Introdução ao JavaScript

Índice

1 - Introdução

Wikipédia - http://pt.wikipedia.org/wiki/Javascript

JavaScript é uma linguagem de programação criada pela Netscape em 1995, que a princípio se chamava LiveScript, a Netscape após o sucesso inicial desta linguagem, recebe uma colaboração considerável da Sun, após esta colaboração, podemos dizer que o JavaScript é uma linguagem compatível com a linguagem Java, por esta razão, a semelhança dos nomes "JavaScript".

A linguagem foi criada para atender, principalmente, as seguintes necessidades:

- Validação de formulários no lado cliente (programa navegador);
- Interação com a página.

Assim, foi feita como uma linguagem de script. Javascript tem sintaxe semelhante à do Java, mas é totalmente diferente no conceito e no uso.

De acordo com seu sistema de tipos JavaScript é:

- fraca - sua tipagem é mutável;
- dinâmica - uma variável pode assumir vários tipos diferentes durante a execução;
- implícita - as variáveis são declaradas sem tipo.

1. É interpretada, ao invés de compilada;
2. Possui ótimas ferramentas padrão para listagens (como as linguagens de script, de modo geral);
3. Oferece bom suporte a expressões regulares (característica também comum a linguagens de script).

Sua união com o CSS é conhecida como DHTML. Usando o Javascript, é possível modificar dinamicamente os estilos dos elementos da página em HTML.

Dada sua enorme versatilidade e utilidade ao lidar com ambientes em árvore (como um documento HTML), foi criado a partir desta linguagem um padrão ECMA, o ECMA-262, também conhecido como ECMAScript. Este padrão é seguido, por exemplo, pela linguagem ActionScript da Adobe(Antigamente Macromedia, porém a empresa foi vendida à Adobe).

Além de uso em navegadores processando páginas HTML dinâmicas, o JavaScript é hoje usado também na construção do navegador Mozilla, o qual oferece para a criação de sistemas GUI todo um conjunto de ferramentas (em sua versão normal como navegador, sem a necessidade de nenhum software adicional), que incluem (e não apenas) um interpretador de Javascript, um comunicador Javascript <-> C++ e um interpretador de XUL, linguagem criada para definir a interface gráfica de aplicações.

O uso de JavaScript em páginas XHTML, pelo padrão W3C, deve ser informado ao navegador da seguinte forma:

```
<script type="text/javascript">

/* aqui fica o script */

</script>
```

Caso contrário, o navegador irá interpretar o script como sendo código HTML, escrevendo-o na página.

2 - Instruções, delimitador de instrução e comentários

http://www.w3schools.com/js/default.asp
http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro

Instrução – é um conunto de comandos suficientes para que o interpretador realize uma função, para que possa executar a instrução e acontecer algo significativo.

Delimitador de instruções – no JavaScript podemos usar o ponto e vírgula ou a quebra de linha.

Comentários – Em JavaScript existem dois tipos de comentários:

// - comentáro oriundo do C++ e para uma única linha
/* ... */ - comentário oriundo do C e para múltiplas linhas.

```
// Aqui comenta apenas esta linha

/*
Aqui pode comentar
várias
linhas
*/
```

3 - Script interno

Podemos criar scripts em JavaScript dentro de um arquivo em HTML, como a seguir:

```
<html>
    <head>
        <title>Título da Página</title>

        <script>
        function alerta(){
            alert('Olá Mundo!');
        }
        </script>
    </head>
    <body>
    <a href="#" onClick="alerta()">Clique a aqui</a>
    </body>
</html>
```

Este link nos serve para executar a função em JavaScript.

4 - Uso nas tags do HTML

Também podemos executar o código JavaScript diretamente dentro das tags do HTML. Vejamos alguns exemplos usando o form em HTML anteriormente mostrado.

```
<body onLoad="document.frmInserir .login.focus()">
<h1>Formulário de Envio da Dados usando JavaScript nas Tags</h1>

<a href="#" onClick="alerta()">Clique a aqui</a>
<form name="frmInserir" action="inserir.php" method="POST" onSubmit="if(login.value==')
{alert('Favor preencher o login');login.focus();return false;}">
Nome<input type="text" name="nome" size="45" maxlength="45" TABINDEX="1" value="João
de Brito" READONLY><br>
Login<input type="text" name="login" size="12" maxlength="12" TABINDEX="2"><br>
Senha<input type="password" name="senha" size="32" maxlength="32" TABINDEX="3"><br>
<input type="hidden" name="controle" value="35">
<input type="submit" name="enviar" value="Enviar" TABINDEX="11">
<input type="reset" name="limpar" value="Limpar" TABINDEX="12">
</form>

</body>
```

Inicialmente jogamos o foco no campo login ao carregar o documento com:
onLoad="document.frmInserir .login.focus()"

Depois ao submeter o form (ao clicar no botão Enviar) nós verificamos se o campo login está vazio, caso esteja emitimos um alert, jogamos o foco novamente no campo login e cancelamos a ação com:
onSubmit="if(login.value==') {alert('Favor preencher o login');login.focus();return false;}"

Mas a forma ideal de trabalhar com JavaScript é em um arquivo externo e realizando um include no HTML ou PHP, como a seguir.

5 - Script externo

Criamos o arquivo, sem adicionar a tar incial <script> nem a final </script>, apenas com as funções e instruções e procedimentos, como no exemplo:

Vamos criar um arquivo chamado **funcoes.inc.js** com o conteúdo a seguir:

```
function login_foco(){
    document.frmInserir.login.focus();
}
function login_preencher(){
    if(document.frmInsert.login.value==') {
        alert('Favor preencher o login');
        return false;
    }
}
```

Agora como fazer com que a página HTML saiba da existência das funções em JS e as use? Devemos fazer o include do arquivo JS com as funções com a linha abaixo escrita entre as tags <head> e </head>:

```
<script type="text/javascript"  src="funcoes.inc.js"></script>
```

Veja o exemplo anterior mas agora usando um arquivo externo:

```
<head>
<script type="text/javascript"  src="funcoes.inc.js"></script>
<head>
<body onLoad="login_foco()">
<h1>Formulário de Envio da Dados usando JavaScript nas Tags</h1>

<a href="#" onClick="alerta()">Clique a aqui</a>
<form name="frmInserir" action="" method="POST" onSubmit="login_preencher()">
Nome<input type="text" name="nome" size="45" maxlength="45" TABINDEX="1" value="João de Brito" READONLY><br>
Login<input type="text" name="login" size="12" maxlength="12" TABINDEX="2"><br>
Senha<input type="password" name="senha" size="32" maxlength="32" TABINDEX="3"><br>
<input type="hidden" name="controle" value="35">
<input type="submit" name="enviar" value="Enviar" TABINDEX="11">
<input type="reset" name="limpar" value="Limpar" TABINDEX="12">
</form>
</body>
```

Obs.: Aqui tive problema quando usei o arquivo externo e criei a função login_preencher(), pois o return false não funcionou com meus testes no Firefox, já o trecho de código dentro da tag <form funcionou sem problemas.

6 - Variáveis

Nomes de variáveis em JavaScript podem ser curtos, com apenas uma letra como também podem ser descritivos, com várias letras e algarismos, sendo que não podem iniciar com um algarismo.

Exemplos:

```
var x;
var y = 4;
var nome = 'Antônio';
```

Veja que podemos apenas declarar a variável, sem qualquer valor como também já podemos declarar com valor.

Também podemos declarar sem o var, assim:

```
y = 4;
nome = 'Antônio';
```

7 - Operadores

Em JavaScript:

= é utilizado para atribuir valores (operador de atribuição)
+ é usado para somar valores ou para concatenar strings (operador aritmético)

Exemplos:

```
x = 5;
z = 3;
y = x + z;
```

Primeiro o lado direito do igual é somado e depois é atribuído a y.

Operadores aritméticos:

Operadores aritméticos

Operador	Descrição	Exemplo(s)
+	Soma valores.	`a = 2 + 3;` `b = b +1;`
-	Subtrai valores (como operador binário).	`x = x - 5;` `x = a - b`
-	Muda sinal (como operador unitário).	`x = -x;` `x = -(a + b);`
*	Multiplica valores.	`a = 2 * 3;` `b = c * 5;`
/	Divide valores.	`a = 50 / 3;` `b = b * 4;`
%	Resto da divisão.	`d = 5 % 3;` d assume valor 2.
++(var)	Incremento de 1 (antes).	Se x é 2, `y = ++x` faz x igual a 3 e depois y igual a 3.
(var)++	Incremento de 1 (depois).	Se x é 2, `y = x++` faz y igual a 2 e depois x igual a 3.
--(var)	Decremento de 1 (antes).	Se x é 2, `y = --x` faz x igual a 1 e depois y igual a 1.
(var)--	Decremento de 1 (depois).	Se x é 2, `y = x--` faz y igual a 2 e depois x igual a 1.

Operadores de atribuição

Operador	Descrição	Exemplo(s)
=	Atribui o valor do operando esquerdo ao operando direito.	`x = 3;` `a = b + c;`
+=	Soma 2 valores e atribui o resultado ao primeiro valor.	`x += 3;` Se x era 1, passa para 4.
-=	Subtrai 2 valores e atribui o resultado ao primeiro.	`x -= 3;` Se x era 1, passa para -2.
*=	Multiplica 2 valores e atribui o resultado ao primeiro.	`x *= 2;` Se x era 4, passa para 8.
/=	Divide 2 valores e atribui o resultado ao primeiro.	`x /= 2;` Se x era 4, passa para 2.
%=	Calcula o resto da divisão de 2 valores e atribui o resultado ao primeiro.	`x %= 2;` Se x era 3, passa para 1.

Operadores de comparação

Operador	Descrição	Exemplo(s), supondo a = 3 e b = 5
==	Verdadeiro se os operandos são iguais. Se não são do mesmo tipo, a linguagem tenta converter para a correta comparação.	`a == 3;` // retorna verdadeiro `a == b;` // retorna falso
!=	Verdadeiro se os operandos não são iguais. Se não são do mesmo tipo, a linguagem tenta converter para a correta comparação.	`a != 3;` // retorna falso `a != b;` // retorna verdadeiro
===	Verdadeiro se os operandos são iguais e do mesmo tipo.	`a === 3;` // retorna verdadeiro `a === "3";` // retorna falso
!==	Verdadeiro se os operandos não são iguais ou não são do mesmo tipo.	`a !== b;` // retorna verdadeiro `a !== "3";` // retorna verdadeiro
>	Verdadeiro se o operando esquerdo é maior que o direito.	`a > b;` // retorna falso `b > a;` // retorna verdadeiro
>=	Verdadeiro se o operando esquerdo é maior ou igual ao direito.	`a >= 3;` // retorna verdadeiro `b >= 7;` // retorna falso
<	Verdadeiro se o operando esquerdo é menor que o direito.	`a < b;` // retorna verdadeiro `b < a;` // retorna falso
<=	Verdadeiro se o operando esquerdo é menor ou igual ao direito.	`a <= 3;` // retorna verdadeiro `a <= 0;` // retorna falso

Operadores de strings

Operador	Descrição	Exemplo(s)
+	Concatenar strings.	str_1 = "Bom"; `str_2 = str_1 + " dia";` str_2 contém "Bom dia"

+=	Concatenar e atribuir o resultado ao operando da esquerda.	`str_1 = "Bom";`
		`str_1 += " dia";`
		
		str_1 contém "Bom dia"

Operadores lógico

Em geral são usados com expressões que retornam valores booleanos, isto é, verdadeiro ou falso.

Operador	Descrição	Exemplo(s), supondo a = 3 e b = 5
&&	E lógico: retorna verdadeiro se ambas as expressões são verdadeiras e falso nos demais casos	`a==3 && b<10` // retorna verdadeiro `a!=3 && b==5` // retorna falso
||	OU lógico: retorna verdadeiro se pelo menos uma das expressões é verdadeira e falso se todas são falsas	`a==3 || b<10` // retorna verdadeiro `a!=3 || b==5` // retorna verdadeiro `a==1 || b==3` // retorna falso
!	NÃO lógico: retorna verdadeiro se o operando é falso e vice-versa	

O operador + é usado em Strings

O operador + usado com strings concatena as strings.
str1 = "Cursos ";
str2 = "Gratuítos Online";
str = str1 + str2;

Dica: Ao adicionar string com número o resultado será um número.

str = 5 + "5"; // O resultado deverá ser 55

Exemplo:
<script>
var x=5;
y="5";
alert(x+y);
</script>

8 - Estruturas de controle

8.1 - if, else e else if

Sintaxe:

```
if (condição) {
    instruções a serem executadas se a condição for verdadeira;
}

if (condição) {
    instruções a serem executadas se a condição for verdadeira;
}else{
    instruções a serem executadas se a condição for false;
}

if (condição) {
    instruções a serem executadas se a condição for verdadeira;
}else if(condição2){
    instruções a serem executadas se a condição2 for verdadeira;
}else{
    instruções a serem executadas se a condição2 for falsa;
}
```

Exemplos:

```
var x = 2;
var y = 5;

if(x > y){
    alert('x é maior que y');
}

if(x > y){
    alert('x é maior que y');
}else{
    alert('y é maior que x');
}

if(x > y){
    alert('x é maior que y');
}else if(y>x){
    alert('y é maior que x');
}else{
    alert('y é menor que x');
}
```

Lembrando que o fluxo somente entrará em }else if(y>x){ se (x > y) for falso.

8.2 - for

Sintaxe:
```
for (var=valorinicial;var<=valorfinal;var=var+incremento){
    instruções a serem executadas tantas vezes quanto seja o laço;
}
```

Exemplo:

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
    document.write("O número é " + i);
    document.write("<br />");
}
</script>
</body>
</html>
```

document.write() é uma função que escreve na tela.

8.3 - while

Sintaxe:
```
while (var<=valorfinal){
    instruções a serem executadas enquanto var for menor que valor final;
}
```

Exemplo:
```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5){
    document.write("O número é " + i);
    document.write("<br />");
    i++;
}
</script>
</body>
</html>
```

Observe que a variável precisou ser incrementada manualmente, caso contrário fica em loop infinito.

8.4 - do .. while

Sintaxe:
```
do{
    instruções a serem executadas;
}
while (var<=endvalue);
```

Observe a diferença marcante entre os laços while e do ... while. No primeiro não será executada nenhuma instrução caso a expressão inicial não seja verdadeira, já no laço do ... while as instruções são executadas pelo menos a primeira vez independente de a expressão ser ou não verdadeira. Devemos sempre ter essa diferença em mente ao usar os laços.

Exemplo:
```
<html>
<body>
<script type="text/javascript">
var i=0;
do{
    document.write("O número é " + i);
    document.write("<br />");
    i++;
}
while (i<=5);
</script>
</body>
</html>
```

8.5 - switch

Sintaxe:
```
switch (n){
    case 1:
        bloco1 a ser executado;
        break;
    case 2:
        bloco2 a ser executado;
        break;
    default:
        bloco a ser executado, caso n seja diferente de 1 e de 2;
}
```

Exemplo:

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.

var d=new Date();
oDia=d.getDay();
switch (oDia){
    case 5:
        document.write("Finalmente Sexta");
        break;
    case 6:
        document.write("Super Sábado");
        break;
    case 0:
        document.write("Domingo de alegria");
        break;
    default:
        document.write("Estou aguardando o final de semana!");
}
</script>
```

8.6 - break e continue

Os laços while, do...while, for e switch podem ser interrompidos usando-se os comandos brek e continue.

break – interrompe o laço e continua o código após o laço.

continue – este interrompe apenas a iteração atual, voltando para o início do laço para continuar a próxima iteração.

Observe que não é interação e sim iteração, ou seja, cada volta que dá o laço.

Exemplo de break:

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++){
    if (i==3){
        break;
    }
    document.write("The number is " + i);
    document.write("<br />");
}
```

```
</script>
</body>
</html>
```

Exemplo de continue:

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++){
    if (i==3){
        continue;
    }
    document.write("The number is " + i);
    document.write("<br />");
}
</script>
</body>
</html>
```

8.7 - for .. in

Este laço varre os elementos de um array ou as propriedades de um objeto.

Sintaxe:

```
for (variavel  in object){
    instruções a serem executadas;
}
```

Exemplo:

```
<html>
<body>
<script type="text/javascript">
var x;
var meuscarros = new Array();
meuscarros[0] = "Ford";
meuscarros[1] = "Volvo";
meuscarros[2] = "BMW";

for (x in meuscarros){
    document.write(meuscarros[x] + "<br />");
}
</script>
</body>
</html>
```

9 - Janelas de Diálogo

O JavaScript conta com 3 janelas especiais de diálogo: alert, confirm e prompt.

alert – exibe uma mensagem em uma caixa de diálogo com apenas um botão OK.

Exemplo:
alert('Algum texto!');
Não existe retorno neste diálogo.

confirm - neste diálogo é exibida uma mensagem e dois botões: OK e Cancel. Se pressionado OK retorna TRUE e se pressionado Cancel retorna FALSE.

Exemplo:
var retorno = confirm('Deseja realmente excluir?');

prompt – Este diálogo exibe uma mensagem e uma caixa de texto para receber um texto do usuário e dois botões OK e Cancel. É semelhante ao confirm, sendo que adicionado da caixa de texto.

```
var name=prompt("Favor entrar seu nome","João Brito Cunha");
if (nome!=null && nome!=""){
    document.write("Olá " + nome + "! Como você está?");
}
```

Confira também: http://www.quackit.com/javascript/tutorial/javascript_popup_boxes.cfm

Janela popup

Tipo especial de janela, onde podemos passar diversos parâmetros como largura, altura, distância da horizontal, da vertical, etc.

```
<head>
<script type="text/javascript">
<!--
function meuPopup() {
window.open( "http://www.google.com/", "minhaJanela",
"status = 1, height = 300, width = 300, resizable = 0" )
}
//-->
</script>
</head>

<body>
<form>
<input type="button" onClick="meuPopup()" value="PopUp!">
</form>
<p onClick="meuPopup()">Clique aqui também!</p>
</body>
```

Alguns parâmetros importantes:

- **dependent** – Subjanela fechará se a janela pai (a janela que abriu esta) fechar
- **fullscreen** – Exibe a janela no modo fullscreen no navegador
- **height** – A altura da nova janela em pixels
- **width** – A largura da nova janela em pixels
- **left** – Distância em pixel da margem esquerda da tela
- **top** – Distância em pixel da margem superior da tela
- **resizable** – Permite ao usuário que redimensione a jenela ou impedir de dimensionar. Atualmente não funciona no Firefox
- **status** – Exibir ou não a barra de status
- **menubar** – exibir ou não a janela de menu do navegador
- **scrollbar** – exibir ou não as barras de rolagem do navegador nas janelas
- **location** – exibir ou não a caixa de texto location do navegador

Tutorial com grande lista de parâmetros:
http://www.java2s.com/Tutorial/JavaScript/0380__Window/windowopen.htm

Exemplo:

```
window.open('window1.htm','the_first_window','location, toolbar, resizable');
```