SUMÁRIO

1 LINGUAGEM DE PROGRAMAÇÃO JAVA SCRIPT – PARTE I

1.1 A LINGUAGEM JAVA SCRIPT

1.1.1 CONCEITO

O JavaScript é uma linguagem de programação implementada para ser executada dentro do navegador web, para que os programamos (scripts) escritos nesta linguagem fossem executados no próprio navegador, sem haver necessidade deste programa passar pelo servidor. Por conta disso, o JavaScript é uma linguagem conhecida como linguagem "client-side".

Esta linguagem é baseada no padrão ECMAScript padronizada pela Ecms International seguindo as especificações ECMS-262 e ISSO/IEC 16262.

Dentre as características e vantagens do JavaScript, podemos citar as seguintes:

- O JavaScript é uma *linguagem interpretada*[1] pelo navegador;
- O JavaScript não é Java;
- Apresenta similaridades com a linguagem Java e C++ na sua sintaxe;
- Apesar de ser padronizada, cada navegador pode apresentar características implementadas de forma diferente;
- O JavaScript permite aumentar a iteração entre o usuário e as páginas web;
- O script pode ser incluído na própria página html ou em arquivo próprio;
- O conteúdo do script não é exibido para o usuário, mas pode ser facilmente recuperado pelo usuário, pois o seu código não é escondido pelo navegador;
- Permite que o conteúdo das páginas seja criado conforme a iteração do usuário com a página;
- Apesenta recursos para acessar recursos dinamicamente do servidor (ajax);
- Possui recursos para trabalhar com expressões regulares, validações de campos, recursos de data e hora e manipulação de tempos e muitos outros que serão tratados neste capítulo.

1.1.2 HISTÓRICO

[1] Linguagem Interpretada: É a linguagem de programação onde o código fonte é executado por um interpretador, não sendo executado diretamente pelo sistema operacional.

A linguagem JavaScript foi inicialmente criada pela Netscape, sendo originalmente chamada de Mocha e posteriormente chamada de LiveScript, sendo lançada no ano de 1995. Posteriormente o seu nome foi rebatizado para JavaScript. Isso se deve ao fato que em 1995 a linguagem de programação Java foi lançada e para aproveitar o sucesso desta linguagem o LiveScript passou a ser chamado de JavaScript.

Rapidamente o JavaScript obteve uma grande aceitação como uma linguagem "*client-side*", por conta disso a Microsoft lançou a sua própria linguagem de scripts, sendo esta compatível com o JavaScript. O nome desta linguagem foi JScript, a qual muitas vezes é confundida com o próprio JavaScript, entranto, apesar de ser compatível com o JavaScript, esta linguagem apresenta características próprias fazendo com que determinados recursos desta não sejam compatíveis com o JavaScript.

O primeiro navegador que teve suporte ao JavaScript foi o navegador Netscape 2.0 da própria Netscape, enquanto que o JScript teve a sua primeira aparição no navegador Internet Explorer 3.0 em 1996.

Em novembro de 1996 a Netscape submeteu o JavaScript ao *Ecma Internacional*[2] para tornar a linguagem um padrão industrial, que resultou no padrão JavaScript conhecido como ECMAScript.

A linguagem JavaScript com isso se tornou um padrão no desenvolvimento para web. Entretanto, inicialmente a linguagem foi bastante denegrida pelos profissionais, pois o público alvo da linguagem era um público leigo. Foi somente a partir do uso de recursos *AJAX*[3] que o JavaScript se consolidou e recebeu mais atenção dos profissionais. A partir disso, observou-se uma grande proliferação de *Frameworks e Bibliotecas JavaScript*[4] além do surgimento de novas práticas de programação, melhorando desta forma o uso do JavaScript.

---

[2] *Ecma Internacional*: Associação fundada em 1961 dedicada a padronização de sistemas de informação. Abreviação de European Computer Manufacturees Association.

[3] Ajax: Abreviatura de Asynchronous JavaScript and XML. Este termo refere-se ao uso das tecnologias JavaScript, CSS, XML e Html para permitir a criação de solicitações assíncronas ao servidor, evitando a necessidade de submeter o conteúdo de toda uma página e consequentemente carregar toda a página html no retorno. O emprego deste recurso permite a criação de páginas mais interativas com o usuário.

[4] Frameworks e Bibliotecas JavaScript: São conjuntos de funcionalidades implementadas em JavaScript que podem ser incorporadas a uma aplicação web através do vínculo do arquivo JavaScript da biblioteca/framework em questão com a página html. Estas bibliotecas normalmente apresentam objetivos específicos, como novos objetos de interface ou facilidades para a programação. Exemplos de framework são o Jquery, ExtJs, AngularJs e vários outros.

Atualmente, qualquer website desenvolvido profissionalmente apresenta recursos que envolvem o uso de JavaScript.

1.1.3 ESTRUTURA DE UM PROGRAMA JAVA SCRIPT

Um script criado em JavaScript pode ser inserido em uma página HTML de três formas: No corpo da página HTML, no cabeçalho da página HTM ou então em um arquivo próprio, sendo apenas referenciado na página.

Para permitir uma boa separação e identificação dos códigos do sistema, manter os scripts em um arquivo separado é sempre o padrão recomendado e que deve ser seguido, mas para fins didáticos serão apresentadas as três formas de uso neste material.

Tanto na inclusão do Script no corpo da página como no cabeçalho é necessário inserir o script dentro da tag html "<*script*>". Tudo o que estiver contido dentro do bloco de código desta tag será interpretado pelo navegador como sendo um script JavaScript.

Para este tipo de inclusão dos scripts e para garantir que os mesmos não sejam exibidos na página é recomendado que o script seja inserido em um bloco de comentário do HTML. Isso é necessário, pois o JavaScript pode estar desabilitado no navegador ou então o navegador pode não possuir suporte ao JavaScript.

Abaixo seguem dois exemplos de script, um inserindo o script no corpo da página e o outro inserindo o mesmo no cabeçalho da página.

```
<HTML>
<HEAD>
      <TITLE>Meu primeiro script</TITLE>
</HEAD>
<BODY>
      <SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
            <!--

            document.write("Olá mundo!");

            -->
      </SCRIPT>
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
	<TITLE>Meu primeiro script</TITLE>
	<SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
		<!--

		document.write("Olá mundo!”);

		-->
	</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Para o caso da inclusão do script em um arquivo separado, neste arquivo basta ser inserido o script, enquanto que na página html deve ser feita a referência a este arquivo, conforme segue o exemplo de código.

```
function OlaMundo()
{
	document.write(“Olá mundo!”);
}
```

```
<HTML>
<HEAD>
	<TITLE>Meu primeiro script</TITLE>
	<SCRIPT language="javascript" scr="arquivo.js"></SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

O primeiro bloco representa o arquivo “arquivo.js”, o qual contém o script JavaScript, enquanto que o segundo bloco apresenta a forma como este arquivo deve ser relacionado na página HTML.

Na página HTML podem ser relacionados diversos arquivos JavaScript.

Caso este exemplo seja executado, ele não irá apresentar nada na página do navegador no momento, pois foi criada uma função JavaScript, assunto este que ainda não foi apresentado.

Além de ser necessário sabermos o local onde o script pode ser inserido no documento, existem alguns outros detalhes que devem ser considerados para a linguagem:

- O JavaScript é uma linguagem case sensitive, ou seja, a linguagem diferencia letras maiúsculas de letras minúsculas. Se a escrita de um comando JavaScript não seguir esta regra, o mesmo não funcionará;

- Ao final de cada linha de instrução da linguagem, deve ser informado um ponto e vírgula (“;”);
- O JavaScript é uma linguagem fracamente tipada, não sendo necessário declarar as suas variáveis. Isso será visto em detalhes no próximo tópico.

Um ponto muito importante que deve ser considerado quando se programa em JavaScript é como identificar se o script funcionou corretamente ou apresentou algum problema. Existem diversas técnicas para identificar estas situações, entretanto, todas estas técnicas estão relacionadas ao uso do Browser para identificar a execução do mesmo.

A maneira mais simples é utilizar o console do navegador para verificar as possíveis falhas do script. O código exibido abaixo terá o seguinte resultado no browser:

```
<HTML>
<HEAD>
	<TITLE>Meu primeiro script</TITLE>
</HEAD>
<BODY>
	<SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
		<!--

		document.write("Olá mundo!”);

		-->
	</SCRIPT>
</BODY>
</HTML>
```

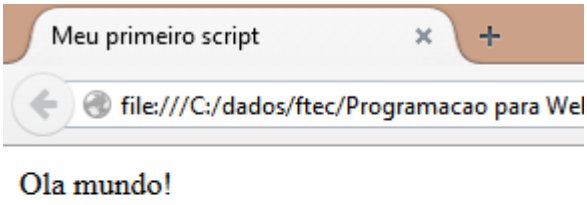

Figura 1 - Resultado do script no browser Firefox

Caso o termo “document.write...” fosse escrito como “Documento.write...” o resultado no navegador seria apenas uma página em branco. Neste caso, possivelmente teríamos dificuldade em encontrar a falha. Para auxiliar nisso, pode-se utilizar o console do Firefox, o qual é acessado através do seguinte caminho:

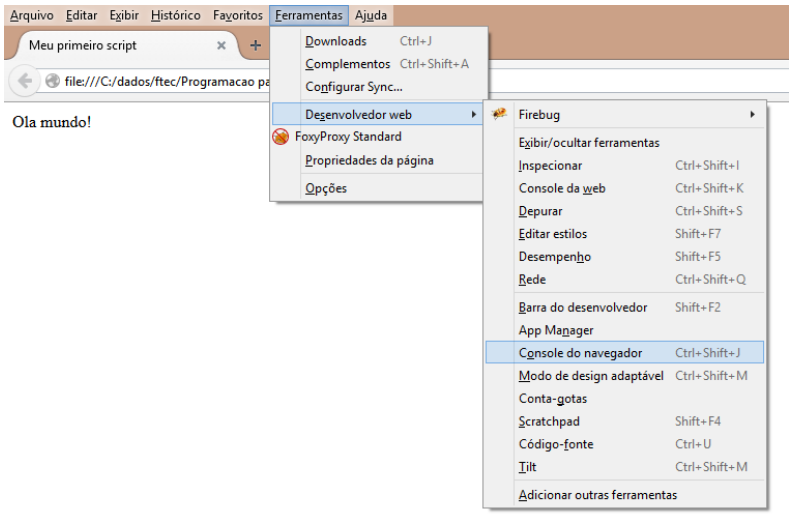

Figura 2 - Acesso ao console do browser

Quando acessado este recurso, será exibida a seguinte informação:

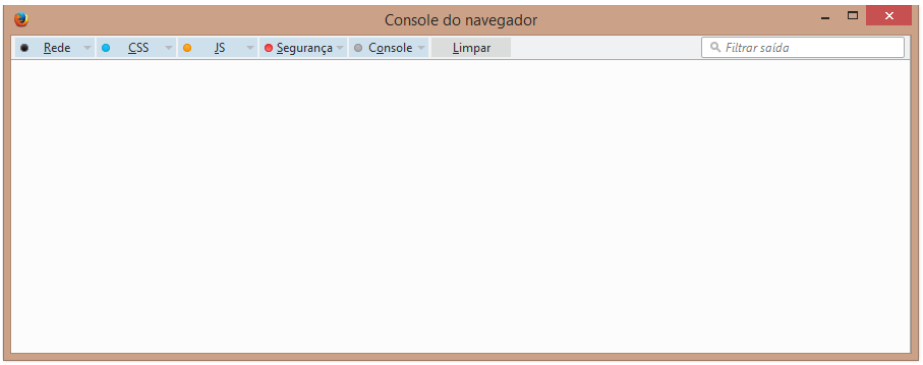

Figura 3 - Console do browser

Neste console serão exibidas mensagens conforme a execução da nossa página. Estas mensagens nos permitem identificar como está sendo realizada a execução da página pelo navegador. Caso deseje-se limpar o console, deve-se clicar no botão “Limpar” disponibilizado na tela. No caso da falha destacada acima, temos a seguinte informação no console:

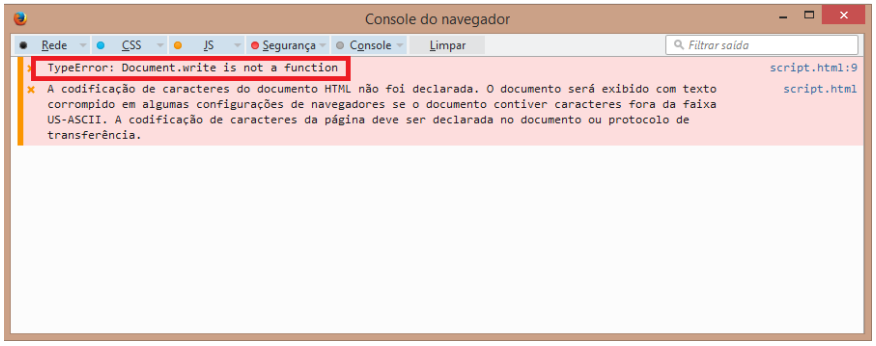

Figura 4 - Console do navegador com mensagem de erro

Observe na mensagem, que o termo “Document.write” não é uma função, logo ela não pode ser executada.

Outro fato do JavaScript é que o mesmo, por ser interpretado, sempre irá executar o script até encontrar uma falha. Será somente a partir da falha detectada que o mesmo deixará de ser executado pelo navegador.

Procure sempre utilizar o console do navegador para identificar possíveis falhas de execução dos scripts JavaScript.

1.1.4 VARIÁVEIS, ATRIBUIÇÕES E OPERAÇÕES

Um dos recursos disponíveis em todas as linguagens de programação é a possibilidade de armazenar uma informação em uma área reservada de memória. Neste conceito, a informação armazenada nesta área é identificada pelo termo “*valor*”, enquanto que a área de memória é referenciada por nós através de uma “*variável*”.

Estas variáveis podem armazenar qualquer tipo de valor e no caso do JavaScript, por ser uma linguagem fracamente tipada o tipo de informação armazenado não é explicitado como ocorre em linguagens de programação como o Java ou C++.

O tipo da variável é assumido de acordo com o valor que é atribuído a mesma. O tipo é muito importante, pois mais adiante iremos perceber que determinadas operações podem ser realizadas somente para determinados tipos de valor.

No JavaScript, para utilizar-se uma variável, o primeiro passo é declarar a mesma e após atribuir um valor a mesma de acordo com a necessidade. A declaração da variável é feita através da palavra reservada “var”, seguida do nome da variável.

Abaixo segue tabela com os tipos de dados possíveis de serem utilizado no JavaScript:

Tabela 1 - Tabela de tipos de dados

Tipo	**Descrição**	**Exemplo**
Number	Permite armazenar qualquer valor do tipo numérico, podendo ser um valor inteiro ou decimal.	3.675 1 2
String	Representa uma cadeia de	“esta é uma string”

	caracteres, sempre entre aspas duplas (“).	
Boolean	Verdadeiro ou falso	True false
Null	Valor vazio, sem significado	
Object	Qualquer valor associado a um objeto	
Function	Permite armazenar uma função em uma variável.	

```
<HTML>
<HEAD>
	<TITLE>Meu primeiro script</TITLE>
</HEAD>
<BODY>
	<SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
		<!--
		var nome;
		var idade;

		nome = "João Pedro";
		idade = 20;

		document.write("Olá, me chamo " + nome);
		document.write("<br>");
		document.write("Tenho " + idade + " anos");
		-->
	</SCRIPT>
</BODY>
</HTML>
```

O código acima apresenta um exemplo de declaração e uso de variáveis. A variável “nome” recebe uma string, enquanto que a variável idade recebe um valor numérico. A partir da atribuições destes valores é que a variável passa a conhecer o seu tipo.

Ainda neste exemplo, assim como nos outros já vistos, percebe-se que o comando “document.write” tem por objetivo escrever informações no navegador. Neste momento é importante saber que este comando aceita apenas escrita de valores do tipo string e caso algum valor do tipo numérico seja usado com este comando, o mesmo será automaticamente convertido para uma string.

Como quem renderiza a informação deste comando é o próprio navegador, então, se na string forem inseridos comandos html, estes serão interpretados e exibidos. No exemplo acima a quebra de linha será interpretada pelo navegador, como pode-se observar na imagem abaixo,

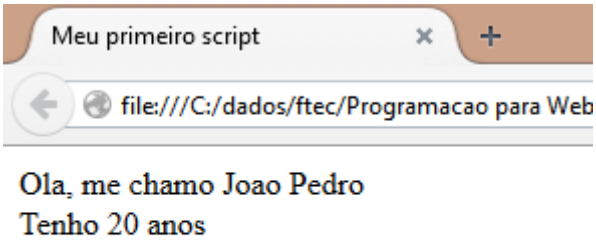

Figura 5 - Exemplo de JavaScript

Para ser possível utilizar as variáveis em uma linguagem de programação, deve-se acima de tudo saber como atribuir valores a estas variáveis. Até agora já vimos como realizar esta atribuição através do sinal de igual (=), porém existem ainda outras formas de fazermos estas atribuições conforme segue tabela.

Tabela 2 - Tabela de atribuições

Atribuição	O que faz
X = y	Define x com o valor de y
X += y	O mesmo que x = x + y
X -= y	O mesmo que x = x - y
X *= y	O mesmo que x = x * y
X /= y	O mesmo que x = x / y
X %= y	O mesmo que x = x % y

Além das variáveis e atribuições, uma linguagem de programação sempre necessita que sejam realizadas operações entre valores e variáveis. Estas operações podem ser matemáticas, lógicas ou envolvendo caracteres (strings) ou datas. Abaixo será apresentada uma lista de operações matemáticas que podem ser feitas pelo JavaScript.

Tabela 3 - Tabela de operações

Operador	O que faz
X + y (numérico)	Soma x e y
X + y (strings)	Contatena x e y. Exemplo: x = “Joao”, y = “ Pedro”. X + y = “Joao” + “ Pedro” = “Joao Pedro”.
X - y	Subtrai y de x
X * y	Multiplica x e y
X / y	Divide x e y
X % y	Resto da divisão de x e y
X++, ++x	Incrementa x em um.
X--. --x	Decrementa x em um.
-x	Inverte o sinal de x.

1.1.5 ITERAGINDO COM O USUÁRIO

Existem diversas maneiras de interagirmos ou obtermos informações do usuário através do JavaScript. Algumas das formas são mais simples e rápidas de serem utilizadas, enquanto que outras são mais complexas. Inicialmente serão abordadas técnicas simples de fazer esta iteração e nos próximos tópicos serão vistas formas mais complexas e corretas de fazer isso.

Para obtermos uma informação do usuário, podemos utilizar o “prompt”. Através do prompt é possível que o usuário informe um valor qualquer que possa ser utilizado no script.

Uma coisa importante que deve ser considerada é que qualquer informação obtida da interface com o usuário sempre será do tipo caractere (strings). Caso, deseje-se utilizar esta informação como um valor numérico, a mesma deve ser convertida para número através de uma função específica.

```
<HTML>
<HEAD>
	<TITLE>Meu primeiro script</TITLE>
</HEAD>
<BODY>
	<SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
		<!--
		var nome;
		var idade;

		nome = prompt("Informe o seu nome");
		idade = prompt("Informe a sua idade");

		idade = Number(idade);

		document.write("Olá, me chamo " + nome);
		document.write("<br>");
		document.write("Tenho " + idade + " anos");
		-->
	</SCRIPT>
</BODY>
</HTML>
```

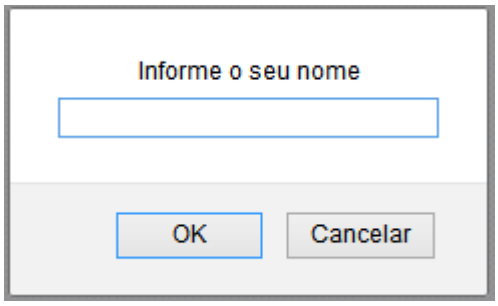

Figura 6 - Prompt solicitando o nome

Após o usuário informar o nome e clicar sobre o botão “OK”, o valor informado será atribuído a variável “nome”. Caso seja clicado no botão “Cancelar”, o valor retornado para a variável será “*null*”.

No caso da idade, quando for informado um valor para a idade o mesmo será tratado como uma string e posteriormente deverá ser convertido através do comando “Number” visto no exemplo acima.

Para retornar uma informação para o usuário pode-se empregar dois comandos no momento. É possível utilizar a função “*alert*”, a qual retorna uma mensagem na tela exigindo que o usuário pressione “*ok*” para que a mensagem possa ser fechada, ou então pode-se utilizar o comando “*document.write*”, que, como visto anteriormente, tem a função de escrever alguma coisa no corpo do documento html, aceitando inclusive comandos html.

A função "*alert*" não tem o seu uso recomendado para a iteração em websites com o usuário, porém, ela deve ser utilizada durante o processo de desenvolvimento, para que o programador possa verificar se a execução do script está correta. Ela pode ser utilizada como uma espécie de auxiliar para a depuração do código fonte.

```
<HTML>
<HEAD>
	<TITLE>Meu primeiro script</TITLE>
</HEAD>
<BODY>
	<SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
		<!--
		var nome;
		var idade;

		nome = prompt("Informe o seu nome");
		idade = prompt("Informe a sua idade");

		idade = Number(idade);

		alert("Olá, me chamo " + nome);
		alert("Tenho " + idade + " anos");
		-->
	</SCRIPT>
</BODY>
</HTML>
```

1.1.6 COMANDOS CONDICIONAIS E EXPRESSÕES

Os comandos condicionais são comandos que permitem que um determinado bloco de código seja executado somente se houver uma determinada condição que permita a sua execução. Esta condição sempre será uma expressão lógica, a qual irá retornar um resultado que poderá ser verdadeiro ou falso.

No JavaScript o comando condicional mais comum é o "*if*" (SE), que apresenta a seguinte sintaxe:

```
if (expressão)
{
	//Codigo executado quando a expressão for verdadeira
}
Else
{
	//Codigo executado quando a expressão for falsa
}
```

Uma expressão lógica é uma expressão que permite a realização de operações de comparação entre outras expressões, as quais podem ser novamente expressões lógicas ou então, expressões matemáticas.

Os operadores necessários para as comparações lógicas são apresentados na tabela abaixo:

Comparação	O que faz
X == y	Retorna verdadeiro se x e y são iguais
X != y	Retorna verdadeiro se x e y não são iguais
X > y	Retorna verdadeiro se x é maior que y
X >=y	Retorna verdadeiro se x é maior que ou igual a y
X < y	Retorna verdadeiro se x é menor que y
X <= y	Retorna verdadeiro se x é menor que ou igual a y
X && y	Retorna verdadeiro se x e y são ambas verdadeiras
X || y	Retorna verdadeiro se x ou y é verdadeira
!x	Retorna verdadeiro se x é falsa

```
<HTML>
<HEAD>
        <TITLE>Meu script de comparacao</TITLE>
</HEAD>
<BODY>
        <SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
                <!--
                var valor1;
                var valor2;

                valor1 = prompt("Informe o valor 1");
                valor2 = prompt("Informe o valor 2");

                valor1 = Number(valor1);
                valor2 = Number(valor2);

                if (valor1 < valor2)
                {
                   document.write(valor1 + " < " + valor2);
                }
                else
                {
                   document.write(valor1 + " < " + valor2);
                }

                -->
        </SCRIPT>
</BODY>
</HTML>
```

Outro comando condicional presenta no JavaScript é o "*switch*". Com este comando, pode-se selecionar uma operação a partir de diversas possibilidades. Abaixo segue um

exemplo do uso do comando switch para identificar qual é o dia da semana de acordo com o seu número.

```
<HTML>
<HEAD>
	<TITLE>Meu script de comparacao</TITLE>
</HEAD>
<BODY>
	<SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
		<!--
		var diaDaSemana;
		var diaPorExtensao;

		diaDaSemana = prompt("Informe o dia da semana");

		diaDaSemana = Number(diaDaSemana);

		switch (diaDaSemana) {
			case 0:
				diaPorExtensao = "Domingo";
				break;
			case 1:
				diaPorExtensao = "Segunda Feira";
				break;
			case 2:
				diaPorExtensao = "Terça Feira";
				break;
			case 3:
				diaPorExtensao = "Quarta Feira";
				break;
			case 4:
				diaPorExtensao = "Quinta Feira";
				break;
			case 5:
				diaPorExtensao = "Sexta Feira";
				break;
			case 6:
				diaPorExtensao = "Sábado";
				break;
			default:
				diaPorExtensao = "Dia Invalido";
		}
		alert(diaPorExtensao);
		-->
	</SCRIPT>
</BODY>
</HTML>
```

1.1.7 ESTRUTURAS DE REPETIÇÃO

Comandos de repetição são utilizados para repetir uma determinada parte de código do script um número finito de vezes; Para que uma estrutura de repetição posa ser criada, ela sempre necessita de três informações:

- O inicio do processo de repetição;
- O código a ser repetido;
- A condição de término da repetição.

No JavaScript existem pelo menos três tipos de comandos de repetição que precisam ser conhecidos neste momento:

For: Neste tipo de estrutura de repetição a definição do início, condição de término e incremento são definidos no início da estrutura.

While: Neste tipo de estrutura de repetição a condição de término é definida no início da repetição de forma que se a condição não for atendida, o bloco de código da estrutura de repetição nunca será executado;

Do-while: Neste tipo de estrutura de repetição a condição de término é definida no final do bloco de código, de forma que se a condição não for atendida o programa irá executar o bloco de código pelo menos uma vez.

```
<HTML>
<HEAD>
        <TITLE>Script de Repeticao</TITLE>
</HEAD>
<BODY>
        <SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
                <!--
                var i;

                //Sintaxe do comando for
                for (i = 0; i < 10; i++)
                {
                    document.write("Estou executando uma repeticao");
                }

                //Sintaxe do comando while
                i = 0; //Necessario inicializar a variavel contadora antes;
                while (i < 10)
                {
                    document.write("Estou executando uma repeticao");
                        i++; //Necessario incrementar o contado no bloco de
codigo;
                }

                //Sintaxe do comando do-while
                i = 0; //Necessario inicializar a variável
                do{
                    document.write("Estou executando uma repeticao");
                    i++; //Necessario incrementar o contado no bloco de codigo;
                }
                while (i < 10);

                -->
        </SCRIPT>
</BODY>
</HTML>
```

1.1.8 FUNÇÕES

Uma função é um bloco de programa encapsulado sob um nome, o qual pode ser executado quantas vezes forem necessárias, sem a necessidade de reescrever todo o bloco de código, sendo necessário apenas chamar a identificação deste bloco, ou seja, o nome. Em determinadas linguagens de programações também existem as sub-rotinas as quais executam uma ação semelhante à das funções.

Uma função pode receber parâmetros de entrada e poderá também retornar um resultado. Além disso, uma função também pode ter variáveis declaradas em seu interior, mas neste caso é importante ressaltar que uma variável criada dentro de uma função apenas poderá ser utilizada nesta função, não possuindo valor ou conteúdo fora dela. Enquanto que uma variável declarada fora da função pode ser utilizada em seu interior.

```
<HTML>
<HEAD>
	<TITLE>Exemplos de funcoes</TITLE>

	<SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
		<!--

		function ExibeAlerta() //Funcao sem parametros de entrada
		{
			alert("Alerta gerado pela funcao");
			//Esta funcao nao apresenta nenhum retorno de informacao
		}

		function ExibeAlertaParametro(mensagem)
		{
			alert(mensagem);
		}

		function Somar(valor1, valor2) //Funcao com dois parametros
		{
			var resultado;

			//A Soma é armazenada na variavel resultado
			resultado = valor1 + valor2;

			//O retorno do resultado é feito através da palavra
			//return;
			return resultado;
		}
		-->
	</SCRIPT>
</HEAD>
<BODY>
	<script>
		var retorno;

		//Retorna da funcao somar atribuido a variavel retorno
		retorno = Somar(2,3);

		//Exibicao da variavel retorno atraves da funcao ExibeAlerta...
		ExibeAlertaParametro(retorno);
	</script>
</BODY>
</HTML>
```

O código de exemplo da página anterior mostra três funções que foram criadas no cabeçalho do script. É importante perceber que apenas uma função criada não executa processamento algum. Ela somente irá executar processamento a partir do momento em que for chamada pelo script. Ainda no exemplo acima, pode-se perceber que as funções criadas foram chamadas no corpo da página html através de outro script.

Funções devem ser utilizadas de forma massiva nos sistemas para tornar o código dos mesmos mais organizado e legível, entretanto, deve-se sempre seguir algumas regras no momento da criação da função;

- Sempre criar funções pequenas (com pouco código);
- Uma função deve ter um único objetivo (Se a função deve fazer uma soma, ela não deve fazer uma multiplicação ou uma média além da soma);
- Por convenção o nome das funções sempre deve iniciar com a primeira letra maiúscula, sem o uso de underline (“_”) para separar nomes compostos. Ainda para os nomes compostos, cada novo nome deve ter a sua inicial maiúscula. Exemplo de nome de função: CalcularImposto, ValidarCpf, ...
- O nome da função sempre deve ter um verbo e deve refletir o objetivo da função.

1.1.9 ORIENTAÇÃO A OBJETOS EM JAVASCRIPT

Orientação a objetos é um paradigma de programação baseado na composição e interação de objetos para formar o programa. Um objeto pode ser entendido como uma abstração computacional de algo real. Exemplos de objetos podem ser um monitor, uma caneta, um cliente, um pedido, uma página html, etc....

Cada objeto é formado por propriedades e métodos. As propriedades definem as características de um objeto, enquanto que os métodos definem as ações de um objeto. Para o exemplo de um monitor, ele pode possuir como propriedades o tamanho, a cor, a resolução, etc, ..., enquanto que pode possuir como método a sua capacidade de ligar, desligar, alterar o brilho, alterar a nitidez, etc, ...

A descrição de um objeto em uma linguagem de programação é definida através de uma classe, enquanto que objeto em si é a criação desta classe em memória para que ela possa ser utilizada durante o programa.

Nem todas as linguagens são orientadas a objetos, entretanto, todas as linguagens modernas e atuais implementam os conceitos de orientação a objetos, inclusive o JavaScript.

Neste curso não iremos criar objetos próprios para serem utilizados, entretanto, iremos utilizar objetos que já são criados pelo próprio JavaScript.

No JavaScript, para acessarmos as propriedades e métodos de um objeto utilizamos o ponto (.) ao lado direito do objeto em questão. Um exemplo de objeto é o próprio “*document*”, o qual já foi utilizado anteriormente. Quando estamos utilizando o comando

"*document.write()*" estamos utilizando o objeto "*document*", o qual representa o documento html e estamos utilizando o seu método "*write*", o qual tem a capacidade de escrever aquilo que passarmos para ele no corpo do documento. Abaixo segue uma tabela com os principais objetos do JavaScript:

Objeto	**Descrição**
Documet	Objeto que armazena as características e métodos relacionados a página html.
Window	Este objeto representa uma janela do navegador que possui uma página aberta.
Date	Objeto que permite obter a data atual e também realizar operações com datas.
Array	O objeto Array é utilizado para armazenar múltiplos valores em uma única variável.
Style	O objeto Style permite o acesso as propriedades CSS de um determinado objeto html.
Math	Objeto utilizado para trabalhar com funções matemáticas
String	Objeto que permite armazenar uma cadeia de caracteres e também realizar operações sobre strings.
XmlHttpRequest	Este objeto permite comunicação assíncrona com o servidor através do uso do javascript. Com isso, é possível enviar e receber informações do servidor sem a necessidade de recarregar toda a página.

1.1.10 EVENTOS

Eventos são ações executadas mediante determinada atividade executada pelo usuário ou pelo próprio sistema. Estes eventos podem ser disparados quando o usuário clicar sobre um botão, movimentar o mouse, o sistema terminar de carregar a página em questão e diversas outras maneiras.

Os eventos são de extrema importância pois podemos associar a eles funções ou comandos JavaScript para que o programa execute algum procedimento sempre que o evento for diasparado.

Um exemplo disso é disparar uma mensagem quando o usuário clicar sobre um botão, ou então alterar informações da tela conforme o usuário for deslocando o cursor do mouse sobre a mesma.

O eventos gerados pelo JavaScript podem ser capturados através do uso de Handlers, os quais representam o evento no código, permitindo assim fazer uma associação do handler com a função a ser executada.

```
<HTML>
<HEAD>
        <TITLE>Meu primeiro evento</TITLE>
        <SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
                <!--

                function exibeAlerta()
                {
                        alert("Eu cliquei no botao!")
                }
                 -->
        </SCRIPT>
</HEAD>
<BODY>
        <FORM id="cadastro">
        <INPUT type="BUTTON" id="bt1" value="Clicar" onClick="exibeAlerta()">
        </FORM>
</BODY>
</HTML>
```

O exemplo acima apresenta o Handler “onClick”, de forma que quando for clicado no botão será executada a função “exibeAlerta”.

Abaixo segue tabela com a apresentação de diversos handlers que podem ser empregados em um script JavaScript.

Handler	**Tipo de Objeto**	**O que trata**
onAbort	Janela	O usuário encerrou o carregamento da página
onBlur	Janela, formulario	O usuário deixou o objeto
onChange	Formulario	O usuário alterou o objeto

onClick	Mouse, formulário	O usuário clicou em um objeto
onError	Janela	O script encontrou um erro
onfocus	Janela, formulario	O usuário tornou um objeto ativo
onLoad	Janela	O objeto conclui o carregamento
onMouseover	Mouse	O cursor se deslocou sobre o objeto
onMouseout	Mouse	O cursor se deslocou para fora de um objeto
onSelect	Formulário	O usuário selecionou o conteúdo de um objeto
onSubmit	Formulário	O usuário enviou um formulário
onUnload	Janela	O usuário saiu da janela

Uma situação muito importante, que precisa ser considerada quanto ao uso de JavaScript e que pode ser resolvida com o uso de eventos é o caso de quando o script precisa acessar um recurso html que ainda não tenha sido processado.

Esta situação acontece porque tanto o HTML como o JavaScript são linguagens processadas e por isso, se um script tentar acessar um recurso que ainda não tenha sido interpretado pelo navegador, este irá retornar uma falha de execução. O exemplo de script abaixo não funcionará.

```
<HTML>
<HEAD>
	<TITLE>Meu primeiro evento</TITLE>
	<SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
		<!--
		objetodiv.innerHTML = "conteudo...";
		-->
	</SCRIPT>
</HEAD>
<BODY>
	<div id="objetodiv">
	</div>
</BODY>
</HTML>
```

O que este código deveria fazer é escrever o texto “conteúdo...” no navegador quando a página for carregada, porém isso não irá acontecer, pois a div com id chamada “objetodiv” será carregada somente depois do script já ter sido executado, pois o script está acima do local de criação da div. O exemplo abaixo irá funcionar corretamente.

```
<HTML>
<HEAD>
	<TITLE>Meu primeiro evento</TITLE>
	<SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
		<!--
		function executa()
		{
			objetodiv.innerHTML = "conteudo...";
		}
		-->
	</SCRIPT>
</HEAD>
<BODY onload="executa()">
	<div id="objetodiv">
	</div>
</BODY>
</HTML>
```

Este exemplo funcionará pois a escrita do conteúdo está encapsulada em uma função e esta função somente será executada quando o evento “*onLoad*” do corpo da página for disparado. O evento “*onLoad*” é disparado somente depois que todo o conteúdo da página html terminar de ser interpretado e carregado pelo navegador.

1.1.11 APLICAÇÃO DE CSS COM JAVASCRIPT

Como visto no capítulo anterior, a formatação de uma página html não é feita pelo html, mas sim pelo CSS. Até o momento, foi apresentada apenas uma maneira de inserir CSS em uma página: Definindo a estrutura do CSS e vinculando a mesma a um determinado elemento, seja através do CSS in-line, incorporado ou linkado. Nestas situações, o CSS sempre tem a necessidade de ser criado de forma “fixa” na página.

Outra forma de inserir CSS em um elemento html é através do seu uso integrado ao JavaScript. Nesta situação, pode-se inserir o CSS de forma dinâmica na página, criando efeitos bastante interessantes em uma página, principalmente quando o seu uso estiver associado a captura de eventos em JavaScript. Para utilizar CSS com JavaScript deve-se utilizar o objeto “style” associado ao elemento html em questão.

O objeto “style” apresenta propriedades às quais são as próprias propriedades do CSS. Abaixo segue exemplo de uso deste objeto.

```
<html>
	<head>
		<script type="text/javascript" language="javascript">
			<!--
			function inicializa()
			{
				quadrado.style.position = "absolute";
				quadrado.style.width = "100px";
				quadrado.style.height = "100px";
				quadrado.style.top = "200px";
				quadrado.style.left = "10px";
				quadrado.style.backgroundColor = "red";
			}
			-->
		</script>
	</head>
	<body onload="inicializa()">
		<div id="quadrado"></div>
	</body>
</html>
```

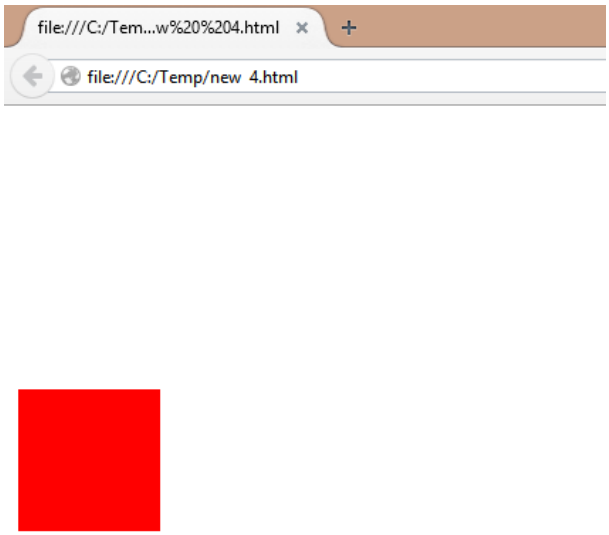

Figura 7 - Resultado do código acima

FONTES BIBLIOGRÁFICAS

[BKK, 2005] Breitmann, Karin Koogan. Web semântica: a internet do future – Rio de Janeiro: LTC, 2005

[SMS, 2008] Silva, Maurício Samy. Criando sites com HTML: Sites de alta qualidade com HTML e CSS – São Paulo: Novatec Editora, 2008

[KJ, 2009] Kalbach, James. Design de navegação; tradução: Eduardo Kessler Piveta. – Porto Alegre: Bookman, 2009

[SMS, 2011] SILVA, Maurício Samy. HTML 5: a linguagem de marcação que revolucionou a web. 1 ed.. São Paulo: Novatec, 2011.

[PM, 2011] Pilgrim, Mark. HTML 5: entendendo e executando. – Rio de Janeiro, RJ: Alta Books, 2011.

Wikipédia: a enciclopédia livre. Acesso em setembro de 214.

NEGRINHO, Tom; Smith, Dori. **JavaScript: Para a World Wide Web**. 4 ed. Rio de Janeiro: Campus, 2001