JavaScript

Diego R. Frank, Leonardo Seibt
FIT – Faculdades de Informática de Taquara
Fundação Educacional Encosta Inferior do Nordeste
Av. Oscar Martins Rangel, 4500 – Taquara – RS – Brasil

difrank@terra.com.br, lseibt@terra.com.br

Resumo. *JavaScript é uma das linguagens mais antigas utilizadas na Web para proporcionar interatividade ou dinamismo nos sites. A finalidade deste artigo é a de apresentar um breve histórico desta linguagem, bem como um resumo das características e funcionamento da mesma.*

1. Histórico

JavaScript é uma linguagem de script criada pela Netscape em 1995, tendo sido lançada junto com o navegador Netscape Navigator 2.0. Com seu lançamento, as páginas na internet começaram a ganhar vida, implementando um mínimo de dinamicidade devido ao modo como a linguagem acessa e manipula os componentes do seu ambiente hospedeiro (nesse caso, o Netscape Navigator).

A linguagem pode ser embutida em diferentes ambientes, não se limitando aos navegadores de Internet.

Apesar da semelhança entre os nomes JavaScript e Java, a primeira linguagem não foi derivada da segunda, apesar de possuírem alguns pontos comuns. JavaScript, por ser uma linguagem interpretada, é mais flexível que o Java, dando a liberdade ao programador de não declarar uma variável antes de utilizá-la ou de adicionar novas propriedades e novos métodos a um objeto a qualquer momento. O JavaScript foi criado para que se pudesse inserir em um site certos efeitos típicos do Java, sem que fosse preciso se preocupar com programação propriamente dito e que se tornasse mais leve do que a aplicação de um arquivo Java. Existe a possibilidade de que as duas linguagens se comuniquem por meio da tecnologia **LiveConnect**.

2. Versões do JavaScript

Versão do Netscape Navigator	***Versão JavaScript***
“2.0X”	“1.0”
“3.0X”	“1.1”
“4.0 – 4.05”	“1.2”
“4.06 – 4.61”	“1.3”
“5.0”	“1.4”
“6.0”	“1.5”

3. O padrão ECMAScript

A ECMA (European Computer Manufacturers Association), em sua especificação de número 262, descreve a linguagem ECMAScript como “uma linguagem de programação multiplataforma de propósitos gerais”.

Ela foi desenvolvida a partir do JavaScript 1.1 e hoje define o núcleo das linguagens JavaScript e Java, o que, atualmente, é a única semelhança entre ambas.

4. Internet Explorer e JavaScript

O Internet Explorer não suporta diretamente JavaScript, mas tenta interpretá-la a partir da linguagem JScript, que é a implementação da Microsoft do padrão ECMAScript. A tabela abaixo mostra qual versão do Internet Explorer suporta qual versão da linguagem JavaScript (completamente ou não).

Versão do Internet Explorer	***Versão JavaScript***
"3.0"	"1.0 (parcial)"
"4.0"	"1.2"
"5.0"	"1.3 e 1.4 (parcial)"
"6.0"	"1.5"

5. Executando código JavaScript

Pode-se identificar a presença de JavaScript no código de uma página pela presença das tags <SCRIPT> e </SCRIPT>, pois é entre elas que as funções do JavaScript farão efeito dentro da página. Os scripts client-side da linguagem JavaScript são incluídos em páginas HTML de três formas:

5.1. Forma I

Colocando as instruções entre as tags <SCRIPT> e </SCRIPT>. Exemplo:

```
<HTML>
<HEAD>
<TITLE>Título da Janela</TITLE>
</HEAD>
<BODY>
Linha em HTML

<SCRIPT>
document.write("Linha em JavaScript");
</SCRIPT>

Linha em HTML
 </BODY>
 </HTML>
```

A tag <SCRIPT> pode ser colocada dentro da tag <HEAD>, da tag <BODY> ou entre as duas. Como regra de uso geral, coloca-se dentro da tag <HEAD> as partes do script que devem ser inicializadas antes de serem acessadas (funções, declaração de variáveis, criação de objetos, etc.) e dentro da tag <BODY> coloque as instruções do script que serão executadas na hora ou que farão chamadas aos elementos previamente inicializados.

5.2. Forma II

Inserindo manipuladores de eventos dentro de tags HTML específicas. Exemplo:

```
<BODY onLoad="umaFunção();">
```

A propriedade **onLoad,** inserida na tag <BODY>, é um manipulador de evento. No nosso caso, **onLoad** será ativado no momento em que o conteúdo inserido entre as tags <BODY> e </BODY> tiver sido carregado – quando isso ocorrer, as instruções JavaScript declaradas entre as aspas serão executadas (nesse exemplo, a função **umaFunção** será chamada).

5.3. Forma III

Inserindo código JavaScript dentro de um arquivo com extensão .JS e colocando o seu nome como valor do atributo **SRC** da tag <SCRIPT>. Exemplo:

```
<SCRIPT SRC=”scriptexterno.js;”></SCRIPT>
```

6. Identificadores

O nome de qualquer variável, constante, função, objeto, propriedade, método ou coleção é considerado um **identificador**. Para nomear identificadores é necessária que se sigam algumas regras para manter a validade dos mesmos:

- o primeiro caractere do nome devee ser uma letra (maiúscula ou minúscula) ou o caractere underline (“_”). Números e demais caracteres não podem ser usados como tal;
- os demais caracteres do nome podem ser letras, números e/ou o caractere underline;
- não deve haver espaços entre os caracteres do nome;
- o nome do identificador não deve ser igual a uma palavra reservada do JavaScript.

6.1. Palavras reservadas do JavaScript

As palavras listadas a seguir não podem ser usadas como nomes de identificadores. Algumas delas não estão sendo usadas atualmente pela linguagem, mas estão reservadas para uso futuro.

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

Vale lembrar que o JavaScript é case-sensitive – distingue caracteres maiúsculos de minúsculos -, ou seja, **var** não pode ser usado como nome de identificador, mas **Var** pode.

6.2. Variáveis e constantes

Variáveis são nomes que identificam endereços de memória, nos quais são armazenados valores que podem ser acessados e modifcados por um programa. Constantes são semelhantes às variáveis, no entanto, uma vez definido o seu valor, ele não pode ser modificado.

A declaração das variáveis possui a seguinte sintaxe:

var *nome_var[=expr][, ..., nome_varN[=exprN]];*

Argumento	***Descrição***
nome_var...nome_varN	Nomes das variáveis declaradas.
expr...exprN	Valores ou expressões que resultarão nos valores das variáveis declaradas (opcional).

Para declarar constantes, apenas deve-se trocar a palavra **var** por **const**.

7. Tipos de dados do JavaScript

Os dados dividem-se em três subcategorias no JavaScript: dados primários ou primitivos, dados especiais e dados compostos ou de referência.

- Dados primários ou primitivos: **string**, **booleano** e **número**.
- Dados especiais: **undefined** e **null**.
- Dados compostos ou de referência: fun**ções** e **objetos**.

8. Funções

Funções são blocos de instruções que atendem principalmente a dois propósitos:

- Diminuir o volume de digitação;
- Estruturar e organizar o código.

A declaração de funções obedece a seguinte sintaxe:

function nomeFunção([par1, ... parN]) {

instrução(ões);

[**return** expressão;]

}

Argumento	***Descrição***
nomeFunção	Nome pelo qual a função será chamada.
par1...parN	Variáveis (parâmetros) que receberão os argumentos passados à função.
instrução(ões)	Instruções executadas quando a função é chamada.
return *expressão*	Retorna o valor definido em expressão e termina a execução da função.

9. Objetos

Objetos, no JavaScript, são coleções de **propriedades** (variáveis, arrays e outros objetos) e **métodos** (funções). Os objetos podem ser divididos em quatro categorias: objetos intrínsecos (nativos da própria linguagem), objetos criados pelo usuário, objetos oferecidos pelo programa executor do código (um navegador, por exemplo) e objetos embutidos, acessados como componentes externos.

Um objeto é criado a partir de uma função especial denominada **construtora** ou a partir de um **inicializador**. As propriedades e os métodos são declarados como variáveis da função construtora. Exemplo:

```
function aluno ( nome, nota1, nota2 ) {
        this.nome = nome;
        this.nota1 = nota1;
        this.nota2 = nota2;
        this media = function media() {
                return ( this.nota1 + this.nota2 ) / 2; }
}
```

10. Referências

Damiani, Edgard B., (2001), “Guia de Consulta Rápida JavaScript”, Novatec Editora Ltda, São Paulo – SP.

Blaz Networks “JavaScript não é Java”, http://www.blaz.com.br/desdev/developer/web/artigo.asp?ID=141

Lisboa, Izaias, (2001), “Tutorial JavaScript”, http://www.codefactory.com.br, Fevereiro.