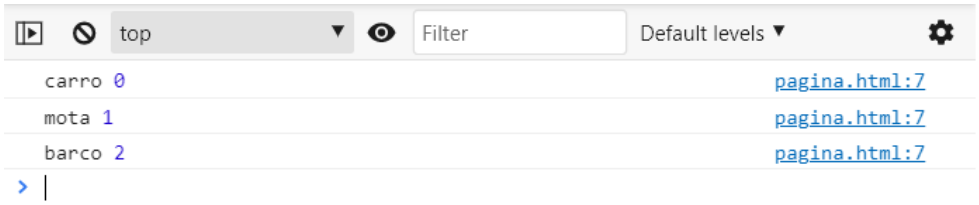
Com este código terá uma matriz de três linhas por três colunas.

Para trabalhar com arrays, existem algumas ferramentas de acesso em JavaScript; as mais comuns são:

- forEach: permite percorrer o array para obter dados ou comparar.

```
var novoArray = [‘carro’, ‘mota’, ‘barco’];
novoArray.forEach(function (elemento, indice, array){
    console.log(elemento, indice);
})
```

Este código percorre o array obtendo o valor do elemento e o índice correspondente.



Dica

console.log

No exemplo anterior, foi utilizada esta instrução que permite exibir valores de variáveis, textos, etc., na console do navegador. Para ver a console do navegador, clique no botão direito do rato e em “inspecionar” (geralmente com Ctrl + Shift + i).

- push: permite adicionar um elemento ao final de um array sem precisar de saber o índice para o atribuir.

```
novoArray.push('comboio');
//para ver o resultado na console
console.log(novoArray);
```

- pop: permite eliminar o elemento final do array.

```
novoArray.pop();
```

- shift: permite eliminar o elemento inicial do array.

```
novoArray.shift();
```

- unshift: permite adicionar o elemento no início do array.

```
novoArray.unshift('triciclo');
```

- indexOf: devolve o índice de um elemento num array. Se o elemento procurado não existir, devolverá-1.

```
novoArray.indexOf('mota'); //devolve 1 (os índices dos arrays
começam em 0)
```

- splice: permite eliminar um elemento específico, indicando o seu índice e a quantidade de elementos a eliminar.

```
novoArray.splice(1,1); //vai eliminar a 'mota'
```

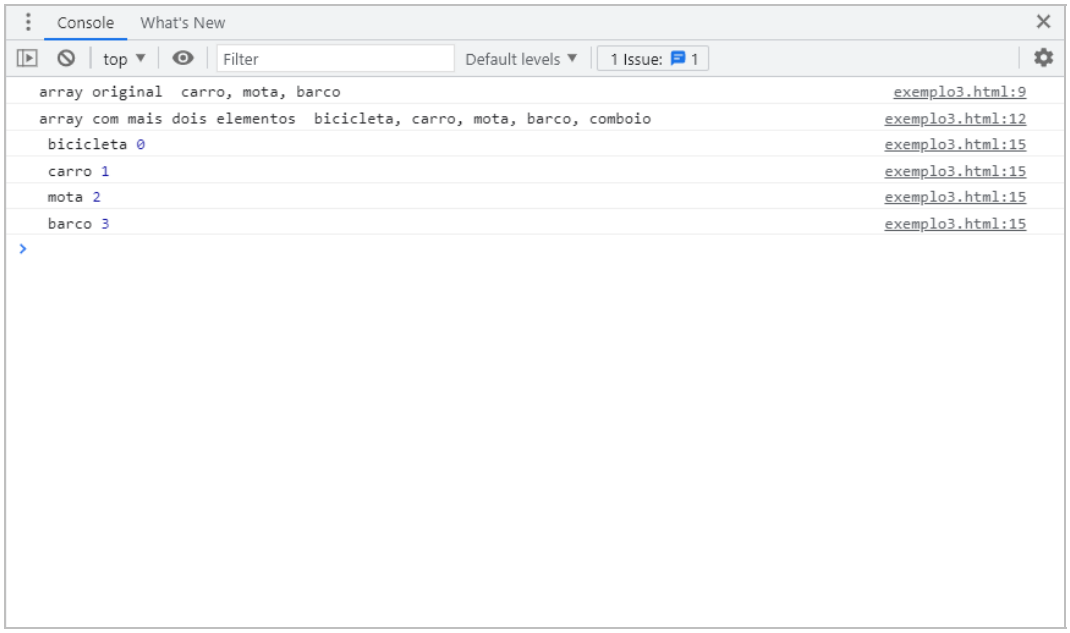
Exemplo 03

O exemplo a seguir mostra uma série de trabalhos realizados no mesmo array, exibindo os dados de saída no console do navegador.

Exemplo:

```
<html>

<head></head>

<body>

    <script type="text/javascript">
        var novoArray = [' carro', ' mota', ' barco'];
        console.log('array original ' + novoArray);
        novoArray.push(' comboio');
        novoArray.unshift(' bicicleta');
        console.log('array com mais dois elementos ' + novoArray);
        novoArray.splice(novoArray.indexOf('mota'), 1);
        novoArray.forEach(function (elemento, indice, array) {
            console.log(elemento, indice);
        });
    </script>

</body>

</html>
```

O resultado observado na console será:



5. UNDEFINED E EVAL

Existem dois usos muito típicos em JavaScript que fazem referência direta às variáveis.

Undefined

Como foi referido no início da unidade, quando uma variável não é definida, ela devolve o valor undefined. É muito comum ter de detetar se uma variável foi definida ou não, tanto ao criar as funções como ao fazer chamadas a scripts externos.

Uma variável é considerada definida quando lhe foi atribuído algum valor (mesmo que seja um valor vazio). Poderá verificar se está definida da seguinte forma:

```
var variavel;
if(variavel === undefined){
 alert('variavel indefinida!');
}else{
 alert('variavel definida!');
}
//mostrará a mensagem que está indefinida
```

Se na criação da variável for atribuído um valor, o exemplo anterior já dará indicação de que a variável está definida:

```
var variavel = 23;
```

Eval

Esta função permite avaliar um código JavaScript representado com uma string, ou seja, tratará a string como possível função ou elementos JavaScript. Segue-se um exemplo:

```
var a = 10;
var b = 20;
var soma = eval('a + b');
console.log(soma);
//irá mostrar o resultado da soma, 30.
```

Este também pode ser usado para executar funções com base no texto da string:

```
var texto = "alert";
eval(texto + '("olá")');
//este código mostrará um alert com o texto "olá".
```

O uso do eval é muito comum quando se pretende recolher, através do Ajax, dados que podem indicar a execução de uma ou outra função, ou passar comandos para executar funções de outras linguagens de programação, como por exemplo PHP.

CONCLUSÃO

As variáveis são um dos elementos mais importantes da programação JavaScript e é muito importante saber como defini-las e usá-las no contexto adequado.

Arrays e matrizes são tipos de variáveis que, embora menos utilizadas, são muito úteis para o JavaScript em interações com o utilizador, como por exemplo folhas de cálculo, listas de resultados, etc.

AUTOAVALIAÇÃO

1. **O que pode conter o nome de uma variável?**

   a) Apenas letras.

   b) Letras e números.

   c) Letras e números, desde que comece por uma letra.

   d) Letras e números, desde que comece por um número.

2. **Qual é a palavra reservada que usamos para definir variáveis?**

   a) Var.

   b) Variable.

   c) Eval.

   d) Boolean.

3. **Quais são as duas formas de converter um número em string?**

   a) Com uma string e cString.

   b) Com uma string e adicionando um espaço em branco.

   c) Apenas com uma string.

   d) Com parseStr.

4. Para que serve a função parseInt em JavaScript?

a) Converter uma string em número inteiro.

b) Converter uma string em texto.

c) Converter um número inteiro em float.

d) Converter um número em string.

5. Qual é a função que devolve o tipo de dados contidos numa variável?

a) varValue.

b) varType.

c) typeData.

d) typeOf.

6. Qual é a diferença entre uma variável local e uma variável global?

a) A variável local só pode conter valores numéricos.

b) A variável global só pode conter valores numéricos.

c) A variável local só pode ser usada na zona onde está definida, enquanto a variável global pode ser usada em toda a página.

d) A variável global está acessível apenas na zona onde está definida.

7. O que é utilizado para criar um array?

a) var vector = new Array();.

b) var array = newVector();.

c) var vector = Array();.

d) var vector = Array(número);.

8. O que indica o código “vector(1) = ‘azul’;”?

a) Que o primeiro valor do array é “azul”.

b) Que o array tem o nome de “azul” e terá apenas um elemento.

c) Que o primeiro valor do array terá a cor azul.

d) Que o segundo valor do array é “azul”.

9. Como é que se pode aumentar a capacidade de um array já existente?

a) Adicionando outro elemento ao próximo espaço vazio do array.

b) Redimensionando o array com Redim.

c) Redimensionando o array com "array = array + 1".

d) Não é possível aumentar a capacidade de um array já existente.

10. Como é que se pode criar uma matriz?

a) Usando “var matriz = matrix(num,num)”.

b) Usando um array preenchido com um loop de outros arrays.

c) Usando “var matriz = new Array(num,num,...)”.

d) Usando “var matriz = new Matrix();”.

SOLUÇÕES

1.	c	**2.**	a	**3.**	b	**4.**	a	**5.**	d
6.	c	**7.**	a	**8.**	d	**9.**	a	**10.**	b

PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

- Para saber mais sobre variáveis em geral e outras linguagens de programação, visite os seguintes sites:
  - https://pt.wikipedia.org/wiki/Vari%C3%A1vel_(programa%C3%A7%C3%A3o)
  - https://www.w3schools.com/js/js_variables.asp

BIBLIOGRAFIA

- VV. AA. (2010), *JavaScript*. Madrid: Anaya Multimedia.