Universidade do Minho
Escola de Engenharia

Relatório:

M.R.P. em Javascript

Disciplina: Linguagens de Programação
Curso: Engenharia e Gestão Industrial

Carolina Fernandes N.º40532
Guilherme Bacinello N.º42119
Diana Rocha N.º40536

Guimarães, 2004

Índice

Introdução

O que é o Javascript?

Javascript é uma linguagem criada pela Netscape que serve basicamente para dar mais recursos ao navegador de internet.

Todas as páginas escritas com HTML são estáticas, conseguem fazer com que o browser apenas leia o que há no código e reproduza aquele conjunto de instruções. Um dos principais recursos do Javascript é a possibilidade de fazer com que a página HTML seja dinâmica, ou seja, capaz de fazer com que o usuário possa interagir com a página, ou usar outros recursos para fazer com que a página se altere sem necessidade de um novo carregamento.

Esta linguagem de programação que funciona interativamente com o código HTML. Consegue ler, entender e manipular os objectos de uma página HTML, tendo a facilidade necessária para alterá-los automaticamente, quando programado. São inseridos nas páginas de uma maneira peculiar, podendo ter apenas uma área para scripts ou espalhados pelo código, do jeito que melhor se adapte às necessidades de cada página/site.

Resumidamente, o Javascript é uma poderosa ferramenta para quem quer fazer páginas de internet mais bonitas e com recursos a mais. Com esta linguagem é possível, por exemplo, colocar saudações ao visitante, como Bom dia/ tarde/ noite, a data e hora actual (existente no computador do usuário), janelas popup, cálculos e uma infinidade de recursos.

Diferença entre Javascript e Java

Apesar da semelhança entre os nomes, Java e JavaScript são duas técnicas diferentes de programação na Internet. Java é uma linguagem de programação, JavaScript é uma linguagem de hiper-texto. A linguagem Java (desenvolvida pela Sun), tem muitas semelhanças com a linguagem C e C++, utilizada para programação. Já o Javascript é uma linguagem dinâmica, orientada aos objectos constantes do HTML, sendo bastante simples e prática.

A programação em Javascript

Em documentos HTML, a utilização da linguagem JavaScript dá-se sob a forma de funções (applets) que são chamadas em determinadas situações ou em resposta a determinados eventos. Estas funções podem estar localizadas em qualquer parte do código HTML, a única restrição é que devem começar com a declaração <SCRIPT> e terminar com o respectivo </SCRIPT>. Por convenção costuma-se colocar todas as funções no início do documento, entre as TAGs <HEAD> e </HEAD>, isso para garantir que o código JavaScript seja carregado antes que o usuário interaja com a Home Page, ou seja, antes do <BODY>.

Eis alguns conceitos básicos para a programação em Javascript:

Variáveis: São valores que são atribuídos dinamicamente, ou seja, não têm um valor pré-definido. Podem ser definidos pelo usuário, bem como por qualquer outro método.

Funções: São comandos ou conjuntos de comandos utilizados para cumprir certos objectivos dentro do script/roteiro de programação. Existem dois tipos de funções em Javascript: As *funções internas (ou intrínsecas)*, e as *funções externas (ou são definidas pelo usuário)*:

- Funções internas ou intrínsecas: são aquelas que não precisam ser especificadas, pois o próprio Javascript já as reconhece e sabe o que fazer com elas.

- Funções externas ou definidas pelo usuário: são aquelas que o usuário precisa explicar para o Javascript quais são as acções que ele precisa tomar, pois a linguagem desconhece seu nome.

Objectos: São os componentes de uma página HTML. É um dos tópicos mais extensos do Javascript. Atente na seguinte visão figurativa: podemos ilustrar uma mesa de computador como uma página HTML. O seu objecto maior é a mesa de computador em si, tendo como um de seus objectos o computador que está sobre ela. O computador tem como seu objecto o teclado, o monitor,

o rato. E por sua vez, o rato tem os botões como seus objectos, enquanto o teclado tem as teclas, e assim temos uma cadeia de objectos hierarquicamente formada. Podemos dizer que as teclas pertencem ao teclado que pertencem ao computador que pertence à mesa de computador.

Eventos: São importantes aliados ao programador de Javascript. O evento, como o próprio nome diz, é algo que acontece. É como se o Javascript aguardasse pelo acontecimento do evento para accionar algum procedimento. Por exemplo, o clique sobre algum objecto da página é um evento. Passar o rato sobre uma figura é um evento. Carregar e descarregar uma página é um evento. Selecionar um campo de formulário é um evento. Assim temos total controle sobre as acções exercidas pelo usuário podendo criar diversas situações.

Resultados

Código Fonte do Trabalho

```
<HTML>
<head>
<script language="javascript">


function calculo(form)
{
 b2=eval(form.b2.value);
 b5=100;
 b4=eval(form.b4.value);
 b6=b2+b5;
 form.b6.value=b6;
 b7=b2-b4+b5;

 form.b7.value=b7;

 b6=eval(form.b6.value);
 b8=b7-b6;
 form.b8.value=b8;
 c4=eval(form.c4.value);
 c5=150;
 c2=eval(form.c2.value);
 if(b7-c4+c5<c2)
   {
    c7=c2;
   }
  else
   {
   c7=b7-c4+c5;
   }
 form.c7.value=c7;
 c6=b7-c4+c5;
 form.c6.value=c6;
 c8=c7-c6;

 form.c8.value=c8;
 d4=eval(form.d4.value);
 d5=210;
 d6=c7-d4+d5

 form.d6.value=d6;
 if(c7-d4+d5<c2)
   {
   d7=c2;
   }
    else
 {
   d7=c7-d4+d5;
   }
 form.d7.value=d7;
 form.b9.value=c8;
 d8=d7-d6;
 form.c9.value=d8;
 form.d8.value=d8;
 form.d9.value=0;



 ab2=eval(form.ab2.value);
 ab5=300;
 ab4=c8*3;
 form.ab4.value=ab4;
 ab6=ab2+ab5;
 form.ab6.value=ab6;
 ab7=ab2-ab4+ab5;

 form.ab7.value=ab7;

 ab6=eval(form.ab6.value);
 ab8=ab7-ab6;
 form.ab8.value=ab8;
 ac4=d8*3;
 form.ac4.value=ac4;
 ac5=400;
 ac2=eval(form.ac2.value);
 if(ab7-ac4+ac5<ac2)
   {
    ac7=ac2;
   }
  else
   {
   ac7=ab7-ac4+ac5;
   }
 form.ac7.value=ac7;
 ac6=ab7-ac4+ac5;
 form.ac6.value=ac6;
 ac8=ac7-ac6;

 form.ac8.value=ac8;
 ad4=0;
```

```
form.ad4.value=ad4;
ad5=200;
ad6=ac7-ad4+ad5

form.ad6.value=ad6;

if(ac7-ad4+ad5<ac2)
  {
  ad7=ac2;
  }
   else
   {
   ad7=ac7-ad4+ad5;
   }
form.ad7.value=ad7;

form.ab9.value=ac8;
ad8=ad7-ad6;
form.ac9.value=ad8;
form.ad8.value=ad8;
form.ad9.value=0;



aab2=eval(form.aab2.value);
aab5=400;
aab4=c8*2;
form.aab4.value=aab4;
aab6=aab2+10;
form.aab6.value=aab6;
aab7=aab2-aab4+aab5;

form.aab7.value=aab7;

aab6=eval(form.aab6.value);
aab8=aab7-aab6;
form.aab8.value=aab8;
aac4=d8*2;
form.aac4.value=aac4;
aac5=200;
aac2=eval(form.aac2.value);
if(aab7-aac4+aac5<aac2)
  {
    aac7=aac2;
   }
 else
   {
   aac7=aab7-aac4+aac5;
   }
form.aac7.value=aac7;
aac6=aab7-aac4+aac5;
form.aac6.value=aac6;
aac8=aac7-aac6;

form.aac8.value=aac8;
aad4=0;
form.aad4.value=aad4;
aad5=200;
aad6=aac7-aad4+aad5

form.aad6.value=aad6;

if(aac7-aad4+aad5<aac2)
  {
  aad7=aac2;
  }
   else
   {
   aad7=aac7-aad4+aad5;
   }
form.aad7.value=aad7;

form.aab9.value=aac8;
aad8=aad7-aad6;
form.aac9.value=aad8;
form.aad8.value=aad8;
form.aad9.value=0;
}

</script>
</head>

<BODY>
<style> body{background: black
url(file:///C:/WINDOWS/Resources/Them
es/Panther/Wallpaper/Aqua_Blue.jpg)
fixed}--> </style>
<p align="center"><b><font size="6"
face="Bookman Old
Style">M.R.P.</font></b></p>
<p> </p>
<CENTER>
<img src="nova árvore.bmp">
<FORM>    &n
bsp;     &
nbsp;     
      
;     &nbs
p;     &nb
sp;     &n
bsp;     &
nbsp;     
      
;     &nbs
```

```
p;     &nb
sp;     &n
bsp;     &
nbsp;     
      
;     &nbs
p;     &nb
sp;     &n
bsp;     &
nbsp;    
<INPUT onclick=calculo(this.form)
type=button value= Calcular 
name=calcular>   &nb
sp;     &n
bsp;
<INPUT type=reset
value= Apagar >
<p> </p>
<TABLE>
  <TR>
    <TD width="190"><b>Produto
</b></TD>
    <TD width="135"><b>Stock Inicial
</b></TD>
    <TD width="135"><b>Stock Segurança
</b></TD>
    <TD width="135"><b>Lead-Time
</b></TD></TR>
  <TR>
    <TD><b>T-shirt </b></TD>
    <TD><INPUT size=12 name=b2>
</TD>
    <TD><INPUT size=12 name=c2>
</TD>
    <TD><p align="center"><b>1
</b></p></TD></TR>
  <TR>
    <TD><b>Período </b></TD>
    <TD><p align="center"><b>1
</b></p></TD>
    <TD><p align="center"><b>2
</b></p></TD>
    <TD><p align="center"><b>3
</b></p></TD></TR>
  <TR>
    <TD><b>Procura </b></TD>
    <TD><INPUT size=12 name=b4>
</TD>
    <TD><INPUT size=12 name=c4>
</TD>
    <TD><INPUT size=12 name=d4>
</TD></TR>
  <TR>
    <TD><b>Recepções Programadas
</b></TD>
    <TD><p align="center"><b>100
</b></p></TD>
    <TD><p align="center"><b>150
</b></p></TD>
    <TD><p align="center"><b>210
</b></p></TD></TR>
  <TR>
    <TD><b>Stock Previsto (Inicial)
</b></TD>
    <TD><INPUT disabled readOnly
size=12 name=b6> </TD>
    <TD><INPUT disabled readOnly
size=12 name=c6> </TD>
    <TD><INPUT disabled readOnly
size=12 name=d6> </TD></TR>
  <TR>
    <TD><b>Stock Previsto (Final)
</b></TD>
    <TD><INPUT disabled readOnly
size=12 name=b7> </TD>
    <TD><INPUT disabled readOnly
size=12 name=c7> </TD>
    <TD><INPUT disabled readOnly
size=12 name=d7> </TD></TR>
  <TR>
    <TD><b>Necessidades Líquidas
</b></TD>
    <TD><INPUT disabled readOnly
size=12 name=b8> </TD>
    <TD><INPUT disabled readOnly
size=12 name=c8> </TD>
    <TD><INPUT disabled readOnly
size=12 name=d8> </TD></TR>
  <TR>
    <TD><b>Lançamentos Previstos
</b></TD>
    <TD><INPUT disabled readOnly
size=12 name=b9> </TD>
    <TD><INPUT disabled readOnly
size=12 name=c9> </TD>
    <TD><INPUT disabled readOnly
size=12 name=d9>
</TD></TR></TABLE>
<p> </p>


<TABLE>
```

```
 <TR>
   <TD width="190"><b>Produto
</b></TD>
   <TD width="135"><b>Stock Inicial
</b></TD>
   <TD width="135"><b>Stock Segurança
</b></TD>
   <TD width="135"><b>Lead-Time
</b></TD></TR>
 <TR>
   <TD><b>Linha </b></TD>
   <TD><INPUT size=12 name=ab2>
</TD>
   <TD><INPUT size=12 name=ac2>
</TD>
   <TD><p align="center"><b>1
</b></p></TD></TR>
 <TR>
   <TD><b>Período </b></TD>
   <TD><p align="center"><b>1
</b></p></TD>
   <TD><p align="center"><b>2
</b></p></TD>
   <TD><p align="center"><b>3
</b></p></TD></TR>
 <TR>
   <TD><b>Procura </b></TD>
   <TD><INPUT disabled readOnly
size=12 name=ab4> </TD>
   <TD><INPUT disabled readOnly
size=12 name=ac4> </TD>
   <TD><INPUT disabled readOnly
size=12 name=ad4> </TD></TR>
 <TR>
   <TD><b>Recepções Programadas
</b></TD>
   <TD><p align="center"><b>300
</b></p></TD>
   <TD><p align="center"><b>400
</b></p></TD>
   <TD><p align="center"><b>200
</b></p></TD></TR>
 <TR>
   <TD><b>Stock Previsto (Inicial)
</b></TD>
   <TD><INPUT disabled readOnly
size=12 name=ab6> </TD>
   <TD><INPUT disabled readOnly
size=12 name=ac6> </TD>
   <TD><INPUT disabled readOnly
size=12 name=ad6> </TD></TR>
 <TR>
   <TD><b>Stock Previsto (Final)
</b></TD>
   <TD><INPUT disabled readOnly
size=12 name=ab7> </TD>
   <TD><INPUT disabled readOnly
size=12 name=ac7> </TD>
   <TD><INPUT disabled readOnly
size=12 name=ad7> </TD></TR>
 <TR>
   <TD><b>Necessidades Líquidas
</b></TD>
   <TD><INPUT disabled readOnly
size=12 name=ab8> </TD>
   <TD><INPUT disabled readOnly
size=12 name=ac8> </TD>
   <TD><INPUT disabled readOnly
size=12 name=ad8> </TD></TR>
 <TR>
   <TD><b>Lançamentos Previstos
</b></TD>
   <TD><INPUT disabled readOnly
size=12 name=ab9> </TD>
   <TD><INPUT disabled readOnly
size=12 name=ac9> </TD>
   <TD><INPUT disabled readOnly
size=12 name=ad9> </TD>
</TR></TABLE>


<p> </p>


<TABLE>
 <TR>
   <TD width="190"><b>Produto
</b></TD>
   <TD width="135"><b>Stock Inicial
</b></TD>
   <TD width="135"><b>Stock Segurança
</b></TD>
   <TD width="135"><b>Lead-Time
</b></TD></TR>
 <TR>
   <TD><b>Tecido </b></TD>
   <TD><INPUT size=12 name=aab2>
</TD>
   <TD><INPUT size=12 name=aac2>
</TD>
   <TD><p align="center"><b>1
</b></p></TD></TR>
 <TR>
   <TD><b>Período </b></TD>
```

```
   <TD><p align="center"><b>1
</b></p></TD>
   <TD><p align="center"><b>2
</b></p></TD>
   <TD><p align="center"><b>3
</b></p></TD></TR>
  <TR>
   <TD><b>Procura </b></TD>
   <TD><INPUT disabled readOnly
size=12 name=aab4> </TD>
   <TD><INPUT disabled readOnly
size=12 name=aac4> </TD>
   <TD><INPUT disabled readOnly
size=12 name=aad4> </TD></TR>
  <TR>
   <TD><b>Recepções Programadas
</b></TD>
   <TD><p align="center"><b>400
</b></p></TD>
   <TD><p align="center"><b>200
</b></p></TD>
   <TD><p align="center"><b>200
</b></p></TD></TR>
  <TR>
   <TD><b>Stock Previsto (Inicial)
</b></TD>
   <TD><INPUT disabled readOnly
size=12 name=aab6> </TD>
   <TD><INPUT disabled readOnly
size=12 name=aac6> </TD>
   <TD><INPUT disabled readOnly
size=12 name=aad6> </TD></TR>
  <TR>
   <TD><b>Stock Previsto (Final)
</b></TD>
   <TD><INPUT disabled readOnly
size=12 name=aab7> </TD>
   <TD><INPUT disabled readOnly
size=12 name=aac7> </TD>
   <TD><INPUT disabled readOnly
size=12 name=aad7> </TD></TR>
  <TR>
   <TD><b>Necessidades Líquidas
</b></TD>
   <TD><INPUT disabled readOnly
size=12 name=aab8> </TD>
   <TD><INPUT disabled readOnly
size=12 name=aac8> </TD>
   <TD><INPUT disabled readOnly
size=12 name=aad8> </TD></TR>
  <TR>
   <TD><b>Lançamentos Previstos
</b></TD>
   <TD><INPUT disabled readOnly
size=12 name=aab9> </TD>
   <TD><INPUT disabled readOnly
size=12 name=aac9> </TD>
   <TD><INPUT disabled readOnly
size=12 name=aad9>
</TD></TR></CENTER></TABLE></F
ORM>

<p><b><font face="Comic Sans
MS">M.R.P. elaborado
por:</font></b></p>
<p><font face="Comic Sans
MS">     
      
;     &nbs
p;     &nb
sp;     &n
bsp;   
<img src="bolinha.gif" width="12"
height="12">
Carolina
Fernandes    &n
bsp;     &
nbsp;     
      
;     &nbs
p;     &nb
sp;   
<img src="bolinha.gif" width="12"
height="12">Diana
Rocha     
      
;     &nbs
p;     &nb
sp;     &n
bsp;     &
nbsp;     
      
;     
<img src="bolinha.gif" width="12"
height="12">Guilherme Bacinello</p>
</font></p>
</BODY></HTML>
```

Conclusão

No decorrer do trabalho, concluímos que Javascript é uma linguagem simples, ideal para se criar, de forma rápida, páginas web dinâmicas e atraentes.

Sendo uma linguagem simples, é de fácil percepção e compreensão. Porém, é necessário ter um bom conhecimento da linguagem Html, caso contrario o aprendizado torna-se complicado.

Em suma, construir um M.R.P. (*Material Requirement Planning*) em Javascript tornou-se uma tarefa mais simples que a esperada pois trabalhamos com uma linguagem simples e clara, onde foi bastante útil os conhecimentos sobre a linguagem Html anteriormente adquiridos.

Referências Bibliográficas

Webliografia

- www.dsi.uminho.pt/disciplinas/PLP
- www.javascript.com
- www.javascriptcity.com
- www.linhadecodigo.com.br/desenvolvimento/ javascript.asp
- http://www.scriptbrasil.com.br/?class=2.1&menu=javascript

Anexos

Estruturas de Javascript

Anexo 1

Variáveis

Em JavaScript, variáveis dinâmicas podem ser criadas e inicializadas sem declarações formais. Existem dois tipos de abrangência para as variáveis:

Global - Declaradas/criadas fora de uma função. As variáveis globais podem ser acessadas em qualquer parte do programa.

Local - Declaradas/criadas dentro de uma função. Só podem ser utilizadas dentro da função onde foram criadas e precisam ser definidas com a instrução Var.

Com relação a nomenclatura, as variáveis devem começar por uma letra ou pelo caracter sublinhado '_', o restante da definição do nome pode conter qualquer letra ou número.

É importante ressaltar que a variável *Codigo* é diferente da variável *codigo*, que por sua vez é diferente de *CODIGO*, sendo assim, é preciso cuidado ao definir o nome das variáveis. Podem existir variáveis globais com o mesmo nome de variáveis locais, porém, esta prática não é aconselhável.

Existem três tipos de variáveis: Numericas, Booleanas e Strings.

Como já era de se esperar, as variáveis numéricas são assim chamadas pois armazenam números, as Booleanas valores lógicos (True/False) e as Strings, sequência de caracteres.

As strings podem ser delimitadas por aspas simples ou duplas, a única restrição é que se a delimitação começar com as aspas simples, deve terminar com aspas simples, da mesma forma para as aspas duplas. Podem ser incluídos dentro de uma string alguns caracteres especiais:

\t - posiciona o texto a seguir, na próxima tabulação;

\n - passa para outra linha;

\f - form feed;

\b - back space;

\r - carrige return.

O JavaScript reconhece ainda um outro tipo de contúdo em variáveis, que é o **NULL**. Ná prática isso é utilizado para a manipulação de variáveis não inicializadas sem que ocorra um erro no seu programa.

Quando uma variável possui o valor NULL, significa dizer que ela possui um valor desconhecido ou nulo, o null não é igual a nada, nem mesmo ao próprio null.

A representação literal para NULL é a string 'null' sem os delimitadores. quando referenciado por uma função ou comando de tela, será assim que NULL será representado. Observe que NULL é uma palavra reservada.

Trabalha-se ainda com Vectores, mas para isso será necessário alguns conhecimentos sobre Programação Orientada a Objetos.

Anexo 2

Operadores

Juntamente com funções e variáveis, operadores são blocos de construção de expressões. Um operador é semelhante a uma função no sentido de que executa uma operação específica e retorna uma valor.

Operadores Unários e binários

Todos os operadores em JavaScript requerem um ou dois argumentos, chamados operandos. Aqueles que requerem um operando apenas, são denominados *operadores unários*, e os que requerem dois operandos são chamados de *operadores binários*.

Operador de String

+	**Operador de concatenação** Exemplo: Nome1="José" Nome2="Silva" Nome = Nome1+" da "+Nome2 // O resultado é: "José da Silva"

Operadores Matemáticos

+	**Adição** Exemplo: V01=5 V02=2 V=V01+V02 // resulta em: 7
-	**Subtração** Exemplo: V01=5

<table>
<tr><td></td><td>V02=2

V=V01-V02 // resulta em: 3</td></tr>
<tr><td>*</td><td>Multiplicação

Exemplo:

V01=5

V02=2

V=V01*V02 // resulta em: 10</td></tr>
<tr><td>/</td><td>Divisão

Exemplo:

V01=5

V02=2

V=V01/V02 // resulta em: 2.5</td></tr>
<tr><td>%</td><td>Módulo da divisão (resto)

Exemplo:

V01=5

V02=2

V=V01%V02 // resulta em: 1</td></tr>
<tr><td>++</td><td>Incremento. Pode acontecer de duas formas: ++Variável ou Variável++

<table><tr><td>Exemplo:

V01 = 5

V02 = ++V01 // Resulta em 6

V03 = V01 // Resulta em 6</td><td>Exemplo:

V01 = 5

V02 = V01++ // Resulta em 5

V03 = V01 // Resulta em 6</td></tr></table></td></tr>
<tr><td>--</td><td>Decremento. Pode acontecer de duas formas: --Variável ou Variável--

<table><tr><td>Exemplo:

V01 = 5

V02 = --V01 // Resulta em 4</td><td>Exemplo:

V01 = 5

V02 = V01-- // Resulta em 5</td></tr></table></td></tr>
</table>

	V03 = V01 // Resulta em 4	V03 = V01 // Resulta em 4

Operadores relacionais

<	Menor que
>	Maior que
==	Igual
!=	Diferente
>=	Maior ou igual
<=	Menor ou igual

Operadores lógicos

&&	E lógico
||	Ou lógico

Operadores de atribuição

=	Atribuir
+=	Soma ou concatenação e atribuição. x+=5 // é o mesmo que: x=x+5
-=	Subtração e atribuição. x-=5 // é o mesmo que: x=x-5
*=	Multiplicação e atribuição. x*=5 // é o mesmo que: x=x*5
/=	Divisão e atribuição. x/=5 // é o mesmo que: x=x/5
%=	Módulo e atribuição. x%=5 // é o mesmo que: x=x%5

Anexo 3

Objectos

Um objecto pode ser qualquer coisa e está sujeito a determinados métodos. Um método geralmente é uma função que gera alguma informação referente ao objecto.

Trabalhar com objectos é a única forma de manipular com **vectores**.

Anexo 4

Estruturas de Controle

Existem algumas estruturas de controle que permitem modificar o fluxo de execução de um programa. Estas estruturas permitem executar o código baseado em condições lógicas ou um número determinado de vezes.

- For...
- For...In
- If...Else...
- While...

For...

Repete uma seção do código um determinado número de vezes. Consiste de uma declaração que define as condições da estrutura e marca seu início. esta declaração é seguida por uma ou mais declarações executáveis, que representam o corpo da estrutura.

Estabelece um contador inicializando uma variável com um valor numérico. O contador é manipulado através da <acção> especificada no comando toda a vez que o loop alcança seu fim, permanecendo nesse loop até que a <condição> seja satisfeita ou a instrução Break seja executada.

Forma geral:

For (<inicialização> ; <condição> ; <acção>)

{ Corpo da Estrutura }

No exemplo abaixo, o bloco de instruções será executado 10 vezes, pois a variável Contador é inicializada com o valor 1 e o bloco de instruções será executado enquanto Contador for menor que 11. A cada execução do bloco de instruções Contador é incrementado.

For (var Contador = 1; Contador < 11; Contador++)

{ document.write(Contador); }

For...In

Este comando tem por objetivo, procurar a ocorrência de uma variável, dentro das propriedades de um objecto, ao encontrar a referida variável, um bloco de comandos pode ser executado.

Forma geral:

For (variavel In objecto)

{

bloco de comandos

}

Exemplo:

Esta função procura por uma propriedade do **Objecto**, cujo o nome esteja especificado pela variável **Procura**, onde **Nome** é uma string correspondendo ao nome do objecto.

Function SearchIn(Procura,Objecto,Nome)

{

Var ResultadoDaBusca = ""

For (Procura In Objecto)

{

document.write(Nome+"."+Procura+"="+Objecto[Procura]+"
");

}

}

If...Else...

A estrutura If... executa uma porção de código se a condição especificada for verdadeira. A estrutura pode também ser especificada com código alternativo para ser executado se a condição for falsa.

Function VerificaIdade(anos**)**

{

If anos >= 16

```
{

Return ('Já pode votar!')

}

else

{

Return (' Ainda é muito cedo para votar...')

}

}
```

Uma alternativa para economizar If's seria a utilização de uma expressão condicional, que funciona para situações mais simples, o exemplo acima ficaria da seguinte forma:

```
VariavelDeRetorno= (anos>=16) ? 'Já pode votar!' : 'Ainda é muito cedo para votar...'
```

While

Outro tipo de loop é aquele baseado numa condição ao invés de no número de repetições. Por exemplo, se necessitar que um determinado processo seja repetido até que um determinado teste dê um resultado verdadeiro ou seja executada a instrução Break.

Forma geral:

```
while (<condição>)

{ Corpo da Estrutura }
```

No exemplo abaixo, o bloco de instruções será executado 10 vezes, pois a variável Contador é inicializada com o valor 1 e o bloco de instruções será executado enquanto Contador for menor que 11. A cada execução do bloco de instruções Contador é incrementado.

```
Var Contador=1;

While ( Contador < 11 )

{ document.write(Contador++) ;}
```

Anexo 5

Comandos

Além das estruturas de controle, o JavaScript oferece alguns comandos:

- Break
- Continue
- Var
- With
- Function
- Return
- Comment

VAR

Em JavaScript, nem sempre é necessário definir uma variável antes de utilizá-la, é o que ocorre com variáveis globais, porém, é importante ressaltar que a utilização da instrução **var**, a nível de documentação é muito bem-vinda. Já nas definições de variáveis locais, é obrigatório a utilização da instrução var.

Você pode armazenar um valor na própria linha de definição da variável, se não o fizer, para o JavaScript, esta variável possui um valor desconhecido ou nulo.

Não é obrigatória a utilização de uma instrução var para cada variável declarada, na medida do possível, você pode declarar várias variáveis em uma só instrução var.

Forma geral:

Var NomeDaVariável [= <valor>];

Exemplo:
Var Contador = 0;
Var Inic="",Tolls=0,Name="TWR";
Var Teste; // Neste caso, Teste possui valor **null**

with

Quando é preciso manipular várias propriedades de um mesmo objeto, provavelmente

não se deseja repetir todas as vezes a digitação do nome do objeto. A instrução With, permite fazer isso eliminando a necessidade de digitar o nome do objeto todas as vezes.

Forma geral:

with (<objecto>)

{

... Instruções

}

Por exemplo vamos supor que será necessário executar uma série de operações matemáticas:

```
with (Math)
{

a=PI;
b=Abs(x);
c=E;

}
```

Break

Pode ser executado somente dentro de loops For... ou While... e tem por objectivo o cancelamento da execução do loop sem que haja verificação na condição de saída do loop, passando a execução a linha imediatamente posterior ao término do loop.

Forma geral:

Break

Exemplo:

Neste exemplo, quando a variável x atinge o valor 5 o loop é cancelado, e é impresso o número 5 na tela.

```
For (var x=1 ; x < 10 ; x++)

{

If (x = = 5)

{

Break
```

```
}

}

document.write(x) // resulta: 5
```

Continue

Pode ser executado somente dentro de loops For... ou While... e tem por objetivo o cancelamento da execução do bloco de comandos passando para o início do loop.

Forma geral:

```
Continue
```

Exemplo:

Neste exemplo, serão impressos os números de 1 a 10, com excepção do número 5, ou seja, quando a variável x atinge o valor 5 a execução do bloco de comandos é interrompida e o controle retorna ao início do loop, onde a variável x será incrementada.

```
For (var x=1 ; x < 10 ; x++)

{

If (x = = 5)

{

continue

}

document.write(x)

}
```

Funções

As funções podem ser definidas como um conjunto de instruções, agrupadas para executar uma determinada tarefa. Dentro de uma função pode existir uma chamada a outra função. Para as funções podem ser passadas informações, as quais são chamadas de parâmetros.

As funções podem ou não retornar alguma informação, o que é feito com o comando **Return**.

A definição de uma função é feita da seguinte forma:

Function NomeDaFunção([parametro1, parametro2, ..., parametroN]**)**

{

...

[**Return(**ValorDeRetorno**)**]

}

A chamada de funções é feita da seguite forma:

NomeDaFunção([parâmetros]**)**

Funções são melhor declaradas entre as tags <head> de sua página HTML. Funções são frequentemente chamadas por eventos accionados pelo usuário. Assim parece razoável colocar as funções entre as tags <head>. Elas são carregadas antes que o usuário faça alguma coisa que possa chamar uma função.

```
<html>
<head>
<script language="LiveScript">
Function hello(){

alert("Alô mundo!!!");

}
</script>
</head>

<body>
...

<script>hello();</script>

...
</body>
</html>
```

Nota: Em JavaScript, não é possível utilizar-se da recursividade, ou seja, uma função não pode chamar ela mesma

Comentários

Comentários podem ser formulados de varias maneiras, dependendo do efeito que se quer. Para comentários longos de várias linhas, ou blocos de comentários, use:

/* O barra-asterisco inicia um bloco de comentário que pode conter quantas linhas você precisar todo o texto após o barra asterisco é ignorado, até que asterisco-barra seja encontrado, terminando assim o bloco de comentário */

Para comentários de uma linha, usa-se barra dupla (//) para introduzir o comentário. Todo o texto seguindo este simbolo até o próximo carrige-return será considerado um comentário e ignorado para fins de processamento. Exemplo:

// este texto será tratado como comentário

Os códigos JavaScript podem ser colocados em campos de comentário de modo que browsers antigos não mostrem o próprio código JavaScript, como vemos a seguir:

```
<html>
<head>
<script language="LiveScript">
<!-- hide script from old browsers
Function hello(){

alert("Olá  mundo!!!");

}
// end hiding contents -->
</script>
</head>

<body>
... <script>hello();</script>

...
</body>
</html>
```