

ESCOLA PROFISSIONAL DE TONDELA



Prof. Ricardo Jorge Santos

Novembro de 2004

INTRODUÇÃO AO JAVASCRIPT

O Que Necessita de Saber à Partida...

Antes de continuar, deve possuir conhecimentos básicos acerca dos seguintes assuntos:

- WWW, HTML e princípios básicos de construção de uma página *Web*
-

O Que é JavaScript?

- *JavaScript* é uma linguagem de *script* interpretada do lado do cliente (*client side script*)
- Uma linguagem de *script* é uma linguagem de programação leve e simples
- *JavaScript* é uma versão ligeira da linguagem de programação *Java* da *SUN Microsystems*
- *Javascript* foi criada para dar interactividade às páginas HTML, sendo integrada directamente na mesma e suportada pelos browsers usuais (como o *Netscape Navigator* e o *Internet Explorer*) sem ser necessário comprar ou adquirir qualquer tipo de licença adicional de utilização

Como Funciona?

Quando *JavaScript* é inserido num documento HTML, o Internet browser irá ler o HTML e interpretar o *JavaScript*. O *JavaScript* pode ser executado imediatamente, ou num acontecimento posterior (recorrendo à programação de procedimentos/funções despoletados por eventos).

Onde Inserir o JavaScript...

Secção Head

Os *scripts* podem ser colocados na secção *Head* da página. Normalmente é aqui que colocaremos toda a programação dos procedimentos/funções a serem chamados. Isto faz com que o *script* seja carregado antes de ser chamado do bloco principal da página (*Body*).

```
<html>
<head>
<script language="Javascript">
  ...
  instruções
  ...
</script>
</head>
```

Secção Body

Scripts que se encontram na secção *body* são executados enquanto a página carrega, pela ordem de sequência das instruções da própria secção.

```
<html>
<head>
</head>
<body>
<script language="Javascript">
  ...
```

```

    instruções
    ...
</script>
</body>

```

Podemos colocar um número ilimitado de *scripts* no documento, podendo ter *scripts* tanto na secção *body* como na secção *head*, juntos.

```

<html>
<head>
<script language="Javascript">
    ...
    instruções
    ...
</script>
</head>
<body>
<script language="Javascript">
    ...
    instruções
    ...
</script>
</body>

```

Podemos ainda criar um documento externo com o código de *Javascript* que queremos utilizar em mais do que um documento HTML e importá-lo através da instrução-etiqueta *<script>*. Isto permite colocar toda a programação *Javascript* comum a vários documentos HTML num único documento, sem necessitar de o duplicar desnecessariamente (um pouco à semelhança do que acontece com as CSS – *Cascading Style Sheets*).

```

<html>
<head>
<script language="Javascript" src="ficheiro.js"></script>
</head>
<body>
    ...
    Instruções HTML
    ...
</body>

```

VARIÁVEIS NO JAVASCRIPT

O Que é uma Variável?

Uma variável é um "contentor" de informação que se pretende guardar. O valor de uma variável pode ser modificado durante na sequência de instruções que compõem o *script*. Podemos referenciar uma variável pelo nome para consultar o seu valor ou alterá-lo. No *JavaScript*, as variáveis são criadas automaticamente pela simples atribuição de valores à mesma, definindo desta forma o seu tipo de dados.

Regras para os Nomes das Variáveis:

- Têm obrigatoriamente de começar por uma letra
 - Não pode ser utilizado o ponto final (.)
 - Não podem exceder 255 caracteres
 - Em *Javascript* as variáveis são *case-sensitive*, o que significa que existe distinção entre caracteres maiúsculos e minúsculos. *X* e *x* são consideradas duas variáveis distintas
-

Declaração de Variáveis

Variáveis podem ser declaradas através da atribuição directa de valores. Exemplo:

```
var nome = "Jaime"
```

Acabou-se de criar uma variável. O nome da variável é *nome*.

Também podem ser criadas variáveis sem as declarar com *var*. Exemplo:

```
nome = "Jaime"
```

Agora também foi criada uma variável. O nome da variável é *nome*. O tipo da variável criada é definido automaticamente através do primeiro conteúdo que se lhe atribui. No caso dos dois exemplos anteriores, o tipo de variável passa automaticamente a ser *string*, uma vez que se lhes atribuiu inicialmente um valor deste tipo aquando da sua criação.

Atribuindo Valores a Variáveis

Podemos atribuir valores às variáveis da seguinte forma:

```
nome = "Hege"  
i = 200
```

O nome da variável fica à esquerda da expressão e o valor que pretendemos que ela assuma fica à direita. Agora a variável *nome* tem o valor "Hege".

Tempo de Vida das Variáveis

Quando se declara uma variável dentro de uma sub-rotina, ela só pode ser acedida dentro dessa sub-rotina. Quando se sai da sub-rotina, a variável é destruída. Estas variáveis são apelidadas de variáveis locais. Podemos ter variáveis locais com o mesmo nome em sub-rotinas diferentes, uma vez que elas só são reconhecidas dentro da própria sub-rotina em que são declaradas.

Se declararmos uma variável fora de uma sub-rotina, todas as sub-rotinas poderão aceder ao seu valor. Estas variáveis são apelidadas de variáveis globais. Apenas serão destruídas quando a página que lhes deu origem for fechada.

Variáveis Vector (Array)

Por vezes queremos guardar mais do que um valor numa só variável. Nestes casos podemos criar uma variável que contenha um conjunto de valores. Este tipo de variável é chamado de vector (*array*). A declaração de uma variável *array* utiliza parentesis () a seguir ao nome da variável. Na instrução seguinte, é definido um vector com espaço para armazenar 3 elementos:

```
var nomes=new Array(3)
```

O número dentro dos parentesis é 3. Começando a numeração dos elementos em zero, implica que o vector contém 3 elementos. Este é um vector de tamanho fixo. Podemos assumir valores para cada elemento da seguinte forma:

```
nomes[0]="Tove"  
nomes[1]="Jani"  
nomes[2]="Stale"
```

De forma semelhante, os valores guardados podem ser consultados/acedidos indicando o nome da variável e a posição do elemento que se pretende. Por exemplo, se quisermos aceder ao primeiro valor guardado no vector podemos fazê-lo da seguinte forma:

```
mother=nomes[0]
```

Para criarmos o mesmo vector e preenchê-lo de valores iniciais de uma só vez, podemos fazê-lo da seguinte forma:

```
var nomes=new Array("Tove","Jani","Stale")
```

Para obter uma matriz (vector de 2 dimensões) com 5 linhas e 7 colunas:

```
tabela=new Array(5)  
tabela[0]=new Array(7)  
tabela[1]=new Array(7)  
tabela[2]=new Array(7)  
tabela[3]=new Array(7)  
tabela[4]=new Array(7)
```

Para aceder ao elemento da 3.^a linha, 5.^a coluna podemos fazê-lo da seguinte forma:

```
coisa=tabela[2][4]
```

OPERADORES DO JAVASCRIPT

Aritméticos		Comparação		Lógicos	
Descrição	Símbolo	Descrição	Símbolo	Descrição	Símbolo
Multiplicação	*	Igualdade	==	Negação Lógica	!
Divisão	/	Desigualdade	!=	Conjunção Lógica	&&
Módulus aritmético	%	Menor que	<	Disjunção Lógica	
Adição	+	Maior que	>		
Subtração	-	Menor ou igual a	<=		
Junção de <i>Strings</i>	+	Maior ou igual a	>=		

SUB-ROTINAS NO JAVASCRIPT

Sub-Rotinas do JavaScript

Existe apenas um tipo de sub-rotina: A função *function*.

Uma sub-rotina *function*:

- é um conjunto de instruções, delimitado pelas instruções *function { e }*
- pode desempenhar acções e **pode ou não devolver** um valor como resultado
- pode ter parâmetros de entrada ou não
- se não tiver parâmetros de entrada, deve incluir um par de parentesis vazios ()
- se quisermos devolver algum valor como resultado da função devemos utilizar a instrução *return* seguido da variável ou expressão cujo valor queremos dar como resultado

```
function myfunction() {
  ...
  Instruções
  ...
  return valor
}

ou

function myfunction(argument1, argument2) {
  ...
  Instruções
  ...
  return valor
}
```

Chamando sub-rotinas function

Quando queremos chamar uma *function*, podemos fazê-lo da seguinte forma:

```
name = findname()
```

Aqui chamamos uma *function* denominada *findname*, que devolve um valor que será guardado na variável *name*.

Ou podemos fazê-lo da seguinte forma:

```
alert ("O seu nome é " + findname())
```

Aqui também chamamos uma *function* denominada *findname*, que retorna um valor que será mostrado numa caixa de mensagem.

ESTRUTURAS DE CONTROLE NO JAVASCRIPT

Estruturas Condicionais

Quando desenvolvemos aplicações, por vezes queremos executar acções diferentes de acordo com decisões condicionais, com estruturas condicionais na programação para o fazer.

Em *JavaScript* temos duas instruções condicionais principais:

- **if (...)** { ... } **else** { ... } – utilizamos esta estrutura de controlo quando desejamos apenas executar um de dois conjuntos de comandos consoante determinada situação é verdadeira ou falsa
- **switch** – utilizamos esta estrutura de controlo para executar determinados conjuntos de comandos de acordo com determinadas hipóteses de acontecimento

if (...) { ... } else { ... }

Devemos utilizar a estrutura de controlo *if (...) { ... } else { ... }* se desejarmos:

- executar determinadas instruções caso determinada condição seja verdadeira
- seleccionar um de dois blocos de instruções a executar

Se quisermos executar apenas **uma** instrução quando uma condição se verifica, podemos escrever a instrução como a seguinte:

```
if (i==10) {  
    document.write("Olá")  
}
```

A instrução acima não tem *..else..* na sua sintaxe. Indicamos apenas que queremos executar **uma acção** (escrever o texto *olá* numa janela de mensagem) caso a condição seja verdadeira (neste caso, se *i* for igual a 10).

Se desejarmos executar uma instrução ou um conjunto de instruções se uma condição for verdadeira e outra instrução ou conjunto de instruções caso a mesma condição for falsa, temos de adicionar o comando *else*:

```
if (i==10) {  
    alert ("Olá");  
} else {  
    alert ("Adeus");  
}
```

O primeiro bloco de instruções será executado se a condição for verdadeira (se *i* for igual a 10) e o outro bloco de instruções será executado em caso contrário (se *i* não for igual a 10).

switch

Podemos utilizar o comando **switch** caso queiramos seleccionar um entre vários conjuntos de instruções a executar:

```
switch (Pagamento) {  
    case "Dinheiro":  
        alert ("Você vai pagar em dinheiro")  
        break  
    case "MB":  
        alert ("Você vai pagar pelo multibanco")  
        break  
    case "VISA":  
        alert ("Você vai pagar com cartão de crédito")  
        break  
    default  
        alert ("Forma de pagamento inválida")  
}
```

Como funciona: Primeiro temos a expressão a avaliar, normalmente uma variável (primeira linha de instruções – *switch (Pagamento)*), que é avaliada. De acordo com o valor da expressão serão executadas as respectivas instruções definidas dentro dos blocos *case* que se referiram a esse mesmo valor. Se não tiver nenhum valor correspondente, será executado o bloco de instruções colocado no comando *default*.

No exemplo mostrado, caso a variável *Pagamento* tenha o valor "MB", por exemplo, será executada a instrução *alert ("Você vai pagar pelo multibanco")*.

ESTRUTURAS DE CICLO EM JAVASCRIPT

Instruções de Ciclo

Muitas vezes ao programar necessitamos de repetir o mesmo bloco de instruções determinado número de vezes. Neste casos, devem ser utilizadas estruturas de ciclo.

Em *JavaScript* existem duas instruções que permitem definir ciclos:

- **for (...)** { ... } **statement** – executa um conjunto de instruções determinado número de vezes
- **while (...)** { ... } – executa repetidamente um conjunto de instruções enquanto determinada condição é verdadeira, testando essa condição no início de cada execução do bloco de instruções
- **do { ... } while (...)** – executa repetidamente um conjunto de instruções enquanto determinada condição é verdadeira, testando essa condição no fim de cada execução do bloco de instruções

Ciclo for (Inicialização; Condição; Incremento) { ... }

Podemos utilizar um ciclo **for (Inicialização; Condição; Incremento) { ... }** para repetir um bloco de instruções, quando sabemos quantas repetições queremos.

Utilizamos uma variável contadora que é incrementada ou decrementada em cada passo de execução do ciclo, como no exemplo seguinte:

```
for (i=1; i<=6; i=i+1) {
  ...
  Instruções
  ...
}
```

A inicialização da instrução **for** especifica a variável contadora (**i**) e o seu valor inicial. Neste exemplo, a variável contadora (**i**) começa por ter um valor inicial igual a 1, sendo incrementado de 1 para cada passo de execução do ciclo, enquanto não atingir um valor igual a 6. Desta forma, a variável (**i**) assumirá os valores 1, 2, 3, 4, 5 e 6 para o 1.º, 2.º, 3.º, 4.º, 5.º e 6.º passo de execução do ciclo, respectivamente.

No exemplo seguinte, a variável contadora (**i**) é incrementada de dois em dois a cada passo de execução do ciclo:

```
for (i=1; i<=11; i=i+2) {
  ...
  Instruções
  ...
}
```

Neste caso, a variável tomará os valores 1, 3, 5, 7, 9 e 11. No exemplo seguinte, a variável contadora (**i**) é decrementada em dois de cada passo de execução do ciclo:

```
for (i=11; i>=1; i=i-2) {
  ...
  Instruções
  ...
}
```

Sair de um for (...)

Podemos sair antecipadamente de um ciclo **for (...)** utilizando o comando *break*.

Ciclos `while (...)` `{ ... }` e `do { ... } while (...)`

Podemos utilizar os comandos envolvendo **while** para executar um bloco de instruções quando não sabemos o número de repetições que precisamos. O conjunto de instruções é repetido enquanto a condição definida for verdadeira.

Repetir Instruções enquanto uma Condição seja Verdadeira

O exemplo seguinte utiliza o comando *while* para testar uma condição num comando *while (...)* `{ ... }`.

```
while (i>10) {  
  ...  
  Instruções  
  ...  
}
```

Se inicialmente *i* for igual a 9, as instruções dentro do ciclo nunca serão executadas.

```
do {  
  ...  
  Instruções  
  ...  
} while (i>10)
```

As instruções dentro deste ciclo serão executadas pelo menos uma vez, mesmo que *i* seja menor que 10.

Saltar fora de um ciclo `while`

Podemos saltar fora de um ciclo *while* com o comando *break*.

```
while (i>10) {  
  i=i-1  
  if (i<10) { break }  
}
```

O código dentro deste ciclo será executado enquanto *i* for diferente de 10, e enquanto *i* for maior que 10.

OBJECTOS, PROPRIEDADES E MÉTODOS JAVASCRIPT

OBJECTO Array

Para criar um vector:

```
var myArray = new Array()  
var myArray = new Array(n.º elementos)  
var myArray = new Array(lista de valores separados por vírgulas)
```

Propriedade: vector.length

Indica o número de elementos máximo definidos para o vector.

Exemplo

```
alunos=new Array(30)  
document.write(alunos.length)  
Output:  
30
```

Método: vector1.concat(vector2)

Retorna a união de dois vectores como um vector só.

Exemplo

```
alunos=new Array(30)  
alunos2=new Array(30)  
alunos3=alunos.concat(alunos2)
```

No exemplo o vector *alunos3* contém todos os elementos do vector *alunos* mais os elementos do vector *alunos2*.

Método: vector.join(separador)

Retorna uma string contendo todos os elementos de um vector separados por um caracter separador.

Exemplo

```
alunos=new Array("Ana", "Manuel", "Sandra", "Zé")  
notas=new Array(10, 8, 20, 12)  
document.write(alunos.join(";"))  
document.write(notas.join(";"))  
Output:  
Ana;Manuel;Sandra;Zé  
10;8;20;12
```

Método: vector.reverse()

Troca por ordem inversa à existente os elementos de um vector, devolvendo esse vector com os seus elementos invertidos.

Exemplo

```
alunos=new Array("Ana", "Manuel", "Sandra", "Zé")
novosalunos=alunos.reverse()
```

Neste caso, o vector *novosalunos* ficaria com:

```
novosalunos[0] = "Zé"
novosalunos[1] = "Sandra"
novosalunos[2] = "Manuel"
novosalunos[3] = "Ana"
```

Método: vector.slice(posição inicial, posição final)

Devolve um vector contendo uma parte de outro vector.

Exemplo

```
alunos=new Array("Ana", "Manuel", "Sandra", "Zé")
novosalunos=alunos.slice(1,2)
```

Neste caso, o vector *novosalunos* ficaria com 2 elementos, retirados a partir do 2.º elemento do vector *alunos* até ao seu 3.ª elemento:

```
novosalunos[0] = "Manuel"
novosalunos[1] = "Sandra"
```

OBJECTO Boolean

Para criar um objecto booleano:

```
var myBool = new Boolean()
var myBool = new Boolean(valor booleano)
var myBool = valor booleano
```

Método: variável.toString()

Devolve o valor booleano da variável como uma string.

Exemplo

```
alunos=new Boolean(true)
alunos2=alunos.toString()
```

O conteúdo de *alunos2* será "true"

OBJECTO Date

Para criar um objecto de data:

```
var myDate = new Date()  
var myDate = new Date("mês dd, aaaa hh:mm:ss")  
var myDate = new Date("mês dd, aaaa")  
var myDate = new Date(aa, mm, dd, hh, mm, ss)  
var myDate = new Date(aa, mm, dd)
```

Este objecto representa o número de milisegundos desde a meia-noite do dia 1 de Janeiro de 1970.

Método: variável.getDate()

Devolve o dia do mês da data actual da variável.

Exemplo

```
alunos=new Date()  
document.write(alunos)  
document.write(alunos.getDate())  
Output:  
19-01-2005  
19
```

Método: variável.getDay()

Devolve o dia da semana da data actual da variável (de 0 a 7: 0-Domingo até 6-Sábado).

Método: variável.getFullYear()

Devolve o ano da data actual da variável.

Método: variável.getHours()

Devolve a hora (de 0 a 23) da data actual da variável.

Método: variável.getMinutes()

Devolve o minuto (de 0 a 59) da data actual da variável.

Método: variável.getMonth()

Devolve o mês da data actual da variável (de 0 a 11: 0-Janeiro até 11-Dezembro).

Método: variável.getSeconds()

Devolve o segundo (de 0 a 59) da data actual da variável.

Da mesma forma, temos métodos análogos para modificar os valores das variáveis data.

São eles: `setDay(valor)`, `setFullYear(valor)`, `setHours(valor)`, `setMinutes(valor)`, `setMonth(valor)` e `setSeconds(valor)`.

Método: variável.toString()

Devolve o valor data da variável como uma string.

OBJECTO Math

Este objecto representa uma biblioteca de constantes e funções matemáticas.

As constantes mais significativas são o e (`Math.E`) e o π (`Math.PI`).

Exemplo

```
var valorE=Math.E
var valorPI=Math.PI
document.write(valorE)
document.write(valorPI)
Output:
2.718281828459045
3.141592653589793
```

Lista das Funções Matemáticas mais Significativas

Função	Descrição	Sintaxe
abs	O valor absoluto de um número	Math.abs(valor)
acos	O valor do arco co-seno em radianos para um [-1; 1]	Math.acos(valor)
asin	O valor do arco seno em radianos para um [-1; 1]	Math.asin(valor)
atan	O valor do arco tangente em radianos para [-oo; +oo]	Math.atan(valor)
cos	O valor do coseno de um número	Math.cos(valor)
log	O valor do logaritmo na base E de um número	Math.log(valor)
pow	O valor da potência de A elevado a B	Math.pow(A,B)
random	Um valor aleatório entre [0; 1[Math.random()
round	O valor de um número arredondado para inteiro	Math.round(valor)
sin	O valor do seno de um número	Math.sin(valor)
sqrt	O valor da raiz quadrada de um número	Math.sqrt(valor)
tan	O valor da tangente de um número	Math.tan(valor)

OBJECTO Number

Para criar um objecto numérico:

```
var myNum = valor  
var myNum = new Number(valor)
```

Método: variável.toString()

Devolve o valor numérico da variável como uma string.

Exemplo

```
alunos=25  
alunos2=alunos.toString()  
  
O conteúdo de alunos2 será "25"
```

OBJECTO String

Para criar uma string:

```
var myString = "Hoje está um dia bonito."  
var myString = new String("Hoje está um dia bonito.")
```

Propriedade: variável.length

Indica o número de caracteres que compõem a string contida na *variável*.

Exemplo

```
alunos="José Manel"  
document.write(alunos.length)  
Output:  
10
```

Método: variável.charAt(posição)

Devolve o caracter existente na *posição* dentro da string contida na *variável*.

Exemplo

```
alunos="José Manel"  
document.write(alunos.charAt(3))  
Output:  
é
```

Método: variável.charCodeAt(posição)

Devolve o valor do código ASCII correspondente a o caracter existente na *posição* dentro da string contida na *variável*.

Método: variável.indexOf(string a procurar, posição inicial)

Retorna a posição da primeira ocorrência da *string a procurar* dentro da string contida na *variável*, procurando a partir da *posição inicial*. Se não encontrar nenhuma ocorrência devolve como resultado -1.

Exemplo

```
alunos= "José Manel"  
document.write(alunos.indexOf("an",0))  
Output:  
6
```

Método: variável.split(caracter delimitador)

Devolve um vector de strings usando o *caracter delimitador* como separador dos elementos a partir da string contida na *variável*.

Exemplo

```
alunos="12;32;90;30"  
var novosalunos=alunos.split(";")  
  
Neste caso, o vector novosalunos ficaria com:  
  
novosalunos[0] = "12"  
novosalunos[1] = "32"  
novosalunos[2] = "90"  
novosalunos[3] = "30"
```

Método: variável.substr(posição inicial, n.º caracteres)

Devolve parte da string contida na *variável*, com o n.º caracteres contidos a partir da *posição inicial*.

Exemplo

```
alunos="José Manel"  
document.write(alunos.substr(5,4))  
Output:  
Mane
```

Método: variável.toLowerCase()

Devolve a string contida na *variável*, toda com caracteres transformados em minúsculas.

Método: variável.toUpperCase()

Devolve a string contida na *variável*, toda com caracteres transformados em maiúsculas.

Método: variável.parseInt()

Devolve um valor inteiro convertendo a string contida na *variável*.

Método: variável.parseFloat()

Devolve um valor real convertendo a string contida na *variável*.

OBJECTO DOM – Document Object Model

O DOM foi criado para proporcionar aos scripts HTML a manipulação genérica das propriedades e métodos de qualquer objecto HTML através de um modelo comum.

A sua sintaxe é:

```
document.all.getElementById("nome do objecto HTML").propriedade=valor
```

ou

```
document.all.getElementById("nome do objecto HTML").método
```

Supondo que no documento existe um objecto parágrafo HTML a quem se deu um ID de "Texto" (<P ID="Texto">) e que queremos alterar a cor da letra do conteúdo desse parágrafo para azul, poderemos fazê-lo em *Javascript* utilizando:

```
document.all.getElementById("Texto").style.color="#0000FF"
```

Os nomes de cada propriedade e método para cada objecto HTML devem ser consultados em bibliografia específica referente ao DOM – Document Object Model.

EXEMPLO PRÁTICO UTILIZANDO JAVASCRIPT

O seguinte exemplo consiste numa listagem de um ficheiro HTML incorporando a utilização de rotinas em *JavaScript* para introduzir uma frase e uma letra a procurar nessa frase, dando como resultado o número de vezes que essa letra existe dentro da frase.

```
<HTML>

<HEAD>
<script language="Javascript">
var i=0

function Alargar() {
document.getElementById("frase").size=document.getElementById("frase").size+1
}

function CoresFundo() {
    i=i+1;
    if (i>6) i=1;
    if (i==1) document.bgColor="yellow";
    if (i==2) document.bgColor="Red";
    if (i==3) document.bgColor="Green";
    if (i==4) document.bgColor="Orange";
    if (i==5) document.bgColor="Blue";
    if (i==6) document.bgColor="Navy";
}

function Carrosel() {
var j;
    for (i=1;i<=6;i=i+1) {
        if (i==1) document.bgColor="Yellow";
        if (i==2) document.bgColor="Red";
        if (i==3) document.bgColor="Green";
        if (i==4) document.bgColor="Orange";
        if (i==5) document.bgColor="Navy";
        if (i==6) document.bgColor="Blue";
        for (j=1;j<=100000;j=j+1) {}
    }
}

function Calcular() {
    var Conta=0;
    var Frase=new String(document.getElementById("frase").value);
    var Letra=new String(document.getElementById("letra").value);
    for (j=0;j<=Frase.length-1;j=j+1) {
        if (Frase.charAt(j)==Letra) { Conta=Conta+1 }
    }
    document.getElementById("resposta").innerText="Existem "+Conta.toString()+"
letras iguais na frase.";
}

function Limpar() {
    document.all.resposta.innerText=""
}

function ActualizaRelogio() {
    relo=setTimeout("ActualizaRelogio()",1000)
    Hr=new Date()
    hh=Hr.getHours()
    mm=Hr.getMinutes()
    ss=Hr.getSeconds()
    document.all.Relogio.innerText="Hora Actual >> "+hh+": "+mm+": "+ss
}

</script>
```

```

</HEAD>

<BODY>

Introduza uma frase:
<input id="frase" type="text" size=30 onKeyDown="javascript:Limpar()"><br>

Introduza a letra a procurar:
<input id="letra" type="text" size=1 onKeyDown="javascript:Limpar()"><br>

<P id="resposta">&nbsp;</P><br>

<input type="button" value="CALCULAR" onClick="javascript:Calcular()">
<input type="button" value="ALARGAR" onClick="javascript:Alargar()">
<input type="button" value="FUNDO" onClick="javascript:CoresFundo()">
<input type="button" value="CARROSEL" onClick="javascript:Carrosel()">
<input type="button" value="VERDE"
onClick="javascript:document.getElementById('frase').style.color='green'">
<input type="button" value="VERMELHO"
onClick="javascript:document.getElementById('frase').style.color='red'">
<input type="button" value="PRETO"
onClick="javascript:document.getElementById('frase').style.color='black'">

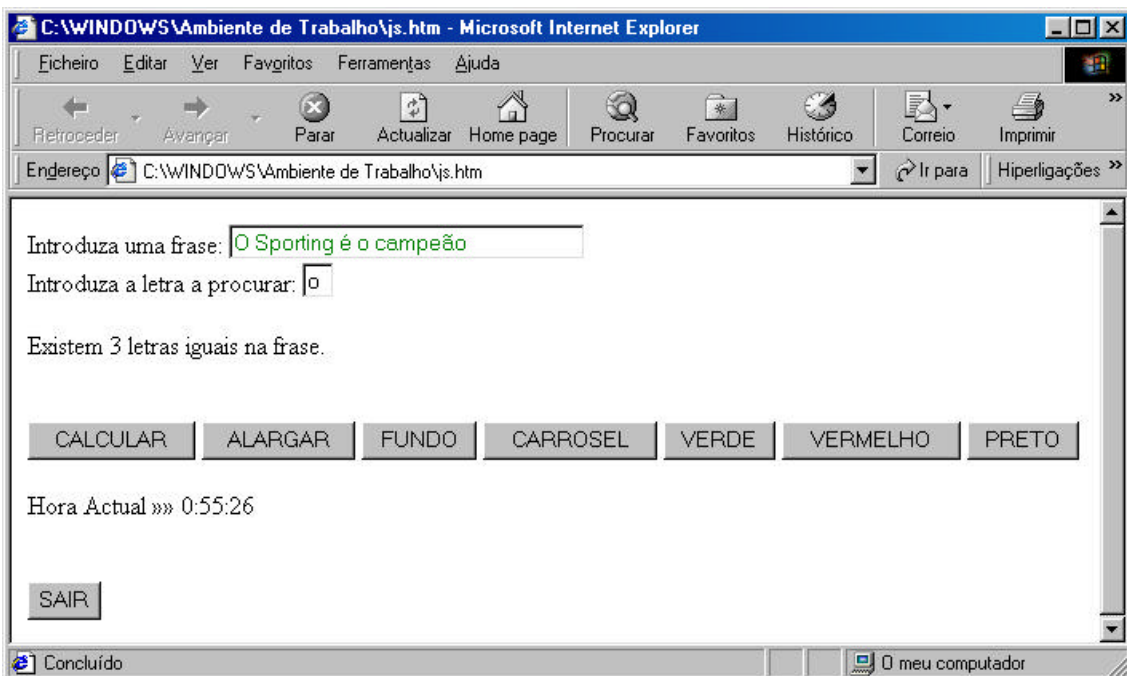
<P id="Relogio">
<script language="Javascript">
  ActualizaRelogio()
</script>
</P><br>

<input type="button" value="SAIR" onClick="javascript>window.close()">

</BODY>

</HTML>

```



Observações: Criaram-se dois objectos *INPUT* do HTML para que o utilizador introduza a frase e a letra. Foi criado um objecto parágrafo (*<P>*) para mostrar o resultado do cálculo da quantidade de repetições da letra na frase.

Referências: www.w3schools.com