Apresentação JavaScript

Marcelo Fey

Introdução

O objetivo deste material é apresentar a linguagem Javascript, muito utilizada atualmente, em conjunto com HTML, para construir paginas dinâmicas e interfaces de aplicações no ambiente Web.

A partir deste estudo, espera-se posicionar a tecnologia Javascript no contexto do desenvolvimento Web, a fim de que desenvolvedores conheçam o potencial das linguagens interpretadas pelos navegadores.

O que é Javascript

Javascript é uma linguagem de script (*scripts são "miniprogramas" interpretados e voltados para execução de tarefas especificas*) com uma sintaxe bastante similar a C, C++, Pascal e Delphi. e é uma linguagem de script orientada a objetos.

Os comandos e funções de Javascript são inseridos dentro de um documento da Web, junto com tags HTML e texto.

Quando o navegador de um usuário acessa este documento, ele formata a página, executando o programa nela inserido.

O que é Javascript

É uma linguagem orientada a objetos com um conjunto de objetos já embutidos. Sempre que algo acontece em uma página Web, ocorre um evento. Eventos podem ser qualquer coisa: um botão recebe um clique, o mouse é arrastado, uma página é carregada, um formulário é enviado, e assim por diante.

Javascript é uma linguagem dirigida por eventos, no sentido de que é projetada para reagir quando um evento ocorre.

O que é Javascript

A linguagem Javascript foi projetada para manipular e apresentar informação através de um navegador.

Ela não é capaz de recuperar informações de outro arquivo ou salvar dados em um servidor da Web, ou no computador do usuário. Isto significa que não é possível escrever um programa Javascript que, por exemplo, ler os diretórios de um computador, ou apagando arquivos do usuário.

O que é Javascript

Javascript é uma linguagem independente de plataforma, ou seja, o código escrito nesta linguagem não depende de uma plataforma específica (Windows, Macintosh, UNIX, etc.), depende apenas do navegador que a interpreta.

Dessa forma, quer o usuário tenha um navegador para Windows, Macintosh ou UNIX, o código Javascript será executado sem que nenhuma adaptação seja necessária.

No contexto de **navegador**, podemos sim encontrar diferenças em seus comportamentos.

Elemento SCRIPT

Container tag: <SCRIPT>...</SCRIPT>

Dentro de um documento HTML, a linguagem Javascript é delimitada pelo par de tags <SCRIPT> e </SCRIPT>.

Podemos posicionar a tag <SCRIPT> dentro dos elementos <HEAD> e </HEAD>, <BODY> e </BODY>, ou ambos – é possível incorporar múltiplos elementos <SCRIPT> dentro de um documento.

Scripts dentro da tag <HEAD> são carregados antes que o resto da página seja carregado, tornando-se um excelente lugar para colocar suas funções Javascript, assim garantir que elas estejam disponíveis para outras partes da página!

Elemento SCRIPT

Antes de começar

Para conseguir usar as funcionalidades no navegador do usuário, precisamos verificar se a tecnologia está ativa no mesmo.

```
<script>
        document.write(”JavaScript Ativado!");
</script>
<noscript>
        JavaScript desativado!
</noscript>
```

Elemento SCRIPT

Usabilidade

Similar ao uso do CSS, podemos imcorporar seu uso de diferentes maneiras:

Inline

```
<TAG_HTML  EVENTO="Comando JavaScript"  >
```

OutLine

```
<SCRIPT> ...Comandos ou Funções JavaScript... </SCRIPT>
```

File

```
<HEAD>
   <SCRIPT SRC="funcoes.js"></SCRIPT>
</HEAD>
```

Elemento SCRIPT

Criando o primeiro script

O exemplo abaixo imprime a frase “Bom dia!” na página. Observe que, apesar de não existirem elementos entre as marcações <BODY> e </BODY>, esta frase é exibida. Isto ocorre porque a função **document.write()** instrui o navegador a colocar o que estiver entre apóstrofos na página.

```
<html>
   <head>
      <script>  document.write(”Texto inserido pelo JavaSacript! "); </script>
  </head>
  <body>
  </body>
</html>
```

Elemento SCRIPT

Criando o segundo script

Uma das razões de utilizar Javascript é a possibilidade de montar um texto para ser exibido, incluindo valores de variáveis que podem ser inseridos pelo usuário.

```
<script language="javascript”>
    var nome;
    nome = window.prompt("Digite o seu nome:");
    document.write("Bom dia, " + nome + "!<BR> * * * ");
</script>
```

Eventos

Javascript é uma linguagem dirigida por eventos.

Eventos (tais como, clicar no mouse, ou pressionar um botão, etc.) são utilizados para controlar a interação do usuário com o aplicativo.

```
function exibe() {
      if ( !(confirm(“Quer encerrar?”) ) {
        documento.write (“Oi, pessoal da Internet!”);
      }
}
...
<a OnClick=” exibe()”> Clica aqui! </a>
```

Manipuladores de eventos

Manipuladores de eventos Javascript servem para interfacear um script com atividades do sistema ou ações do usuário.

Eles são divididos em 2 categorias:

eventos de sistema

eventos de mouse.

Manipuladores de eventos

Eventos de sistema

Os eventos de sistema disponíveis na tag BODY:

OnLoad *(quando carregar)*

OnUnload *(quando descarregar)*

Eles não exigem a interação do usuário para serem ativados.

Manipuladores de eventos

ONLOAD

Este evento é ativado após a página HTML ser completamente carregada. Ele pode ser associado as tags <BODY> ou <FRAMESET>.

Manipuladores de eventos

ONLOAD

```
<html>
   <head>
      <script>
            function chegada(){
               window.alert(“Seja bem-vindo ao nosso site!”);
            }
      </script>
   </head>
   <body OnLoad=“chegada()”>
         Veja que interessante utilização do evento <I>OnLoad</I>.
   </body>
</html>
```

Manipuladores de eventos

ONUNLOAD

Este evento é ativado após a página HTML ser abandonada *(seja através do clique sobre algum link, ou sobre os botões de avanço/retrocesso do browser).*

Ele pode ser associado as tags <BODY> ou <FRAMESET>

Manipuladores de eventos

Eventos mouse.

OnClick	*(quando Clicado)*
OnFocus	*(quando entra em Focu)*
OnBlur	*(quando retira o Focu)*
OnChange	*(quando for alterado)*
OnSelect	*(quando for selecionado)*
OnSubmit	*(quando for submetido)*
OnMouseOver	*(quando o mouse passa em cima)*

Eles exigem a interação do usuário para serem ativados. *(através do mouse ou não)*

Manipuladores de eventos

ONCLICK

O evento mais básico de mouse é tratado pelo manipulador OnClick. Este evento é ativado sempre que se dá um clique sobre um objeto que aceita evento de clique de mouse. Objetos que aceitam um evento OnClick são links, caixas de verificação e botões.

Manipuladores de eventos

ONCLICK

```
<html>
  <head>
    <script>
            function mensagem() {
              window.alert("Você clicou neste campo");
          }
    </script>
  </head>
  <body>
    <a href="exemplo3.html" OnClick="mensagem()"> <i>Link</i> comum</a><br>
    <form>
            <input type="radio" OnClick="mensagem()"><i>Radio</i> <br>
            <input type="checkbox" OnClick="mensagem()"><i>Checkbox</i> <br>
            <input type="reset" OnClick="mensagem()"> <br>
            <input type="submit" OnClick="mensagem()"> <br>
    </form>
  </body>
</html>
```

Manipuladores de eventos

ONFOCUS

O foco ocorre quando um objeto torna-se o item em foco. Isto acontece quando o usuário clicar ou alternar para um objeto específico na pagina. Este evento pode ser associado aos objetos text, password, textarea e select *(definidos pelas tags <INPUT>, <TEXTAREA> e <SELECT>)*.

Manipuladores de eventos

ONFOCUS

```
<html>
  <head>
    <script>
      function foco() {
        window.alert("O campo EMAIL está em foco");
      }
  </script>
  </head>
  <body>
    <form>
      Nome: <input name="nome" type="text"><br>
      Email: <input name="email" type="text" OnFocus="foco()"><br>
      Telefone: <input name="telefone" type="text">
    </form>
  </body>
</html>
```

Manipuladores de eventos

ONBLUR

Este evento é ativado quando um objeto torna-se fora de foco - quando se muda para outra janela, ou aplicativo, ou quando se passa para outro objeto utilizando cliques do mouse, ou a tecla TAB. Ele é associado aos objetos text, password, textarea e select *(definidos pelas tags <INPUT>, <TEXTAREA> e <SELECT>).*

Manipuladores de eventos

ONBLUR

```
<html>
  <head>
    <script>
      function semfoco() {
        window.alert("O campo EMAIL perdeu o foco");
      }
    </script>
  </head>
  <body>
    <form>
      Nome: <input name="nome" type="text"> <br>
      Email:<input name="email" type="text" OnBlur="semfoco()"> <br>
      Telefone: <input name="telefone" type="text">
    </form>
  </body>
</html>
```

Manipuladores de eventos

ONCHANGE

Este evento é ativado sempre que um objeto perde o foco e o seu valor é alterado. Ele é associado aos objetos text, password, textarea e select *(definidos pelas tags <INPUT>, <TEXTAREA> e <SELECT>).*

Manipuladores de eventos

ONCHANGE

```
<html>
  <head>
    <script>
      function mudou1() {
          document.form1.completo.value=document.form1.nome.value;
      }
    </script>
  </head>
  <body>
      <form name=form1>
        Nome: <input name="nome" type="text" OnChange="mudou1()”/> <br>
        Sobrenome: <input name="sobrenome" type="text”/>
      </form>
  </body>
</html>
```

Manipuladores de eventos

ONSELECT

Este evento é ativado quando o usuário seleciona *(deixa em destaque)* parte do texto em um dos objetos aos quais está associado.

São eles: text, password e textarea *(definidos pelas tags <INPUT> e <TEXTAREA>).*

Manipuladores de eventos

ONSELECT

```
<html>
  <head>
    <script>
      function selecao() {
          window.alert("Campo selecionado");
      }
    </script>
  </head>
  <body>
    <form>
      Campo input texto: <input type="text" OnSelect="selecao()">
      Campo input senha: <input type="password" OnSelect="selecao()">
      Campo textarea: <textarea OnSelect="selecao()”></textarea>
  </form>
 </body>
</html>
```

Manipuladores de eventos

ONSUBMIT

•Este evento é ativado no momento de enviar os dados do formulário. Ele é associado ao objeto form *(definido pela tag <FORM>).*

```
function submete() {
   window.alert("Evento OnSubmit ativado!");
}
...
<form name=form1 OnSubmit="submete()">
        <input type=submit>
</form>
```

Manipuladores de eventos

ONSUBMIT

Muitas vezes, os dados que são inseridos em um formulário precisam ser separados, analisados, manipulados ou verificados quanto a erros antes de serem transmitidos para o servidor.

O evento OnSubmit permite a captura e, se necessário, a **interrupção** do envio dos dados de um formulário.

Isto é realizado chamando-se a função a partir do manipulador OnSubmit, fazendo com que ela retorne verdadeiro ou falso.

Se a função associada ao manipulador retornar falso, os dados do formulário não serão enviados. Esta funcionalidade pode ser verificada a partir do código a seguir.

Manipuladores de eventos

ONSUBMIT

```
<html>
  <head>
    <script>
      function submete() {
        if (document.form1.campo1.value == "" || document.form1.campo2.value == "")
          return false;
        else
          return true;
      }
    </script>
  </head>
  <body>
    <form name=form1 action="exemplo8b.html" OnSubmit="return submete()">
      Campo 1: <input type="text" size=10 name=campo1> <br>
      Campo 2: <input type="text" size=10 name=campo2> <p>
      <input type=submit>
    </form>
  </body>
</html>
```

Manipuladores de eventos

ONMOUSEOVER

Este evento é ativado quando o ponteiro do mouse passa sobre um objeto do tipo links ou botões.

```
function ativa() {
    window.alert("Evento OnMouseOver ativado!");
}

...

<input type="submit" value="Botão Submit" OnMouseOver="ativa()">
```

Construções de Javascript

Javascript apresenta algumas restrições quanto ao nome de variáveis/funções:

Não é permitido colocar espaço em branco em um nome;

Não é permitido incluir um hífen (“-”) em um nome;

Não é permitido colocar os seguintes caracteres em um nome: . , ; “ ‘ ?

Embora seja possível usar dígitos em um nome, ele precisa começar com uma letra;

Não é permitido utilizar, como nome de uma nova variável/função, alguma das palavras reservadas de Javascript, segue a relação:

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

Declaração de Variáveis

Cada variável tem que ser declarada como global ou local. A única diferença entre estes dois tipos em Javascript é onde elas estão localizadas dentro do código. É possível definir variáveis antes de atribuir um valor a elas, ou no momento em que fizer a atribuição.

Variáveis Locais:

São definidas dentro do corpo de uma função. Elas são validas apenas dentro do corpo da função onde foram definidas *(escopo limitado).*

Variáveis Globais:

São definidas fora de todos os corpos de funções de um arquivo Javascript. Elas são validas dentro de qualquer função do arquivo.

Declaração de Variáveis

Variáveis Locais e Variáveis Globais

```
var total = 0;  // Global

function adiciona(valor){
   var a = valor + 10; // Local
   total = total + a;
}
function subtrai(valor){
   var b = valor - 10; // Local
   total = total - b;
}
```

Tipos de Valores

Existem 4 (quatro) tipos de variáveis reconhecidos por Javascript:

Número: qualquer número positivo ou negativo. Este número pode ser inteiro no formato decimal, hexadecimal ou octal. Pode também ser número de ponto flutuante com/sem exponencial;

Booleano: true ou false (sem aspas);

String: conjunto de caracteres limitados por aspas/apóstrofos.

Nulo: null (palavra chave que denota o valor nulo).

Expressões

Expressão é um conjunto de literais *(constantes)*, variáveis e operadores que, avaliados, resultam em um único valor *(número, String ou booleano).*

Existem 3 (três) tipos de expressões em Javascript:

expressões aritméticas: resultam em um número;

expressões de String: resultam em uma sequência de caracteres;

expressões lógicas: resultam em verdadeiro ou falso;

Operadores

Operadores são símbolos especiais que controlam como uma expressão deve ser avaliada.

Os operadores podem, ainda, ser classificados de acordo com o tipo dos operandos que manipulam:

- operadores **aritméticos**,
- operadores de **comparação**,
- operadores de **String**,
- operadores **lógicos**,
- operadores **bit** a **bit**
- operadores de **atribuição**.

Operadores

OPERADORES ARITMÉTICOS

Operadores aritméticos constroem expressões aritméticas. Eles recebem e retornam números.

Operador	Função
+	Soma
-	Subtração
*	Multiplicação
/	Divisão

Operadores

OPERADORES DE COMPARAÇÃO

Um operador de comparação compara seus operandos e retorna um valor booleano. Estes operandos podem ser números ou String.

Operador	Função
==	Igual a
!=	Diferente de
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a

Operadores

OPERADORES LÓGICOS

Os operadores lógicos retornam valores booleanos.

Operador	Função
&&	E, AND
||	OU, OR
!	NÃO, NOT
&	E, AND
|	OU, OR
^	OU Exclusivo, XOR
~	NÃO, NOT
<<	Deslocamento à esquerda
>>	Deslocamento à direita
>>>	Deslocamento à direita com preenchimento de zeros

Operadores

OPERADORES DE ATRIBUIÇÃO

Javascript dá suporte a um método abreviado de escrever operações.

Operador	Função
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
x ^= y	x = x ^ y
x |= y	x = x | y

Objetos

HIERARQUIA

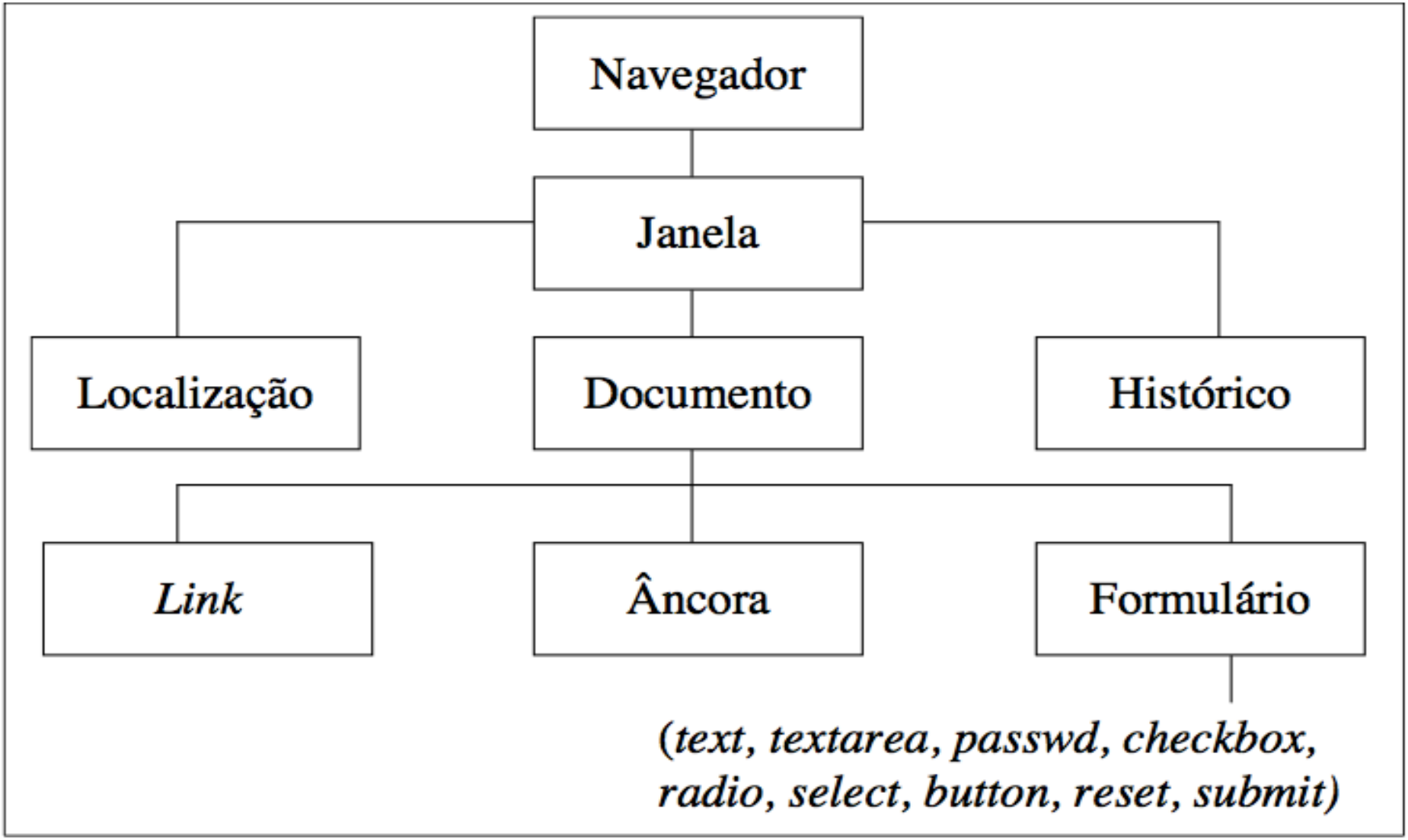
O nível mais alto de objetos em Javascript consiste naqueles objetos que pertencem a *navigator* (navegador). Diretamente abaixo deste nível, estão os objetos *window* (janela).

Cada janela tem uma árvore de níveis que se ramifica a partir dela. Estas árvores consistem em *location* (localização), *history* (histórico) e document (documento).

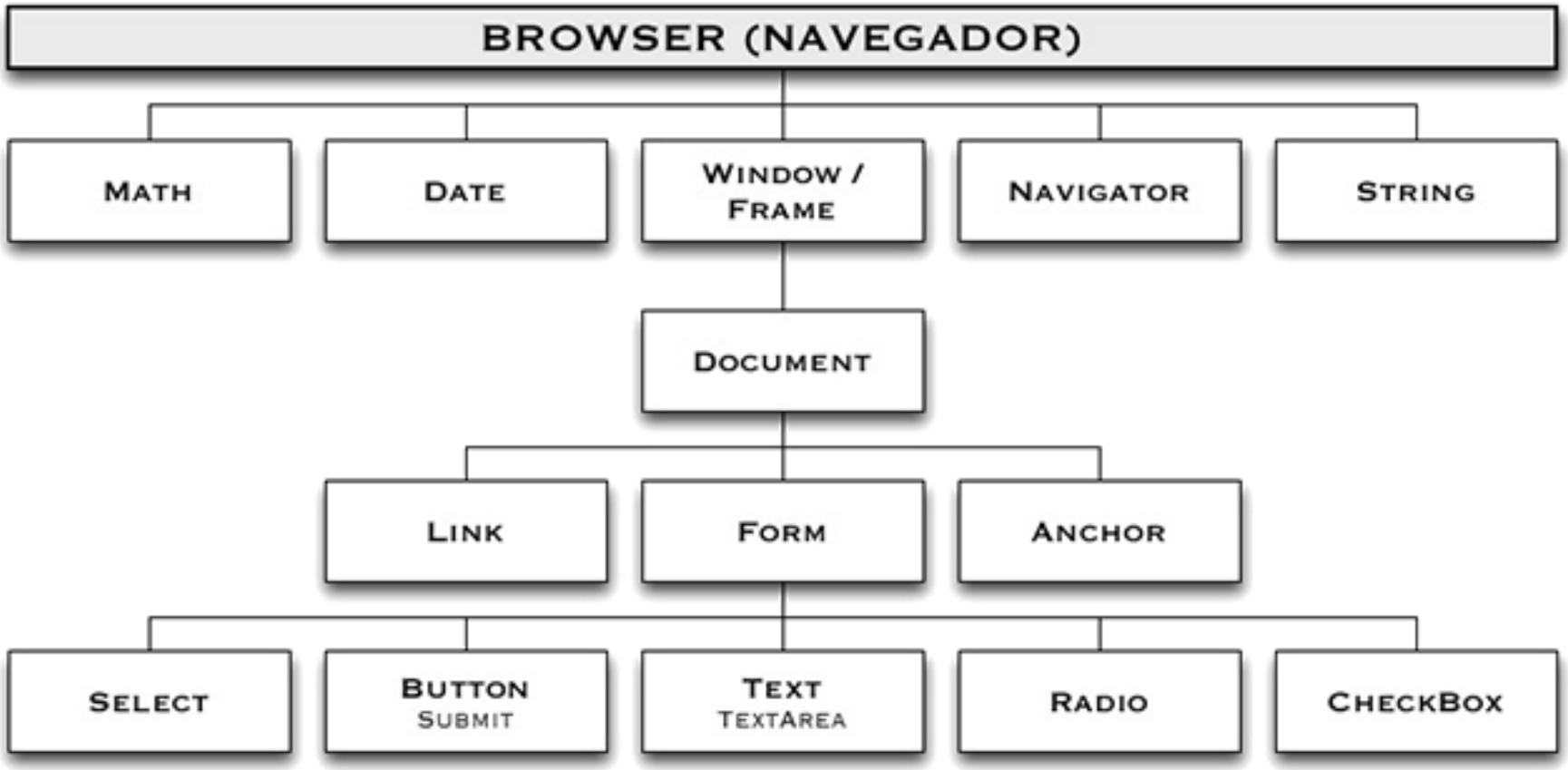
A cada nível há outros objetos e abaixo da árvore de documentos há outra ramificação.

No browser, os objetos seguem a mesma estrutura hierárquica da página HTML: de acordo com essa hierarquia, os descendentes dos objetos são propriedades dos mesmos.

Objetos


Navegador
Janela
Localização
Documento
Histórico
Link
Âncora
Formulário
(text, textarea, passwd, checkbox,
radio, select, button, reset, submit)

Objetos


BROWSER (NAVEGADOR)
MATH
DATE
WINDOW / FRAME
NAVIGATOR
STRING
DOCUMENT
LINK
FORM
ANCHOR
SELECT
BUTTON
SUBMIT
TEXT
TEXTAREA
RADIO
CHECKBOX

Objetos

Declaração *for... in*

Repete uma variável sobre todas as propriedades de um objeto. Este é um tipo especial de declaração de laço que varre todas as propriedades de um objeto.

Sintaxe:

```
for (propriedade in objeto) {
    declarações
}
```

Onde, objeto é o objeto em que se está trabalhando, e propriedade é o nome da propriedade do objeto. A cada iteração do laço, propriedade recebe uma propriedade diferente do objeto. O laço se encerra quando todas as propriedades de um objeto já foram visitadas.

Objetos

Declaração *for... in*

```
<html>
   <head>
      <title>Javascript</title></head>
   <body>
      <h3>Propriedades do Documento</h3>
      <hr>
      <script>
            for (val   in document)
               document.write( val + "<br>");
      </script>
    </body>
</html>
```

Objetos

Declaração *for... in*

```
<html>
   <head>
      <title>Javascript</title></head>
   <body>
      <h3>Propriedades do Documento</h3>
      <hr>
      <script>
            for (val   in document)
                document.write( val + "<br>");
      </script>
    </body>
</html>
```

A Natureza Orientada a Objetos de Html

Javascript considera HTML uma linguagem orientada a objetos, na qual os diversos tags HTML correspondem a diferentes tipos de objetos Javascript.

```
<html>
    <head>
       <title>Minha pagina</title>
    </head>
    <body>
        <form name="formulario1">
              <input type="button" name="botao1">
        </form>
   </body>
</html>
```

A partir do código acima, obtemos os seguintes objetos Javascript:

- document.title : título da página
- document.formulario1 : formulário da página
- document.formulario1.botao1 : botão do formulário

A Natureza Orientada a Objetos de Html

OBJETO NAVIGATOR

Este objeto dá informações sobre o navegador.

appName: retorna o nome do browser do usuário.

appVersion: retorna a versão do browser e a versão do sistema operacional

appCodeName: retorna o nome do código de desenvolvimento interno do desenvolvedor de um browser específico.

userAgent: usada em cabeçalhos HTTP para fins de identificação, é a combinação das propriedades appCodeName e appVersion. Servidores Web usam esta informação para identificar os recursos que o navegador dispõe.

A Natureza Orientada a Objetos de Html

OBJETO LOCATION

Este objeto é utilizado para identificar o documento corrente.

***protocol*:** retorna o protocolo de transporte do documento.

***hostname*:** identifica o nome do computador hospedeiro.

***port*:** especifica a porta para o endereço. Esta informação é utilizada apenas se uma porta não padrão estiver sendo usada.

***pathname*:** define o caminho e o nome do arquivo.

***search*:** retorna quaisquer comandos de consulta que possam estar embutidos na URL corrente. Valores de *search* são separados do resto da URL por um sinal de interrogação ("?").

***hash*:** retorna quaisquer âncoras que possam ter sido passadas na URL. Valores de hash são separados do resto da URL por um sinal de cerquilha ("#").

A Natureza Orientada a Objetos de Html

OBJETO CHECKBOX

Utilizado na construção de caixas de verificação. Suas propriedades são:

name: especifica o nome da caixa.

value: especifica o valor da caixa.

```
nomeForm.nomeCheckbox.value = “1”
```

checked: valor booleano que especifica o estado de seleção da caixa *(selecionada ou não-selecionada).*

```
if ( nomeForm.nomeCheckbox.checked == true ) {
    comandos…
}
```

defaultChecked: valor booleano que especifica o estado default de seleção da caixa.

A Natureza Orientada a Objetos de Html

OBJETO RADIO

Corresponde a um array de botões, onde todos os botões compartilham a mesma propriedade *name*. Suas propriedades são:

name: especifica o nome do objeto.

checked e **defaultChecked**: funcionamento idêntico ao definido em ***checkbox***.

length: especifica o comprimento do array.

A Natureza Orientada a Objetos de Html

OBJETOS INPUTs

Utilizado para entrada/saída de dados.

name: especifica o nome do objeto.

value: especifica o valor do objeto.

defaultValue: especifica o valor default do objeto.

A Natureza Orientada a Objetos de Html

OBJETO SELECT

Utilizado para construir caixas de seleção.

name: especifica o nome do objeto.

options: array que contém uma entrada para cada opção de uma caixa de seleção.

length: especifica o comprimento do array de opções.

O exemplo a seguir identifica que opções foram selecionadas na caixa de seleção.

A Natureza Orientada a Objetos de Html

OBJETO SELECT

```
function listSelected(obj) {
    for (i=0; i < obj.length; i++) {
        document.write(“ “ + obj.options[i].name + “ “);
        if (!obj.options[i].selected) {
            document.write(“não está selecionada<BR>”);
        }
        else {
            document.write(“está selecionada<BR>”);
        }
}
```

A Natureza Orientada a Objetos de Html

OBJETO ARRAY

É possível criar um vetor através do objeto *Array*, pré-definido no Javascript.

```
<script>
    var nome_do_array = new Array() ;
    nome_do_array[3] = 'valor';
    alert(nome_do_array[3]);
</script>
```

O primeiro elemento é o de número 0.

A Natureza Orientada a Objetos de Html

OBJETO STRING

Em Javascript, toda ***string*** é um ***objeto***, e, portanto, tem métodos e propriedades associadas.

length: retorna o comprimento de um string.

indexOf( ): retorna a posição da primeira ocorrência do caractere procurado.

Sintaxe: string.indexOf(caracter_procurado, posição_inicial_de_busca)

charAt( ) : retorna o caracter encontrado na posição indicada.

Sintaxe: string.charAt(posição)

toUpperCase( ) : traduz todo caracter dentro de uma string para letra **maiúscula**.

toLowerCase( ) : traduz todo caracter dentro de uma string para letra **minúscula**.

substring( ) : retorna uma sequência de caracteres de uma string maior.

Sintaxe: string.substring(ini, fim) ambos valores numéricos.

A Natureza Orientada a Objetos de Html

OBJETO DATE

O objeto ***Date*** lhe ajuda a manipular datas.

Para criar um objeto do tipo ***Date***, deve-se utilizar a seguinte sintaxe:

var **minhaData** = new Date();

Se não for indicado nenhum parâmetro, será criado um objeto com a data e a hora atual do sistema.

A Natureza Orientada a Objetos de Html

OBJETO DATE

```
var d = new Date();
var dia  = d.getDate();
var mes= d.getMonth() + 1; //Months are zero based
var ano = d.getFullYear();
document.write(dia+ "-" + mes+ "-" + ano);
```

Ou

```
var d1=new Date();
d1.toString('yyyy-MM-dd');
d1.toString('dddd, MMMM ,yyyy');
```

A Natureza Orientada a Objetos de Html

OBJETO WINDOW

O objeto ***window*** representa a janela do navegador ou um frame.

É criado um objeto window sempre que o navegador encontra uma tag <BODY> ou <FRAMESET>.

Também são criados objetos para cada frame definido.

Propriedades mais utilizadas:

defaultStatus: a mensagem que será exibida quando não tiver nenhuma outra na status bar do navegador.

Height: esta propriedade contém a altura, em pixels, da janela do navegador;

Width: semelhante à propriedadeanterior, porém trabalha com alargura;

A Natureza Orientada a Objetos de Html

OBJETO WINDOW

name: representa o nome da janela;

status: especifica a mensagem a ser exibida na status bar do navegador. É muito útil para comunicar ao usuário pequenas mensagens.

alert( ): exibe uma mensagem para o usuário;

back( ): é equivalente a apertar o botão back do navegador.

forward():temo mesmo efeito do botão forward do navegador.

A Natureza Orientada a Objetos de Html

OBJETO WINDOW

open( ): abre uma nova janela. O método recebe como parâmetros uma URL (o endereço da página que vai ficar na nova janela), o nome da janela e uma string com suas carcterísticas;

close( ): fecha a janela especificada. O Javascript somente pode fechar automaticamente janelas abertas por ele. Caso contrário, aparece uma caixa de confirmação para o usuário;

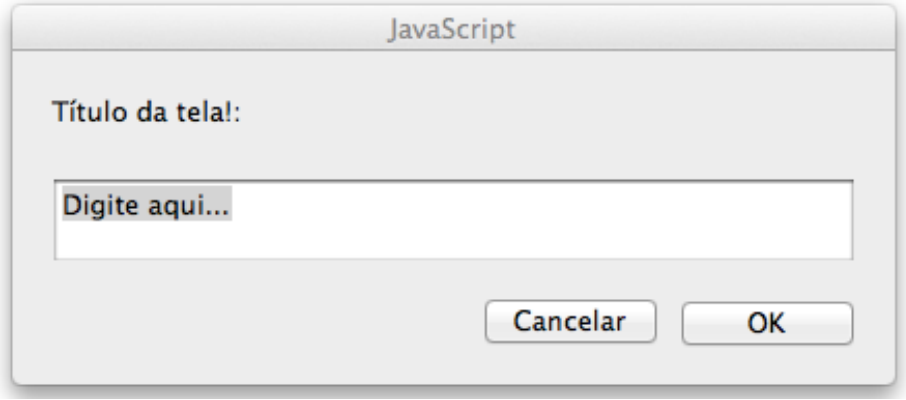
confirm( ): exibe uma caixa de mensagem para o usuário com duas opções: OK e Cancel. Caso o usuário pressione OK, o método retorna true. Caso contrário, false. Ele recebe como parâmetro uma string com a mensagem a ser exibida para o usuário;

A Natureza Orientada a Objetos de Html

OBJETO WINDOW

prompt( ): exibe uma caixa de mensagem e campo para o usuário entrar com uma string. O método retorna a string digitada pelo usuário. São aceitos dois parâmetros. O primeiro é uma string com a mensagem a ser exibida e o segundo é o valor padrão da string a ser digitada pelo usuário.

```
<script>
    var varM;
    varM = window.prompt('Título da tela!: ','Digite aqui...');
    alert('Foi digitado = '+varM);
</script>
```



Orientação a Objetos em JavaScript

```
<script>
  function Pessoa(){
    var nome;
    this.getNome  = getNome;
    this.setNome  = setNome;
    this.mostraValores = mostraValores;

    function getNome() {
      return nome;
    }

    function setNome(_nome) {
      nome = _nome;

    }
    function mostraValores() {
      return 'Apropriedade "Nome" desse instancia é = ( ' + this.getNome() +' ).' ;
    }
  }

  var objPes = new Pessoa();
  objPes.setNome('Marcelo Fey');
  document.write(objPes.mostraValores());
</script>
```