MARCOS DE MELO

JavaScript

1º Edição

JavaScript

Sumário

Aula 1 – Introdução

Desenvolver aplicações interativas para internet com o uso da linguagem de programação criada pela Netscape, criar validações de formulários e interações com o usuário através de aplicações *clientside*.

Introdução e Eventos

Tópicos da Aula

- O que é o JavaScript?
- História do JavaScript?
- Introdução
- JavaScript é o mesmo que Java?
- Como estudar e entender os exemplos?
- Sintaxe básica

O que é o JavaScript?

JavaScript é uma linguagem de programação criação pela Netscape, para trabalhar com validações simples no lado do cliente, usando interações em tempo real sem a necessidade de envio de dados para um servidor.

História do JavaScript

Quando o JavaScript apareceu pela primeira vez em 1995, seu principal objetivo era lidar com algumas das entradas de validação que anteriormente tinha sido deixado ao lado do servidor em linguagens como Perl. Antes disso o tempo de ida e volta para o servidor foi necessária para determinar se um campo obrigatório havia sido deixado em branco ou se um valor digitado era inválido. O Netscape Navigator tentou mudar isso com a introdução de JavaScript. A capacidade de lidar com algumas validações básicas no cliente era uma característica nova e excitante num momento em que o uso de modems de telefone foi generalizada. A associação as velocidades lentas virou fez com que cada viagem para o servidor fosse um exercício de paciência.

Desde aquela época, o JavaScript tornou-se uma característica importante de cada navegador principal no mercado. Não é mais obrigatória a validação de dados simples, JavaScript agora interage com quase todos os aspectos da janela do navegador e os seus conteúdos. JavaScript é reconhecido como uma completa linguagem de programação, capaz de cálculos complexos e interações, incluindo funções, e até metaprogramação.

O aumento de JavaScript a partir de uma simples entrada de validação para uma poderosa linguagem de programação não podia ter sido prevista. JavaScript é ao mesmo tempo muito simples e muito complicada, linguagem que leva alguns minutos para aprender, mas anos para dominar. Para começar o caminho para o uso Potencial do JavaScript, é importante compreender a sua natureza, história e limitações.

Introdução

A introdução de JavaScript em páginas web imediatamente ocorreu na linguagem HTML. Como parte de seu trabalho original em JavaScript, a Netscape tentou descobrir como fazer coexistir JavaScript em páginas HTML, sem quebrar as páginas de renderização em outros navegadores.

Através de erro, experimentação, e polêmica, várias decisões foram finalmente feitas e acordado para trazer suporte a scripts universais para a web. Muito do trabalho feito nestes primórdios da Web sobreviveu e ficou formalizado na especificação HTML até hoje.

JavaScript é o mesmo que Java?

A linguagem JavaScript é filha da linguagem Java, ela é muito semelhante mais não é a mesma, após concluir o curso de JavaScript você terá facilidade em começar a programar em Java.

Como estudar e entender os exemplos?

Existe uma nomenclatura básica que pode lhe auxiliar na criação de scripts com JavaScript, existem quatro objetos principais que são criados ao carregar uma página, estes objetos fazem toda diferença na hora do estudo.

Veja os quatro objetos abaixo:

window: O primeiro objeto da hierarquia JavaScript acima dele só o navegador, suas propriedades são aplicadas a todas as janelas, mesmo objetos de um documento com frames.

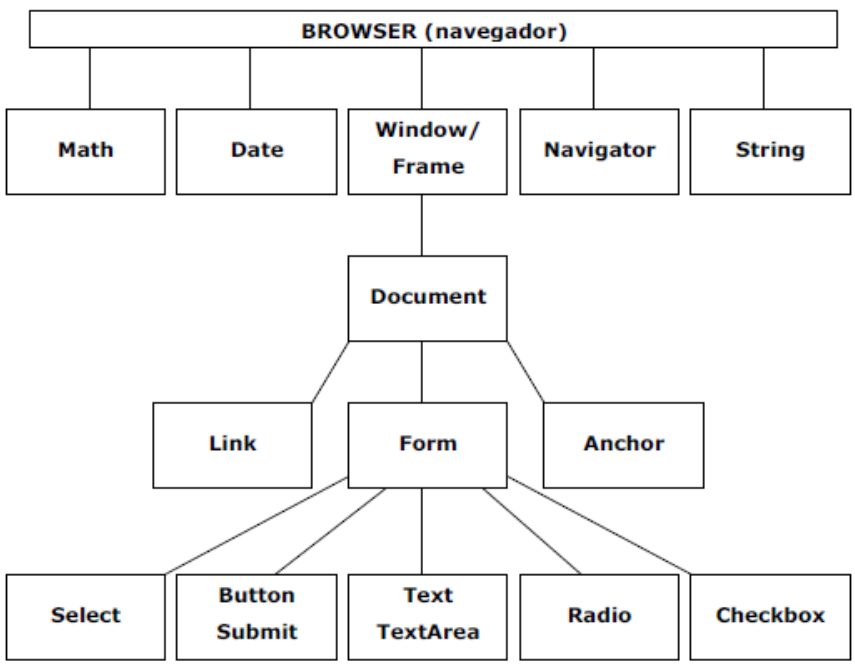
location: Possui propriedades da URL atual.

history: Possui propriedades das URLs visitadas.

document: Possui propriedades do documento contido na janela, tais como o seu formulários, conteúdos, títulos, cores, css, etc.

Estrutura detalhada

Observe abaixo a hierarquia completa do JavaScript.



Veja como descrito, que cada objeto possui sua propriedade, mesmo o objeto Form que tem propriedades como Button e Text, todos estes são propriedades de Document.

Sintaxe básica

A utilização de JavaScript é muito semelhante a utilização de HTML, no HTML temos a tag <html> e dentro dela temos tags como <head> e <body>, podemos dizer que estas duas são propriedades do <html>, no JavaScript isso seria como o objeto document e as suas propriedades form e link.

Um código de utilização JavaScript é bastante simples, com poucas linhas já podemos criar uma interação com o usuário.

Veja sua sintaxe:

```
nomeObjeto.propriedade
```

ECMAScript

O JavaScript é definido como uma linguagem ECMAScript, a linguagem definida em ECMA-262, não está vinculado a navegadores web. Na verdade, a linguagem não tem métodos para entrada ou saída alguma. ECMA-262 define esta linguagem como uma base sobre a qual as linguagem mais robustas de script podem ser construidas (Exemplo Java).

Navegadores da Web são apenas um meio de acolhimento em que uma aplicação ECMAScript pode existir. Um ambiente de servidor fornece a base de ECMAScript e implementação de extensões destinadas a interface com o ambiente em si. Extensões, como o Document Object Model (DOM), usa ECMAScript do tipo básico e uma sintaxe para fornecer funcionalidade adicional que é mais especificamente para o ambiente.

Ambientes de servidor incluem um Node.js que gerencia o lado do servidor da plataforma para controle de JavaScript e aplicações com Adobe Flash.

Partes da linguagem ECMA-262

O que significa exatamente que o ECMA-262 não faz referência a navegadores? Em um nível muito básico, que descreve as seguintes partes da linguagem:

- Sintaxe
- Tipos
- Demonstrações
- Palavras-chave
- As palavras reservadas
- Operadores
- Objetos

ECMAScript é simplesmente uma descrição de uma linguagem de execução de scripts e implementaçãoes. JavaScript implementa ECMAScript, assim também como o Adobe ActionScript utilizado no Flash.

O Document Object Model (DOM)

O Document Object Model (DOM) é uma interface de programação de aplicações (API) para XML que foi estendido para uso em HTML. Os mapas DOM formam uma página inteira como uma hierarquia de nós. cada parte de uma página HTML ou XML é um tipo de nó que contém diferentes tipos de dados.

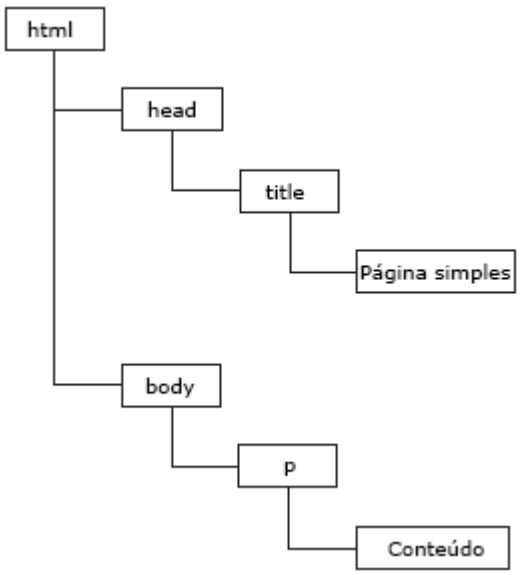
Veja a estrutura abaixo da página HTML:

```
<html>
  <head>
    <title>Página simples</title>
  </head>
  <body>
    <p>Conteúdo</p>
  </body>
</html>
```

Este código pode ser diagramado em uma hierarquia de nós usando o DOM. Ao criar uma árvore para representar um documento, o DOM permite aos desenvolvedores um nível sem precedentes de controle sobre seu conteúdo e estrutura.

Nós podem ser removidos, adicionados, substituídos, e modificados facilmente usando a API DOM.

Veja abaixo a hierarquia de nós.



Por que o DOM é Necessário

Com o Internet Explorer 4 e Netscape Navigator 4 cada um deu suporte diferente as formas de HTML dinâmico (DHTML), os desenvolvedores pela primeira vez podiam alterar a aparência e o conteúdo de uma página web sem recarregar. Isto representou um enorme passo para a tecnologia web, mas também um grande problema. Netscape e Microsoft seguiram caminhos separados em desenvolvimento de DHTML, no final do período e analise dos navegadores os desenvolvedores podem escrever uma única página HTML que pode ser acessado por qualquer navegador web.

Decidiu-se que algo tinha que ser feito para preservar a natureza multi-plataforma da Web. O medo era que se alguém não tivesse o navegador Netscape ou o navegador da Microsoft, a Web iria se desenvolver em duas facções distintas que eram exclusivas para os navegadores alvo. Foi então que a World Wide Web Consortium (W3C), órgão encarregado de criar padrões para a comunicação web, começou a trabalhando no DOM.

Versões do JavaScript

O navegador Mozilla, como um descendente do Netscape original, é o único navegador que continuou a sequência de numeração de versão do JavaScript original. Quando o código fonte do Netscape foi desmembrada em um projeto open-source (nomeado o Projeto Mozilla), o navegador suportava a versão do JavaScript 1.3 À medida que o Mozilla Foundation continuou a trabalhar em JavaScript, adicionando novas funcionalidades, palavras-chave e sintaxes, o número da versão JavaScript foi incrementado. A tabela a seguir mostra a versão do JavaScript e a progressão no Netscape/Mozilla.

Browser	**Versão do JavaScript**
Netscape Navigator 2 1.0	Netscape Navigator 2 1.0
Netscape Navigator 3 1.1	Netscape Navigator 3 1.1
Netscape Navigator 4 1.2	Netscape Navigator 4 1.2
Netscape Navigator 4.06 1.3	Netscape Navigator 4.06 1.3
Netscape 6+ (Mozilla 0.6.0+) 1.5	Netscape 6+ (Mozilla 0.6.0+) 1.5
Firefox 1 1.5	Firefox 1 1.5
Firefox 1.5 1.6	Firefox 1.5 1.6
Firefox 2 1.7	Firefox 2 1.7
Firefox 3 1.8	Firefox 3 1.8
Firefox 3.5 1.8.1	Firefox 3.5 1.8.1
Firefox 3.6 1.8.2	Firefox 3.6 1.8.2

É importante notar que somente os navegadores Netscape/Mozilla siguem este esquema de versionamento. O Internet Explorer, por exemplo, tem versão diferente para números de JScript. Estas versões de JScript não correspondem a qualquer versão JavaScript mencionadas na

tabela anterior. Além disso, a maioria dos navegadores fala sobre o suporte a JavaScript em relação ao seu nível de ECMAScript em conformidade e suporte ao DOM.

Como adicionar JavaScript em uma página

Navegadores da Web são construídos para entender de HTML e CSS e converter estas linguagens em uma exibição visual na tela. A parte do navegador que entende HTML e CSS é chamado de mecanismo de layout ou motor de renderização.

Mas a maioria dos navegadores também tem um interpretador de JavaScript. Essa é a parte do navegador que entende JavaScript e pode executar os passos de um programa JavaScript. Uma vez que o navegador web geralmente espera somente HTML, você deve especificar quando o navegador JavaScript deve usar a tag <script>.

A tag <script> é padrão HTML. Ele age como um interruptor que na verdade diz "Olá navegador, ai vem um código JavaScript", você não sabe o que fazer com ele, assim entregue para o interpretador de JavaScript." quando o navegador encontra a tag de fechamento </script> ele sabe que atingiu o fim do programa de JavaScript e pode executar de volta às suas funções normais.

Grande parte do tempo, você vai adicionar a tag <script> dentro da tag <head> da página HTML.

Veja um o exemplo:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Minha Página</title>
  <script type="text/javascript">
  </script>
</head>
<body></body>
</html>
```

O atributo tag <script> indica o formato e o tipo de roteiro que se segue. Neste caso, type = "text/javascript" significa que o roteiro é um texto regular (como HTML) e que é escrito em JavaScript.

Se você estiver usando HTML5, a vida é ainda mais simples. Você pode pular o atributo ***tipo*** inteiro:

Veja o exemplo:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Minha Página</title>
   <script>
   </script>
</head>
<body></body>
</html>
```

Na verdade, os browsers permitem que você deixe de fora o atributo do tipo em HTML 4.01 e XHTML 1.0, porém, sua página não irá validar corretamente sem o atributo do tipo.

Este livro usa HTML5 para o doctype, mas o código JavaScript será o mesmo e funcionam da mesma forma para HTML 4.01 e XHTML 1.

Em seguida, adicione o código JavaScript entre a abertura e fechamento de <script>:

Veja o exemplo:

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Minha Página</title>
    <script>
       alert('Bem vindo!');
     </script>
  </head>
<body></body>
</html>
```

Você vai descobrir o que realmente faz este JavaScript em outro momento. Por enquanto, concentre sua atenção para as marcas <script> abertura e fechamento. Para adicionar um script à sua página, comece inserindo essas marcas. Em muitos casos, você vai colocar as tags <script> na página entre as tags <head></head>, a fim de manter o seu código JavaScript bem organizado em uma área da página web.

No entanto, é perfeitamente válido colocar tags <script> em qualquer lugar dentro da página HTML. Na verdade, como você verá mais adiante neste livro, há um comando JavaScript que permite você gravar informações diretamente em uma página web. Usando esse comando, você coloca o <script> dentro da página (em algum lugar dentro do corpo), onde você queira que o script escreva sua mensagem.

É ainda comum colocar tags <script> logo abaixo do fechamento da tag </body>, esta abordagem faz que a página seja carregada e que o visitante veja toda a página antes de executar qualquer JavaScript.

Objeto Window

Podemos alterar as propriedade de uma janela através do objeto window. Ao abrir uma página web um objeto window é criado automaticamente no navegador, e isso é feito sem que você tenha que alterar nada no HTML.

Este objeto possibilita a criação e abertura de novas janelas de formas diferentes.

Veja o exemplo:

```
window.propriedade
window.método
```

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Objeto Window</title>
<script>
function novaJanela(url,nomejanela,recursos) {
  window.open(url,nomejanela,recursos);
}
</script>
</head>
```

```
<body>
<a href="#"
onClick="MM_openBrWindow('http://www.google.com.br','novoja
nela','width=500,height=500')">Clique</a>
</body>
</html>
```

Objeto window/frame

Este objeto possui propriedades que podem modificar a janela do browser e seus frames caso possua.

Propriedades	Descrição
frames[]	Cria um Array de frames.
length	Informa a quantidade de itens.
name	Informa o nome do objeto.
parent	Indica o frame ou janela pai.
self	Indica o frame atual.
top	Indica o frame ou superior.
window	Indica o frame ou janela pai.
status	Exibe uma mensagem na barra de status do navegador
defaultStatus	Define uma mensagem padrão na barra de status do navegador.

Veja o exemplo:

As propriedades window e parent podem ser identificada neste exemplo.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Objeto Window/Frame</title>
<script>
function outraUrl() {
  var i, args=outraUrl.arguments; document.retorno = false;
  for (i=0; i<(args.length-1); i+=2)
eval(args[i]+".location='"+args[i+1]+"'");
}
</script>
</head>
<body>
<input name="pagina" type="button"
onClick="outraUrl('parent','http://www.adobe.com');return
document.retorno" value="Adobe">
</body>
</html>
```

Métodos

Os objetos criados pela linguagem JavaScript pode conter funções pré-definidas chamadas de métodos, estes podem tem a função de executar operações determinadas pelo programador.

Estes métodos são considerados atributos do objeto, o programador pode receber e alterar valores em tempo de execução, não tente passar métodos onde não é necessário, isso criará erro na saída do JavaScript.

Veja exemplo:

```
nome_objeto.método(valor)
```

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Metodo</title>
<script>
function AbrirMensagem(valor){
window.alert(valor);
}
</script>
</head>
<body onLoad="AbrirMensagem('Olá seja bem vindo!');">
</body>
</html>
```

No exemplo acima `nome_objeto` é a referência ao objeto que será alterado pelo método, o médoto possui uma identificação de passagem de parâmetros representado pelo sinal de parênteses, o valor do argumento pode ser ou não nulo, isso significa que determinados métodos vão possuir valores obrigatórios e outros não.

Métodos do objeto document

Temos cinco médotos principais do objeto document, veja abaixo a descrição e utilização de cada item.

Método clear

O método clear limpa a tela da janela atual.

Ex: document.clear();

```
Texto incluso no corpo de um documento HTML
<input type=”button” value=”Limpar”
onClick=”document.clear()” />
```

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Clear</title>
<script>
        function Clear () {
            document.open();
            document.close();
        }
    </script>
</head>
<body>
    <p><img
src="http://www.google.com/images/srpr/logo3w.png"
alt="Google" width="275" height="95"></p>
    <button onclick="Clear();">Limpar conteúdo deste
documento!</button>
</body>
</html>
```

Método close

O método close() fecha a janela do navegador web, este método foi alterado e agora não é mais o **document** que atribui o médoto e sim o objeto **window**.

Ex: document.close();

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
```

```
<title>Close</title>
<style type="text/css">
#info {
      background-color: #F0DB4F;
      height: 300px;
      width: 300px;
}
</style>
<script>
function ocultarConteudo() {
      document.getElementById('info').style.visibility =
'hidden';
}
function mostrarConteudo() {
      document.getElementById('info').style.visibility =
'visible';
}
</script>
</head>
<body>
<p>
  <input name="ocultar" type="button"
onClick="ocultarConteudo()" value="Ocultar Imagem">
  <input name="mostrar" type="button"
onClick="mostrarConteudo()" value="Mostar Imagem">
</p>
<div id="info"><img
src="http://upload.wikimedia.org/wikipedia/commons/thumb/6/
6a/JavaScript-logo.png/200px-JavaScript-logo.png" alt="JS"
width="200" height="200"></div>
<p>
  <input type="submit" name="fechar" id="fechar"
value="Fechar o Browser" onClick="window.close();">
</p>
</body>
</html>
```

Método write e writeln

Podemos inserir textos em nosso documento a qualquer momento com estes métodos, eles são muito parecidos, a diferença principal é que o método writeln insere uma nova linha ao final da string. Esta nova linha é ignorada pelo HTML, as únicas tags que farão diferença são tags <pre> e <textarea> .

Entre estas duas a mais conhecida e utilizada por programadores é o método write do objeto document.

Veja exemplo:

```
document.write(texto);
document.writeIn(texto);
```

Você pode inserir o que quiser dentro do argumento deste método, é possível criar cadeias de texto que devem ser atribuídas entre aspas, ou atribuições de valores matemáticos, como você pode perceber, as possibilidades são inúmeras, é possível criar uma página inteira com estes métodos.

Veja exemplo:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Write</title>
<script>
      document.write("<title>Bem Vindo</title>");
      document.write("<h1>Obrigado pela Visita</h1>");
      document.write("<textarea>Curso ");
      document.write("JavaScript</textarea>");
</script>
</head>
<body>
<p>Exemplo utilizando document.write()</p>
</body>
</html>
```

Observe agora o uso do método writeln que permitirá ao texto criado no tag <textarea> a quebra de linha entre eles:

```
document.writeln("<textarea>Curso ");
document.writeln("JavaScript</textarea>");
```

Veja outro exemplo a seguir da apresentação de um cálculo sendo exibido através do método write.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Write - Calculo</title>
<script>
document.write("O resultado de 5 + 2 é: ", 5 + 2);
</script>
</head>
<body></body>
</html>
```

Neste caso o navegador web vai primeiramente fazer o calculo e após o calculo exibir a mensagem na tela, todo este processo ocorre muito rapidamente por isso você só vera a mensagem final.

Objeto document

Contém informações gerais sobre o documento atual, este objeto é uma propriedade do objeto window.

Veja exemplo:

```
document.propriedade
document.método()
```

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Document</title>
<script type="text/javascript">
document.title = "Este é o título do documento";
alert(document.lastModified);
</script>
</head>
<body></body>
</html>
```

Propriedades do objeto document

Estas propriedades são relacionadas ao objeto document e podem fazer alterações na página em tempo de execução.

- alinkColor – Define a cor de um link que foi acionado.
- anchors[] – Lista de âncoras do documento.
- bgColor – Cor de fundo da página.
- cookie – Armazena informações do usuário.
- defaultStatus - Especifica um texto para a barra de status do navegador.
- fgColor - Especifica o valor do atributo TEXT, que apresenta a cor dos textos presentes no documento.
- forms[] - Array que contém todos os formulários do documento.
- lastModified - Apresenta a data da última modificação feita no documento.

- linkColor - Especifica o valor do atributo LINK, que determina a cor dos links não visitados.
- links[] - Array que contém os hiperlinks do documento.
- location - Especifica a URL completa do documento corrente.
- referrer - Especifica a URL que originou o documento corrente.
- status - Especifica o texto atual da barra de status do navegador.
- title - Especifica o valor do atributo TITLE do documento.
- vlinkColor - Especifica o valor do atributo VLINK, que determina a cor dos links visitados.

Abaixo você vê um exemplo simples de utilização da tag bgcolor, onde você receberá um alerta mostrando a cor atual do documento..

```
<script>
alert(document.bgColor);
</script>
```

No exemplo a seguir foi usada a propriedade bgColor no evento onMouseover que mudará a cor de fundo do documento assim que o ponteiro do mouse passar sobre o nome de uma cor.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>bgColor</title>
<script>
function novaCor(cor) {
      document.bgColor=cor;
}
</script>
</head>
<body>
<pre>
<a href onMouseover="novaCor('blue')">Azul</a>
<a href onMouseover="novaCor('red')">Vermelho</a>
<a href onMouseover="novaCor('green')">Verde</a>
<a href onMouseover="novaCor('pink')">Rosa</a>
<a href onMouseover="novaCor('silver')">Cinza</a>
```

```
<a href onMouseover="novaCor('purple')">Púrpura</a>
<a href onMouseover="novaCor('orange')">Laranja</a>
<a href onMouseover="novaCor('magenta')">Magenta</a>
<a href onMouseover="novaCor('yellow')">Amarelo</a>
<a href onMouseover="novaCor('black')">Preto</a>
</pre>
</body>
</html>
```

Já neste outro exemplo foi criado quatro botões de radio que ao clicar sobre um dos botões, a cor de fundo da página é alterada.

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>bgColor</title>

</head>

<body>

<pre>

<input type="radio" name="grupo" onClick="document.bgColor='blue';">Azul

<input type="radio" name="grupo" onClick="document.bgColor='red';">Vermelho

<input type="radio" name="grupo" onClick="document.bgColor='yellow';">Amarelo

<input type="radio" name="grupo" onClick="document.bgColor='silver';">Cinza

</pre>

</body>

</html>
```

Questionário

1. Por qual empresa foi desenvolvida a linguagem JavaScript?

_____
_____
_____
_____

2. O que é Javascript?

_____
_____
_____
_____

3. O que o Objeto Window permite que o usuário crie?

_____
_____
_____
_____

4. O que são eventos?

_____
_____
_____
_____

5. O que é possível com o método prompt?

_____
_____
_____
_____

Aula 2 -Elementos e Janelas

Tópicos da Aula

- Utilizando Eventos
- Objeto document
- Elementos da Linguagem
- Trabalhando com janelas

A Gramática de JavaScript

Aprender uma linguagem de programação é muito parecido com o aprendizado de qualquer idioma:

Há palavras para aprender, e um novo conjunto de regras. E assim como você precisa aprender a gramática da língua francesa a para falar francês, você deve se familiarizar com a gramática de JavaScript para programar em JavaScript.

Este capítulo aborda os conceitos que todos os programas JavaScript utilizam.

Se você já teve alguma experiência com programação JavaScript, muitos destes conceitos podem parecer parecidos com os de outras linguagens, e pode ser que você já tenha alguma familiaridade com estes códigos. Mas se você é um novo programador, ou você ainda não tem certeza sobre os fundamentos, este capítulo apresenta uma básica (mas crucial) introdução.

Demonstrações

Uma declaração de JavaScript é uma unidade básica de programação, geralmente representando uma única passo em um programa em JavaScript. Pense em uma declaração como uma sentença: Assim como você junta frases em um conjunto para criar um parágrafo (ou um capítulo, ou um livro), você combina declarações para criar um programa em JavaScript. No último capítulo, que viu vários exemplos de declarações. Por exemplo:

```
alert('Bem vindo!');
```

Esta declaração só abre uma janela de alerta com a mensagem "Bem vindo!" Nele. Em muitos casos, uma declaração é uma única linha de código. Cada instrução termina com um ponto e vírgula e é como um período no final de uma frase. O ponto e vírgula deixa claro que o passo é longo e que o interpretador de JavaScript deve passar para a próxima ação.

***Nota**: Oficialmente, colocar um ponto e vírgula no fim de uma instrução é opcional, e alguns programadores não colocam para tentar deixar o seu código mais curto ou por preguiça mesmo. Não seja um deles. Deixando de fora o ponto e vírgula torna a leitura do código mais difícil e, em alguns casos, podem ocorrer erros no seu JavaScript.*

Seu primeiro programa JavaScript

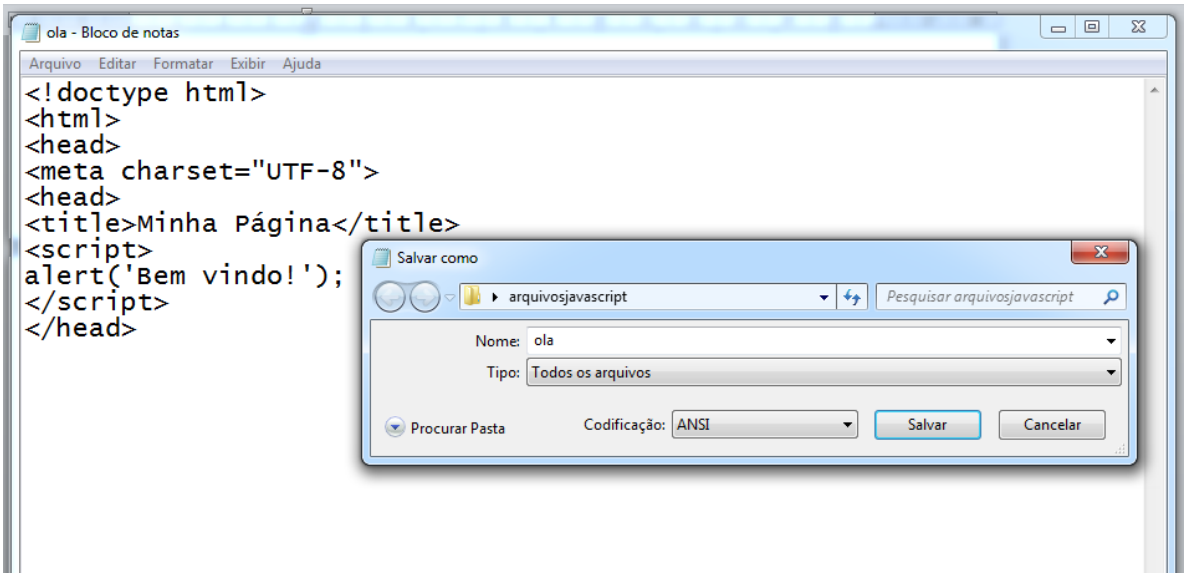
A melhor maneira de aprender programação JavaScript é programando. Ao longo Neste livro, você encontrará exemplos práticos que vão levá-lo passo a passo através do processo de criação de programas de JavaScript. Para começar, você precisará de um editor de texto e um navegador web.

Para fornecer uma introdução suave em JavaScript, seu primeiro programa será muito simples:

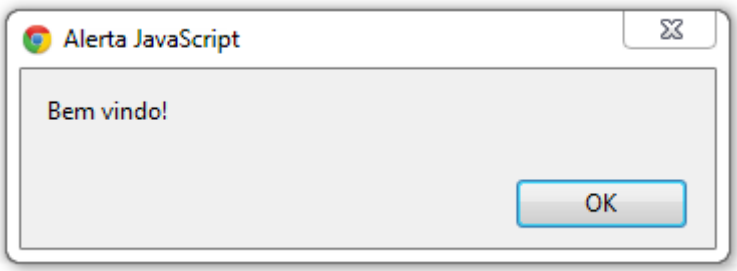
1. Abra seu editor de texto favorito e adicione o código abaixo:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Minha Página</title>
  <script>
    alert('Bem vindo!');
  </script>
  </head>
<body><body>
</html>
```

2. Salve seu arquivo como ola.html



3. Abra este arquivo e veja o que acontece.



4. Clique no botão OK da caixa de alerta para fechá-lo.

Quando a caixa de alerta desaparece, a página web aparece na janela do navegador.

Escrevendo texto na página web

O último script mostrou uma caixa de diálogo no meio do seu monitor. E se você deseja imprimir uma mensagem diretamente na página web usando JavaScript? Há muitas maneiras de fazê-lo, e você vai aprender algumas técnicas sofisticadas, mais adiante neste livro.

No entanto, você pode alcançar este objetivo simples, com um simples comando JavaScript, e é isso que você vai fazer no seu segundo script:

1. Abra seu editor de texto e adicione o código abaixo:

```
<!doctype html>
<html>
```

```
  <head>
    <meta charset="UTF-8">
    <title>Minha Página</title>
  </head>
  <body>
  <script>
    document.write('<p>Bem vindo!</p>');
  </script>
  </body>
</html>
```

2. Salve seu arquivo como ola2.html

Como a função alert(), document.write() é um comando JavaScript que literalmente escreve o que você colocar entre a abertura e fechamento de parênteses.

Neste caso, o HTML Bem vindo! é adicionado à página.

Os dois scripts que você acabou de criar podem deixar você se sentindo um pouco impressionado com JavaScript... ou com este livro. Não se preocupe. É importante começar com uma compreensão completa dos princípios.

Você estará fazendo algumas coisas muito úteis e as vezes complicadas. Na verdade, no restante deste capítulo, você vai obter alguns dos recursos avançados e vai ser capaz de adicionar a suas páginas da web.

Recebendo informações

No último script, você viu como criar variáveis, mas você não começa a sentir como as variáveis podem responder ao usuário e produzir conteúdo exclusivo e personalizado. Em este próximo exercício, você aprenderá como usar o comando prompt() para recolher contributos de um usuário e alterar a exibição da página com base nessa entrada.

1. Em um editor de texto, crie um arquivo chamado **prompt.html**.

2. Crie uma estrutura básica de página HTML, e entre as tags <head> e </head> adicione o seguinte código:

```
<script>
```

```
var nome = prompt ('Qual é seu nome?','');
</script>
```

A função prompt () produz uma caixa de diálogo semelhante à função alert (). No entanto, em vez de apenas exibir uma mensagem, a função prompt () também pode recuperar uma resposta. Além disso, para utilizar a função de alert(), você deve fornecer duas strings separadas por uma vírgula entre os parênteses. A primeira sequência aparece como o questão da caixa de diálogo ("Qual é seu nome?" neste exemplo).

```
prompt('Qual é o seu nome?', " );
```

A função prompt () é uma maneira de recuperar dados de entrada do usuário. Ele deve ser criado passando duas cadeias de caracteres, a primeira é o título da janela do prompt e o segundo é um texto que pode ser passado como predefinido.

Este exemplo utiliza uma string vazia, que é representada com apenas duas aspas simples (''). No entanto, você pode fornecer uma instrução útil como "Por favor digite seu nome e sobrenome", infelizmente, o visitante terá que primeiro excluir o texto, e depois digitar seus dados.

A função prompt() retorna um string contendo o que o visitante digitou na caixa de diálogo. Nessa linha de código JavaScript, que resultado é armazenado em um novo nome variável.

Neste exemplo, a função prompt () retorna um string que você armazene no nome da variável.

3. Salve a página e visualize em um navegador web.

Quando a página é carregada, você verá uma caixa de diálogo. Observe que nada acontece, preencha a caixa de diálogo e clique em OK.

Você também vai notar que nada acontece após clicar em OK, isso é porque, neste momento, é apenas coletadas e armazenada a resposta.

4. Volte para o seu editor de texto. Localize o conjunto de marcas <script> e adicione o código em negrito:

```
<script>
var nome = prompt ('Qual é seu nome?','');
document.write ('<p> Bem-vindo' + nome + '</ p>');
</script>
```

Aqui você aproveita as informações fornecidas pelo visitante. Você está combinando strings, unindo o conteúdo do prompt, escrevendo na página e misturando string com variável.

5. Salve a página e visualize em um navegador web.

Quando a caixa de diálogo Prompt aparecer, digite um nome e clique OK. Note que o nome que você digita aparece na página web. Recarregue a página web e digite um novo nome, nome que ele muda, assim como uma boa variável deveria fazer.

O poder das variáveis: Esta página customiza sua mensagem, com base em uma resposta do visitante.

Funções internas

JavaScript e navegadores permitem que você use vários comandos para fazer as coisas acontecerem em seus programas e em suas páginas web. Estes comandos, chamadas funções, são como verbos em uma sentença. Eles fazem as coisas acontecerem. Por exemplo, a função alert() que você usou anteriormente faz o navegador abrir uma caixa de diálogo e exibir uma mensagem.

Algumas funções, como alert() ou document.write (), que você também já usou, são específicos para navegadores web. Em outras palavras, eles só funcionam com páginas web, de modo que você não vai encontrá-los quando for programar em outros ambientes que usam JavaScript (como, por exemplo, quando for inserir um script em aplicações Adobe Acrobat ou no Dreamweaver ou no ActionScript do Flash que é baseado em JavaScript).

Outras funções são universais e o JavaScript e podem ser trabalhadas em qualquer programação baseada em JavaScript. Por exemplo, isNaN() é uma função que verifica se um determinado valor é um número ou não, esta função é útil quando você quiser verificar se um visitante digitou corretamente um número para uma pergunta que requer uma resposta numérica (por exemplo, "Quantos anos você tem?"). Você vai aprender sobre isNaN() e como usá-lo mais a frente.

O JavaScript tem muitas funções diferentes, que você vai aprender neste livro. Um modo rápido para identificar uma função de um programa é o uso de parênteses. Por exemplo, você pode dizer isNaN() é uma função por causa dos parênteses, assim como alert().

Além disso, JavaScript permite criar suas próprias funções, de modo que você pode fazer os seus scripts fazerem coisas além do que os comandos padrão de JavaScript oferecem.

Tipos de Dados

Você pode lidar com diferentes tipos de informação a cada dia. Seu nome, o preço dos alimentos, o endereço do consultório do seu médico, e a data do seu próximo aniversário, são informações importantes para você.

Você toma decisões sobre o que fazer e como viver sua vida com base nas informações que você tem. Programas de computador não são diferentes. Eles também contam com informações para fazer as coisas. Por exemplo, para calcular o total para um carrinho de compras, você precisa saber o preço e a quantidade de cada item solicitado.

Para personalizar uma página da Web com o nome de um visitante (exemplo: "Bem vindo, Fabio"), você precisa saber o nome dele.

Linguagens de programação normalmente categorizam a informação em diferentes tipos, e tratam cada tipo de uma maneira diferente. Em JavaScript, os três tipos mais básicos de dados são número, string e boolean.

Number (números)

Números são usados para contar e calcular, você pode manter o controle do número de dias que faltam até que as férias de verão, ou calcular o custo da compra de dois bilhetes para um filme.

Números são muito importantes na programação JavaScript: Você pode usar números para manter o controle de quantas vezes um visitante entrou em uma página web, para especificar a posição exata do pixel de um item em uma página web, ou para determinar quantos produtos um visitante quer comprar.

Em JavaScript, um número é representado por um caractere numérico, 5, por exemplo, é o número cinco. Você também pode usar números fracionários com decimais, como 5,25 ou 10,3333333. JavaScript ainda permite que você use números negativos, como o -130.

Como os números são frequentemente utilizados para os cálculos, seus programas, muitas vezes, incluem operações matemáticas. Você vai aprender sobre operadores mais adiante, mas apenas para fornecer um exemplo do uso de JavaScript com números, digamos que você queira imprimir o valor total de 5 mais 15 em uma página web, você poderia fazer isso com esta linha de código:

```
document.write(5 + 15);
```

Este trecho de JavaScript adiciona os dois números e imprime o total (20) em uma página web. Há muitas maneiras diferentes de trabalhar com números, e você vai aprender mais sobre eles neste livro.

String (textos)

Para exibir um nome, uma frase, ou qualquer série de palavras, você usa strings. Uma string é apenas uma série de letras e outros símbolos fechados dentro de aspas. Por exemplo, "Bem vindo Fabio", e "Você está aqui" são exemplos de cadeias de string.

Você usou uma string no último capítulo com o alerta de comando de alert("Bem vindo! '). Um aspas de abertura das citações é uma indicação para o interpretador de JavaScript que o que se segue é uma sequência de apenas uma série de símbolos. O intérprete aceita os símbolos literalmente, em vez de tentar interpretar a seqüência como algo especial para JavaScript como um comando.

Quando o interpretador encontra a marca de citação final, ele entende que chegou ao fim da cadeia de caracteres e continua até a próxima parte do programa.

Você pode usar tanto aspas duplas ("Bem vindo") ou aspas simples (‘Bem Vindo’) para colocar os caracteres, mas você deve certificar-se de usar o mesmo tipo de citação para marcar o início e o final da cadeia (por exemplo, "isso não está certo', não é válida porque a string começa com uma marca de aspas duplas, mas termina com uma citação de aspas simples).

***Nota**: Você vai notar que, no corpo principal deste livro, usamos aspas “” e ‘’, nos códigos JavaScript utilizaremos aspas simples (como alert ('Atenção aviso!');), não se preocupe você pode criar a sua regra, mas lembre de segui-la.*

Então, a janela pop-up de um alerta com uma mensagem de aviso de advertência, você poderia escrever:

```
alert ('Atenção!');
```

ou

```
alert ("Atenção!");
```

Você vai usar strings frequentemente em sua programação ao adicionar mensagens de alerta, quando se lida com a entrada do usuário em formulários web, e a o manipular o conteúdo de uma página web. Elas são tão importantes que você vai aprender muito mais sobre como usar strings mais adiante.

Booleans (verdadeiro ou falso)

Considerando que números e strings oferecem infinitas possibilidades, o tipo de dados Boolean é simples. Ele representa sempre um dos dois valores: verdadeiro ou falso. Você vai encontrar dados booleanos ao criar programas JavaScript que respondam de forma inteligente para a entrada do usuário e ações. Por exemplo, se você quer ter certeza de que um visitante forneceu um endereço de e-mail antes de submeter um formulário, você pode adicionar uma lógica para sua página com uma simples pergunta: "Será que o usuário digitou um endereço de email válido" A resposta a esta pergunta é um valor booleano: ou o endereço de e-mail é válido (verdadeiro) ou não é (falso). Dependendo da resposta à pergunta, a página poderia responder de maneiras diferentes. Por exemplo, se o endereço de e-mail é válido (verdadeiro), em seguida, enviar o formulário, se ele não é válido (falso), em seguida, exibir uma mensagem de erro e impedir que o formulário seja enviado.

Na verdade, valores booleanos são tão importantes que o JavaScript inclui duas palavras-chave especiais para representar os valores:

Verdadeiro (*true*) e falso (*false*)

Você vai aprender como valores booleanos entram em jogo quando for adicionada lógica para seus programas.

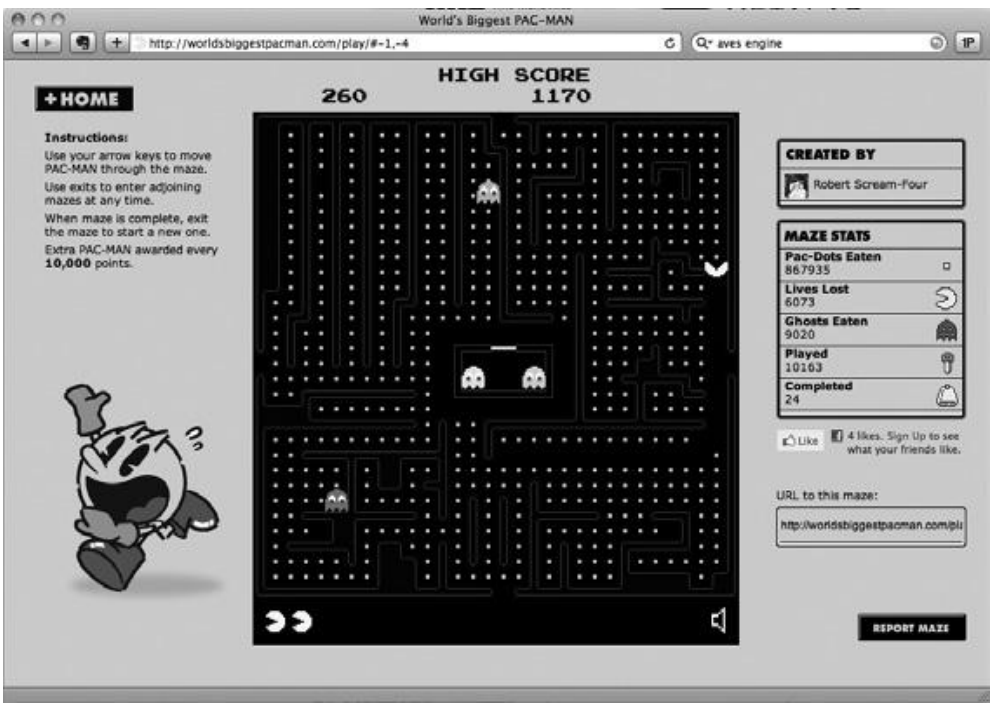
Variáveis

Você pode digitar um número, string ou valor booleano diretamente em seu programa JavaScript, mas esses tipos de dados só funcionam quando você já tem a informação de que precisa. Por exemplo, você pode fazer a string "Oi Fabio" aparecem em uma caixa de alerta como neste exemplo:

```
alert('Oi Fulano');
```

Mas essa afirmação só faz sentido se todos os que visitam da página se chamarem Fabio. Se você pretende apresentar uma mensagem personalizada para visitantes diferentes, o nome precisa ser diferente dependendo de quem está vendo a página: "Oi Fabio ','Oi Andréa','Oi Henrique' e assim por diante. Felizmente, todas as linguagens de programação fornecer algo conhecido como variável para lidar com apenas este tipo de situação.

Uma variável é uma maneira de armazenar informações para que mais tarde você possa usá-lo e manipulá-lo. Por exemplo, imagine um jogo de pinball baseado em JavaScript, onde o objetivo é obter a maior pontuação. Quando um jogador inicia o jogo, sua pontuação será zero, mas como ela bate o pinball em metas, a pontuação vai ficar maior. Neste caso, o resultado é uma variável, uma vez que começa em 0, mas muda conforme o jogo progride, em outras palavras, uma variável contém informações que podem variar. Veja a figura abaixo um exemplo de outro jogo que usa variáveis.



Pense em uma variável como um tipo de cesta: Você pode colocar um item em uma cesta, olhar para dentro da cesta, despejar o conteúdo de uma cesta, ou até mesmo substituir o que está dentro da cesta com outra coisa. No entanto, mesmo que você possa mudar o que está dentro do cesta, ele ainda continua a ser a mesma cesta.

Criando uma variável

Criar uma variável é um processo de duas etapas que envolve declarar a variável e nomeá-lo. Em JavaScript, para criar uma pontuação variável, você deverá digitar:

```
var pontos;
```

A primeira parte, **var**, é uma palavra-chave JavaScript que cria, ou, em programação declara a variável. A segunda parte da declaração, **ponto**, é o nome da variável.

O que nome que você colocar para a variável não importa, mas existem algumas regras que você deve seguir ao nomear variáveis:

- Os nomes de variáveis devem começar com uma letra, $, ou _. Em outras palavras, você não pode começar um nome de variável com um número ou pontuação: assim 1coisa e &coisa não vai dar certo, mas, $ponto, e _ponto estão corretos.
- Os nomes de variáveis podem conter apenas letras, números, $ e _. Você não pode usar espaços ou quaisquer outros caracteres especiais em qualquer lugar do nome da variável peixe&batatas e peixe e batatas fritas não são legais, mas peixe_e_batatas e pLAN9 são.
- Os nomes das variáveis são case-sensitive, ou seja, o interpretador JavaScript vê maiúsculas e letras minúsculas como distintos, uma variável chamada **PONTO** é diferente de uma variável chamada de **ponto**, que também é diferente de variáveis chamadas de **pONTO** e **Ponto**.
- Evite palavras-chave. Algumas palavras em JavaScript são específicos para a linguagem em si: **var**, por exemplo, é usado para criar uma variável, então você não pode nomear uma variável **var**. Além disso, algumas palavras, como ***alert***, ***document*** e ***window***, são considerados propriedades especiais do navegador. Você vai acabar com um erro de JavaScript, se você tentar usar essas palavras como nomes de variáveis.

Você pode encontrar uma lista de algumas palavras reservadas na tabela abaixo. Nem todas estas palavras reservadas vai causar problemas em todos os navegadores, mas o melhor é ficar longe desses nomes ao nomear variáveis.

Palavra-chave	Reservado para JavaScript	Resevado para o navegador
break	abstract	alert
case	boolean	blur
catch	byte	closed
continue	char	document
debugger	class	focus
default	const	frames
delete	double	history
do	enum	innerHeight
else	export	innerWidth
false	extends	length
finally	final	location
for	float	navigator
function	goto	open
if	implements	outerHeight
in	import	outerWidth
instanceof	int	parent
new	interface	screen
null	let	screenX
return	long	screenY

switch	native	statusbar
this	package	window
throw	private	
true	protected	
try	public	
typeof	short	
var	static	
void	super	
while	synchronized	
with	throws	
	transient	
	volatile	
	yield	

Além dessas regras, faça os nomes de suas variáveis claras e significativa. Variáveis de nomeadas de acordo com o tipo de dados que estará armazenando-os torna muito mais fácil olhar para o seu código de programação e entender imediatamente o que esta acontecendo. Por exemplo, o resultado é um grande nome para uma variável usada para controlar um leitor de resultado do jogo. O nome da variável **p** também funcionariam, mas a única letra "p" não dara nenhuma idéia sobre o que está armazenado na variável.

Dessa forma, fazer os nomes das variáveis de fácil leitura. Quando você usa mais de um palavra em um nome de variável, use um sublinhado entre as palavras ou capitalizar a primeira letra de cada palavra depois da primeira. Por exemplo, imagepath não é tão fácil de ler e entender como image_path ou ImagePath.

Usando variáveis

Uma vez que uma variável é criada, você pode armazenar qualquer tipo de dados que você gostaria nele. Para fazer assim, você usa o sinal =. Por exemplo, para armazenar o número 0 em uma pontuação de variável chamada ponto.

Você pode digitar o seguinte código:

```
var ponto;
ponto = 0;
```

A primeira linha de código acima cria a variável, a segunda linha armazena o número 0 na variável. O sinal de igualdade é chamado um operador de atribuição, porque ele é usado para atribuir um valor a uma variável. Você também pode criar uma variável e armazenar um valor na mesma com apenas uma declaração JavaScript como esta:

```
var pontuação = 0;
```

Você pode armazenar strings, números e valores booleanos em uma variável:

```
var nome = 'Fulano';
var sobrenome = 'de Tal;
var idade = 25;
var professor = true;
```

***Dica**: Para economizar digitação, você pode declarar múltiplas variáveis com uma única palavra-chave **var**, como esta:*

```
var x, y, z;
```

Você pode até declarar valores em múltiplas variáveis em um instrução JavaScript:

var professor = true, TemMedoDeAltura = false;

Uma vez que você tenha armazenado um valor em uma variável, você pode acessar esse valor simplesmente usando o nome da variável. Por exemplo, para abrir uma caixa de diálogo de alerta e exibir o valor armazenado na variável de pontuação, você usaria o seguinte:

```
alert(ponto);
```

Observe que você não usa aspas com uma variável, que é apenas para strings, de modo que o código alert ('ponto') vai exibir a palavra "ponto" e não o valor armazenado na variável pontuação. Agora você já sabe por que strings devem estar entre aspas: O intérprete JavaScript trata palavras sem aspas como objetos especiais de JavaScript (como o comando alert()) ou como nomes de variáveis.

Nota: Você só deve usar a palavra-chave **var** uma vez para cada variável quando você criar a variável. Depois disso, você está livre para atribuir novos valores para a variável sem usar **var**.

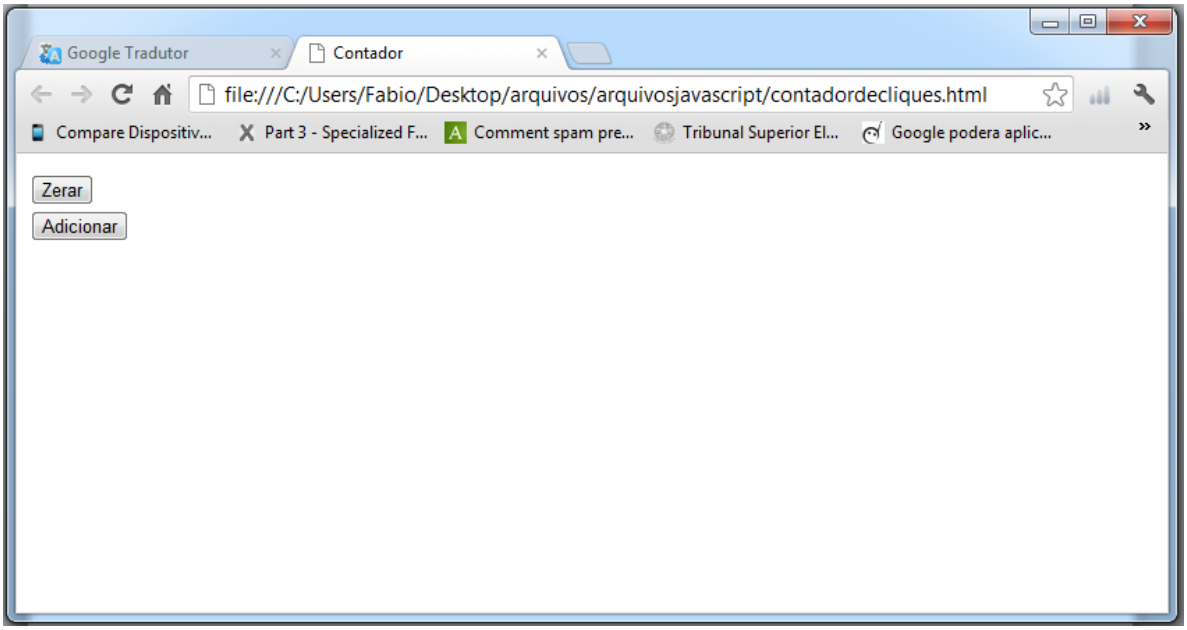
Vamos criar um exemplo com tudo que foi apresentado aqui.

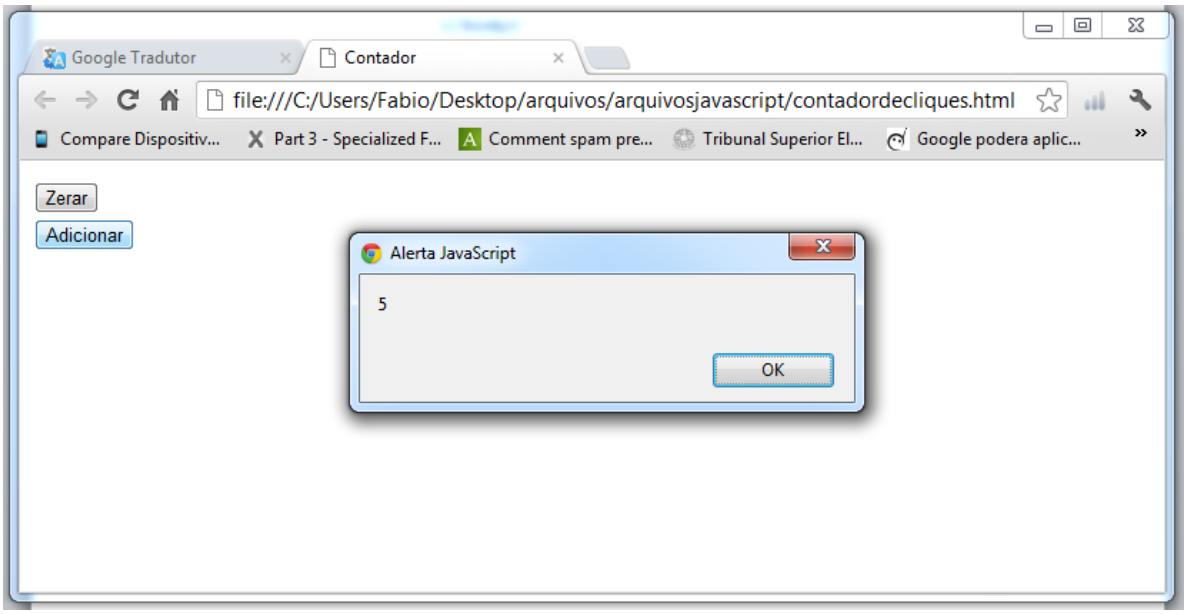
1. Abra seu editor de textos e insira o seguinte código e salve como **contadordecliques.html**:

```
<!doctype html>
<html>
<head>
   <meta charset="UTF-8">
   <title>Contador</title>
   </head>
   <body>
   <script>
   var ponto = 0;
   function Maisum(){
      ponto = ponto + 1;
alert(ponto)
   }
   </script>
   <pre>
   <input type="button" value="Zerar" onClick="ponto = 0">
   <input type="button" value="Adicionar"
onClick="Maisum()">
   </body>
```

```
</html>
```

2. Salve seu arquivo e teste no navegador.





Window.status e defaultstatus

Tanto status como defaultStatus são utilizadas para atribuir uma string na barra de status, com a diferença que a propriedade defaultStatus pode ser utilizada como um simples objeto apesar de ser ainda uma propriedade do objeto window, mas além disto a propriedade defaultStatus permite armazenar a mensagem padrão que será exibida, ou seja, aquela que voltará a ser exibida após modificações temporárias provocadas por mensagens colocadas na propriedade status.

A sintaxe básica das duas propriedades seguem o seguinte parâmetro:

```
window.status("mensagem");
window.defaultStatus = "Esta é a mensagem padrão";
defaultStatus = "Esta é a mensagem padrão";
```

Veja o funcionamento disto acionando os botões abaixo:

Método open

Este método tem como objetivo abrir uma nova janela do browser. Com este método o usuário poderá abrir uma nova janela em branco ou uma janela que contém um documento específico. Sua sintaxe tem a seguinte formação:

```
NomeJanela = window.open(URL, alvo, opções);
```

Onde NomeJanela é uma variável que será uma referência a nova janela.

Onde URL é o endereço da janela a ser aberta.

Onde alvo é o nome da janela para ser usado no atributo target dos tag's <FORM> ou <A>.

Onde opções é o parâmetro que controla o comportamento da janela.

Método close

O método close do objeto window tem a finalidade de fechar uma janela.

Normalmente utiliza-se este método atribuído à um botão de ação criado com os formulários. Sua sintaxe básica tem a seguinte formação:

```
window.close();
```

No exemplo abaixo temos uma página com um botão chamado Fechar, onde quando o usuário clicar sobre o mesmo é acionado este evento.

```
<input type=”button” value=”Fechar” onClick=”window.close()”>
```

Neste caso, foi utilizado o evento onClick que executa a instrução

window.close() assim que o usuário clica sobre o botão.

Método settimeout

Através do método setTimeout o usuário poderá criar um contador de tempo que executa alguma ação após um determinado intervalo de tempo.

Sua sintaxe segue o seguinte padrão:

```
variável = setTimeOut(“função”, intervalo);
```

Onde é função é alguma instrução que o usuário quer que execute após o intervalo especificado em milissegundos (milésimos de segundos). Na maioria das vezes, função é uma chamada de uma função criada anteriormente.

Onde é intervalo é o tempo até que a função seja executada.

Um dos exemplos mais comuns de uso do método setTimeout é o da criação de contadores em uma página e textos rolantes na barra de status.

Veja pelo exemplo do script a seguir, o uso de texto piscante na barra de status:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>setTimeout</title>
```

```
<script>
var texto = "Curso JavaScript";
var velocidade = 150;
var controle = 1;
function flash(){
      if (controle == 1) {
            window.status = texto;
            controle=0;
      } else {
            window.status = "";
            controle=1;
}
setTimeout("flash();", velocidade);
}
flash();
</script>
</head>
<body>
</body>
</html>
```

Método cleartimeout

Através do método clearTimeout o usuário poderá cancelar um marcador de tempo que foi criado pelo método setTimeout. Sua sintaxe segue o seguinte padrão:

```
clearTimeout(tempo);
```

Veja Exemplo:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>setTimeout</title>
<script>
var texto = "Curso JavaScript";
var velocidade = 150;
var controle = 1;
```

```
var tempo;
function startFlash(){
      if (controle == 1) {
            window.status = texto;
            controle=0;
      } else {
            window.status = "";
            controle=1;
      }
      tempo=setTimeout("startFlash();", velocidade);
}

function stopFlash(){
  clearTimeout(tempo);
  controle=0;
}
</script>
</head>
<body>
<input name="iniciar" type="button" value="iniciar pisca-pisca"
onClick="startFlash();">
<input name="parar" type="button" value="parar pisca-pisca"
onClick="stopFlash();">
</body>
</html>
```

Onde é tempo é o manipulador de identificação do timer criado pelo método setTimeout.

Questionário

1. O que usuário pode controlar com o evento onBlur?

_____
_____
_____
_____

2. O que o usuário pode determinar com o evento onUnLoad?

_____
_____
_____
_____

3. O que é o objeto document?

_____
_____
_____
_____

4. Em qual versão do Internet Explorer funciona o método clear?

_____
_____
_____
_____

5. Qual a finalidade de utilizarmos variáveis no JavaScript?

_____
_____
_____
_____

Aula 3 - Declarações e Erros

Tópicos da Aula

- Literais
- Declarações
- Desenvolvimento de scripts
- Notificação de erros

Trabalhando com janelas

Qualquer usuário que costuma navegar na Internet, sabe que é possível manipular a janela aberta de um site através de comandos do próprio browser como por exemplo o comando Tela Cheia encontrado no menu Exibir do navegador, entre outras opções.

Mas através da linguagem JavaScript é possível realizar as mesmas operações através de sua programação.

Se o usuário desejar abrir uma nova janela à partir de uma janela já aberta, basta utilizar o método open em sua estrutura.

Veja sua sintaxe:

```
window.open("URL");
janela = window.open("URL");
```

Onde:

janela: é referente ao nome dado para a janela a ser aberta para fins das instruções de programação.

window.open: OPEN é o método do objeto window para abrir uma nova janela.

URL: é o endereço da página que será aberta na nova janela. Caso o usuário omitir este endereço, será aberta uma nova janela vazia.

A omissão de uma URL, será aberta uma nova janela em branco.

Veja no exemplo abaixo, a criação de uma nova janela chamada GOOGLE onde será aberto o site do google.

```
GOOGLE = window.open("http://www.google.com.br");
```

É possível através de outros argumentos, definir o comportamento de como a nova janela deverá ser apresentada. Veja os argumentos existentes para o método open:

Propriedades	Descrição
Toolbar	Permite a exibição da barra de ferramentas do navegador.

Location	Permite a exibição da barra de endereço do navegador.
Directories	Permite a exibição da barra de links do navegador.
Status	Permite a exibição da barra de status do navegador.
Menubar	Permite a exibição da barra de menu do navegador.
Scrollbars	Permite a exibição das barras de rolagem do navegador.
Resizable	Permite ao usuário redimensionar a nova janela do navegador.
Width	Especifica a largura da nova janela do navegador em pixels.
Height	Especifica a altura da nova janela do navegador em pixels.

Quaisquer uns destes argumentos possuem valores booleanos (yes/no,1/0).

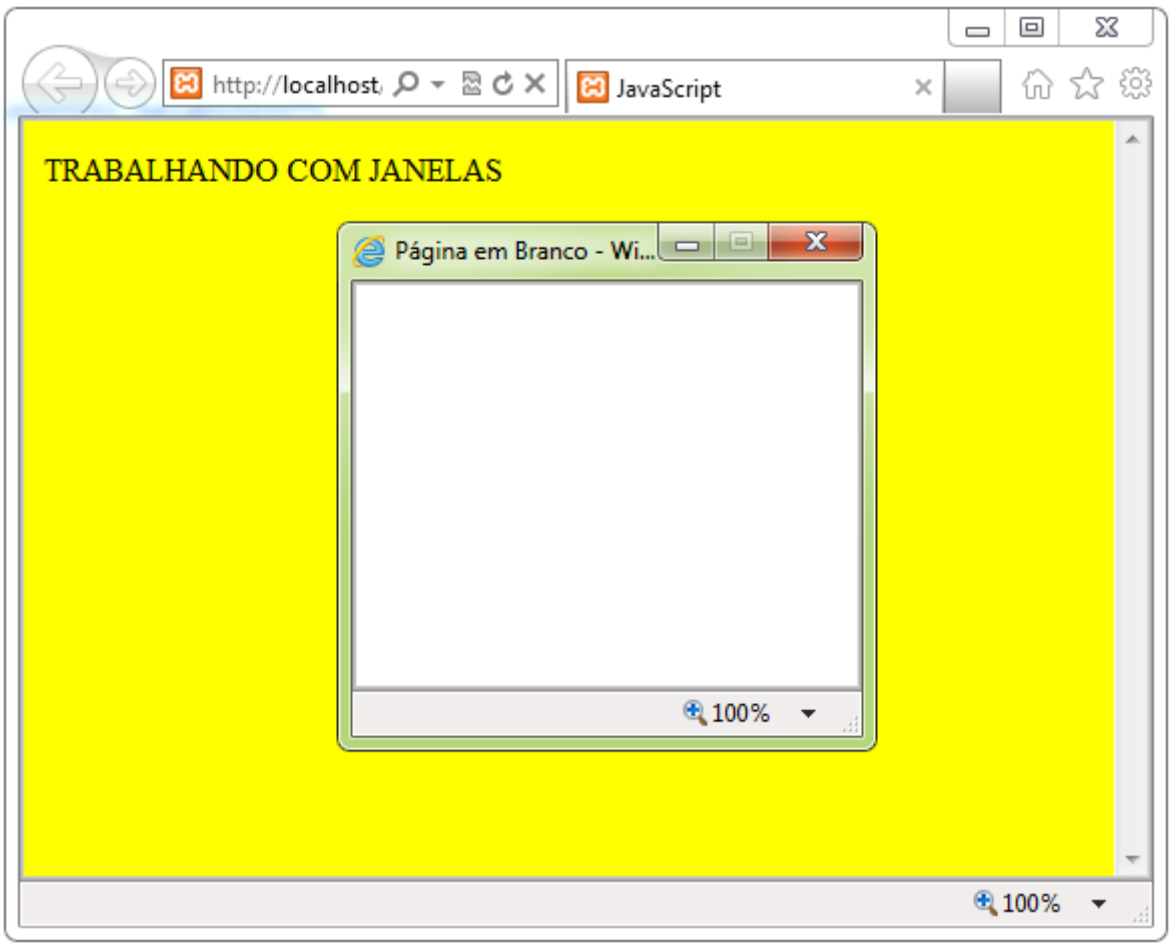
Se utilizar mais de um argumento, os mesmos deverão estar separados por vírgula. Aquelas opções não definidas irão assumir valores falsos.

No exemplo apresentado a seguir, é mostrada a abertura de uma segunda página apenas com a barra de status e com tamanho de 250 x 200 pixels:

```
<html><head>
<title>JavaScript</title>
<script>
janela2 = window.open("", "", "status=yes, width=250, height=200");
</script>
</head>
```

```
<body bgColor=”#FFFF00”>
Trabalhando com Janelas
</body>
</html>
```

Veja o efeito deste código apresentado na figura a seguir:



Se o usuário desejar controlar o código HTML da janela gerada, basta determinar no código JavaScript da janela principal os tag’s específicos para a segunda janela.

Veja pelo exemplo a seguir:

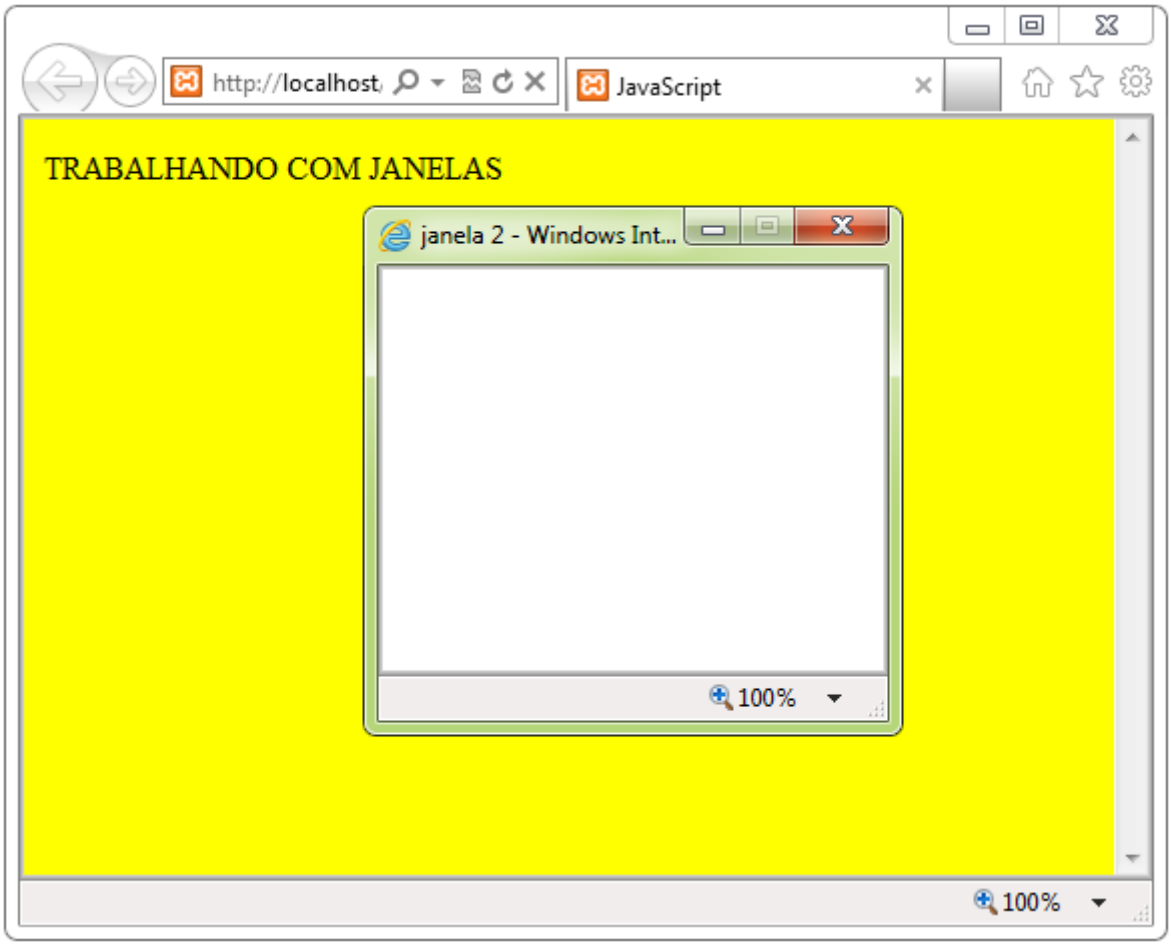
```
<html><head>
<title>JavaScript</title>
<script>
janela2 = window.open(“”, ””, ”status=yes, width=250, height=200”);
```

```
janela2.document.write("<head><title>Janela 2</title></head>");
</script>
</head>
<body bgColor="#FFFF00">
Trabalhando com Janelas
</body>
</html>
```

Neste código foi usado o objeto document atribuído a variável que representa a janela secundária "JANELA2" para especificar os tag's de cabeçalho e título para esta nova janela.

```
janela2.document.write("<head><title> Janela 2 </title>
</head>")
```

Veja pelo exemplo da próxima figura que a nova janela será apresentada com o título JANELA 2 em sua barra de título:

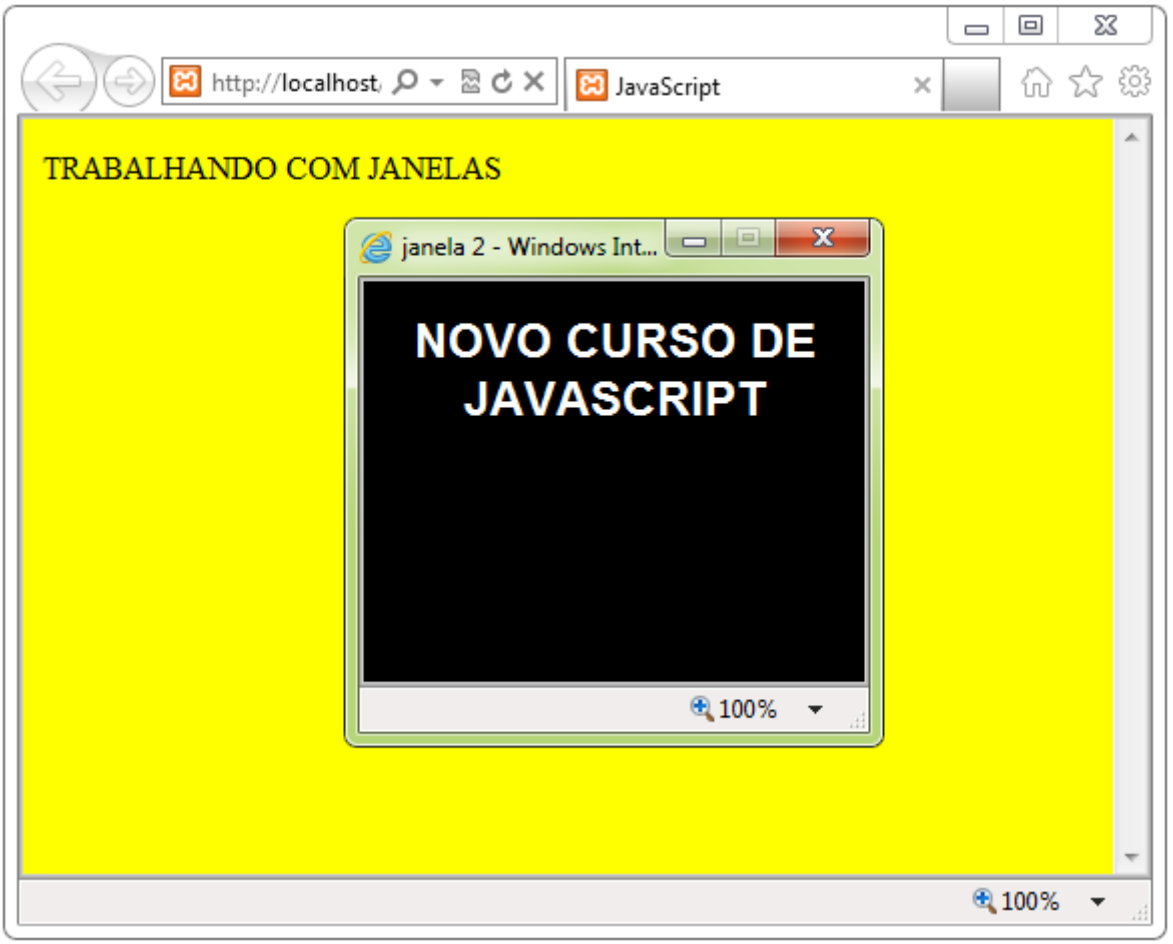


Vamos agora incrementar nosso código, controlando o corpo da nova janela com cores e até com um texto presente. Veja no código a seguir o uso do objeto document e seu método write que irá permitir o controle sobre a segunda janela aberta a partir da primeira:

Vamos adicionar ao nosso script existente a seguinte linha além da já existente:

```
janela2.document.write("<body bgColor=black>");
janela2.document.write("<center><h2><font face=arial color=white>","Novo
curso de <br> JavaScript</h2></center></font>");
```

Teremos o seguinte efeito conforme mostrado na figura a seguir:



Aproveitando a criação desta nova janela, iremos criar um botão de formulário onde atribuiremos uma ação de evento que fará o fechamento automático desta janela.

Para isto, basta utilizar o método close para o objeto window.

Veja pelo exemplo do código a seguir:

```
<html><head>
<title>JavaScript</title>
<script>
janela2 = window.open("","","status=yes, width=250, height=200);
janela2.document.write("<head><title> Janela 2</title> </head>";
janela2.document.write("<body bgColor=black>");
janela2.document.write("<center><h2><font face=arial color=white>","Novo
curso de <br> JavaScript</h2></center></font>");
```

```
janela2.document.write("input type='button' name='fecha'","value='Fechar
Janela' onClick='window.close()'>");
</body>
</html>
```

O método onClick foi utilizado para acionar o objeto window assim que o usuário clicar sobre este botão. Com isto, ele executará o método close que tem a função de fechar a janela onde ele se encontra. Veja pelo exemplo da figura a seguir:

Criaremos agora na janela principal um novo botão com a finalidade de abrir uma nova janela, para isto deve-se definir o botão no corpo da janela principal.

Para que se abra a mesma janela que foi fechada, é necessário que se crie uma função para aproveitar o código já utilizado.

```
<html><head>
<title>JavaScript</title>
<script>
function abrirJanela(){
janela2.document.write("<head><title> Janela 2</title> </head>";
janela2.document.write("<body bgColor=black>");
janela2.document.write("<center><h2><font face=arial color=white>","Novo
curso de <br> JavaScript</h2></center></font>");
janela2.document.write("input type='button' name='fecha'","value='Fechar
Janela' onClick="window.close()'>");
}
<script>
</head>
<body bgColor="#FFFF00">
Trabalhando com Janelas<br>
<input type="button" name="botao1" value="Abrir Janela"
onClick="abrirJanela()" />
</body>
</html>
```

Abrindo páginas em fullscreen (tela cheia)

Através do argumento fullscreen o usuário poderá abrir a janela do browser em tela cheia, fazendo-o ocupar toda à área do monitor. O código a seguir permite ao usuário abrir sua página em tela cheia

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>fullscreen</title>
<script>
var tela = window.open('','Tela','toolbar=no, location=no, directories=no,
status=no, menubar=no, scrollbars=yes, resizable=no, fullscreen=yes');
tela.location.href = 'http://www.google.com'; //IE
</script>
</head>
<body></body>
</html>
```

Caracteres especiais

Estes caracteres são especificados dentro de uma string. Veja na tabela abaixo estes caracteres e sua descrição:

Caractere	Descrição
\n	Insere uma quebra de linha.
\t	Insere uma tabulação.
\r	Insere um retorno.
\f	Insere um caractere de barra.
\t	Tabulação.
\’	Apóstrofo.
\”	Aspas.
\\	Barra Invertida.

Caractere representado pela codificação Latin-1.

Exemplo \251 representa o caractere de copyright ©.

OBS: As letras dos operadores devem apresentar-se em letras minúsculas.

Trabalhando com tipos de dados e variáveis

Armazenar um determinada informação como um número ou string em uma variável é normalmente apenas um primeiro passo em um programa. A maioria dos programas também podem manipular dados para obter novos resultados. Por exemplo, adicionar um número a uma pontuação para o aumentar, multiplicar o número de itens de pedidos pelo custo do item para obter um total geral, ou personalizar uma mensagem genérica.

O JavaScript fornece vários operadores para modificar os dados. Um operador é simplesmente um símbolo ou palavra que pode mudar um ou mais valores em outra coisa. Por exemplo, você usar o símbolo operador + para adicionar números. Existem diferentes tipos de operadores para tipos de dados diferentes.

Matemática Básica

O JavaScript suporta operações matemáticas básicas, como divisão, adição, subtração, e assim por diante. A tabela abaixo mostra os operadores matemáticos mais básicos e como usá-los.

Operador	O que ele faz	Como usar
+	Adição	5 + 25
-	Subtração	25 - 5
*	Multiplicação	5 * 10
/	Divisão	15/5
%	Resto da divisão	10 % 5

Você deve estar acostumado a utilizar x para multiplicações (4 x 5, por exemplo), mas no JavaScript, você usa o símbolo * para multiplicar dois números.

Você também pode usar variáveis em operações matemáticas. Uma vez que a variável é apenas um recipiente para outro valor como um número ou string, usando uma variável é o mesmo como usar o conteúdo dessa variável.

```
var preco = 10;
var items_comprados = 15;
var total = preco * items_comprados;
```

As duas primeiras linhas de código criam duas variáveis (**preco** e **items_comprado**) e é atribuído um valor a cada uma delas. A terceira linha de código cria uma outra variável (**total**) e armazena os resultados da multiplicação. Neste caso, o valor (150) é armazenado no variável total.

Este código de exemplo também demonstra a utilidade de variáveis. Suponhamos que você escreva um código que faça parte de um sistema de compras para um site de e-commerce. Ao longo do programa, você precisa usar o preço de um determinado produto para fazer várias cálculos.

Sem a criação da variável você teria que ir em cada lugar onde foi definido o valor do produto e alterar manualmente, imagina uma loja como mais de cinco mil produtos, daria um trabalho muito grande e cansativo.

A Ordem de Operações

Se você executar várias operações matemáticas de uma vez, por exemplo, você totalizar vários números em seguida, multiplicá-los todos em 10, você precisa ter em mente a ordem em que o interpretador de JavaScript executa seus cálculos. Alguns operadores tem precedência sobre outros operadores, por isso eles são calculados primeiro. Este fato pode causar alguns resultados indesejados, se você não tiver cuidado.

Veja este exemplo:

```
4 + 5 * 10
```

Você pode pensar que isso simplesmente é calculado a partir da esquerda para a direita: 4 + 5 é de 9 e 9 * 10 é 90. Não é, a multiplicação vai ser feita por primeiro, então essa equação ficaria assim 5 * 10 é 50, mais 4 é 54. Multiplicação (o símbolo *) e divisão (o símbolo /) tem precedência sobre a adição (+) e subtração (-).

Para se certificar de que a matemática funciona do jeito que você quiser, use parênteses para agrupar operações. Por exemplo, você poderia reescrever a equação acima da seguinte forma:

```
(4 + 5) * 10
```

Qualquer matemática que é realizado dentro de parênteses acontece por primeiro, portanto, neste caso, o 4 é adicionado ao 5 e o resultado, 9, é então multiplicado por 10. Se você quiser que a multiplicação ocorrer primeiro, seria mais clara a escrever um código como este:

```
4 + (5 * 10)
```

Concatenando Strings

A combinação de duas ou mais strings para fazer uma única string é uma tarefa comum em programação. Por exemplo, se uma página web tem um formulário que recebe o primeiro nome de uma pessoa em um outro campo que recebe o sobrenome, é preciso combinar os dois campos para obter o seu nome completo. Além do mais, se você quiser exibir uma mensagem ao usuário para que ele saiba que seu formulário foi enviado, você precisa combinar a mensagem com o nome da pessoa: "Fulano de Tal, Obrigado por sua compra."

Strings combinadas são chamadas de concatenação, e realizá-lo com o operador +. Sim, isso é o operador + mesma que você usa para adicionar valores numéricos, mas com cordas que se comporta um pouco diferente. Aqui está um exemplo:

```
var nome = 'Fulano';
var sobrenome = 'de Tal';
var nomecompleto = nome  + sobrenome;
```

Na última linha do código acima, o conteúdo da variável nome é combinadas (ou concatenado) com o conteúdo da variável sobrenome os dois são literalmente unidos e o resultado é colocado na variável nomecompleto. Neste exemplo, a string resultante é "FabioSantos" não há um espaço entre os dois nomes, este concatenar apenas funde as strings. Em muitos casos (como este), você precisa para adicionar um espaço vazio entre as strings que você pretende combinar:

```
var nome = 'Fulano';
var sobrenome = 'de Tal';
var nomecompleto = nome + ' ' + sobrenome;
```

As aspas simples na última linha deste código é uma citação simples. Este código é simplesmente uma string que contém um espaço vazio. Quando colocado entre as duas variáveis neste exemplo, ele cria a string "Fulano de Tal ".

Este último exemplo também demonstra que você pode combinar mais de duas cadeias de uma vez, neste caso, três strings.

***Nota**: Lembre-se que uma variável é apenas um contêiner que pode conter qualquer tipo de dados, como uma string ou número. Então, quando você combina duas variáveis com string (nome + sobrenome), é o mesmo que juntar dois strings assim: 'Fulano' + 'de Tal'.*

Combinando Numeros e Strings

A maioria dos operadores matemáticos só faz sentido para os números. Por exemplo, ele não faz qualquer sentido se você tentar multiplicar 2 vezes "ovos". Se você tentar este exemplo, você vai receber um código especial de JavaScript com valor NaN, que significa "não um número."

No entanto, há momentos em que você pode querer combinar uma string com um número. Por exemplo, digamos que você deseja apresentar uma mensagem em uma página web que especifica quantas vezes um visitante foi para o seu site. O número de vezes que ela é visitada é um número, mas a mensagem é uma string. Neste caso, você pode usar o operador + para juntar as duas coisas: converter o número para uma string e concatenar com a outra string. aqui está um exemplo:

```
var num_visitantes = 101;
var mensagem = Você é o visitante número ' +
num_visitantes;
```

Neste caso, a mensagem contento a string "Você é o visitante 101." O intérprete JavaScript reconhece que há uma sequência envolvida, para que ele perceba que não vai estar a fazer qualquer matemática (sem adição). Em vez disso, ele trata o + como o operador de concatenação, e, ao mesmo tempo percebe que o número deve ser convertida para uma cadeia também.

Este exemplo pode parecer uma boa forma de imprimir palavras e números na mesma mensagem. Neste caso, é óbvio que o número faz parte de uma sequência de letras que torna-se uma frase completa, e sempre que você usar o operador + com uma string e um número, o interpretador de JavaScript converte o número para uma string.

Essa característica, conhecida como a conversão automática de tipo, pode causar problemas, no entanto. Por exemplo, se um visitante responde a uma pergunta em um

formulário ("Quantos pares de sapatos gostaria? "), digitando um número (2, por exemplo), que a entrada é tratada como um string '2 '. Assim, você pode ter em uma situação como esta:

```
var num_sapatos = '2 ';
var num_meias = 4;
var total = num_sapatos + num_meias;
```

O valor esperado na variável total seria 6 (2 sapatos + 4 pares de meias). Em vez disso, porque o valor em num_sapatos é uma string, o interpretador de JavaScript converte o valor da variável num_meias para uma string, assim, e você acaba com a sequência "24" na variável total. Existem algumas maneiras de evitar esse erro.

Primeiro, você pode adicionar + ao início da string que contém um número como este:

```
var num_sapatos = '2';
var num_meias = 4;
var total = +num_sapatos + num_meias;
```

Adicionando um sinal de + antes de uma variável (certifique-se que não há espaço entre os dois) diz ao interpretador JavaScript para tentar converter a string para um número de valor se a cadeia contém apenas números, como "2", você vai acabar com a seqüência convertida para um número.

Neste exemplo, você acaba com 6 (2 + 4). Outra técnica é utilizar o comando Number() como abaixo:

```
var num_sapatos = '2 ';
var num_meias = 4;
var total = Number(num_sapatos) + num_meias;
```

Number () converte uma string para um número, se possível. (Se a string é apenas letras e não números, você obter o valor NaN para indicar que você não pode transformar letras em um número.)

No geral, você vai encontrar na maioria das vezes os números como strings quando chegar a entrada de um visitante da página, por exemplo, ao recuperar um valor que um visitante entrou em um campo de formulário. Então, se você precisa fazer qualquer adição usando a entrada de

dados coletados de uma forma ou de outra fonte de entrada de visitantes, certifique-se de executá-lo através do Number().

***Nota**: Este problema só ocorre quando a adição de um número com uma string que contém um número.*

Alterando os valores das variáveis

As variáveis são úteis porque podem conter valores que mudam com o programa, e podem ser alteradas quando você faz pontos e um jogo, por exemplo. Assim como você pode alterar o valor da variável? Se você quiser apenas substituir o que está contido dentro de uma variável, é só atribuir um novo valor para a variável. Por exemplo:

```
var ponto = 0;
ponto = 100;
```

No entanto, você frequentemente deseja manter o valor que está na variável e apenas adicionar alguma coisa com ele ou alterá-lo de alguma forma. Por exemplo, com uma pontuação do jogo, você nunca vai dar uma nova pontuação, você sempre vai adicionar ou subtrair a pontuação atual. Para adicionar o valor de uma variável, você pode usar o nome da variável como parte da operação como abaixo:

```
var ponto = 0;
ponto = ponto + 100;
```

Essa última linha de código pode parecer confuso no início, mas ele usa uma técnica muito comum. Eis como funciona: Toda a ação acontece com o sinal de = , isto é, a pontuação + 100 da peça. Traduzido, isto significa "tirar o que está armazenado atualmente na pontuação (0) e, em seguida, adicionar 100 a ela. "O resultado dessa operação é então armazenado de volta para o variável de ponto. O resultado final dessas duas linhas de código é que a pontuação variável agora tem o valor de 100.

A mesma lógica se aplica a outras operações matemáticas como subtração, divisão, ou de multiplicação:

```
ponto = ponto - 10;
ponto = ponto * 10;
ponto = ponto / 10;
```

Na verdade, realizar cálculos sobre o valor em uma variável e depois armazenar o resultado na variável é tão comum que há atalhos para fazê-lo, como retratado na tabela abaixo.

Operador	O que ele faz	Como usar	O mesmo como
+ =	Adiciona um valor ao valor atual	ponto += 10	ponto = ponto + 10
- =	Subtrai um valor do valor atual	ponto -= 10	ponto = ponto - 10
* =	Multiplica um valor do valor atual	ponto *= 10	ponto = ponto * 10
/ =	Divide um valor do valor atual	ponto /= 10	ponto = ponto / 10
++	Adiciona 1 ao valor atual.	ponto++	ponto = ponto + 1
- -	Subtrai 1 ao valor atual	ponto--	ponto = ponto - 1

As mesmas regras se aplicam ao concatenar uma string em uma variável. Por exemplo, digamos que você tem uma variável com uma string e nela você quer adicionar outro par de strings para essa variável:

```
var name = 'Fulano';
var mensagem = "Olá";
mensagem = mensagem + ' ' + nome;
```

Tal como acontece com os números, há um operador atalho para concatenar uma string a uma variável. O operador + = adiciona o valor da sequência à direita do sinal = para o final de cada string variável. Assim, a última linha do código acima poderia ser reescrito assim:

```
mensagem + = ' ' + nome;
```

Você verá o operador + = com frequência quando se trabalha com strings, e ao longo este livro.

Operadores lógicos

Para estes operadores, são exigidos valores booleanos, como operandos, e será retornado um valor lógico.

Operador	Descrição
&& ou AND	E
|| ou OR	OU
! ou NOT	NÃO

O operador “&&” retorna TRUE somente se todas as expressões forem verdadeiras.

O operador “||” retorna TRUE se uma das expressões forem verdadeiras. Se as duas forem falsas, será retornado FALSE.

O operador “!” nega uma expressão. Se for verdadeira, será retornado FALSE. Se for falsa, será retornado o valor TRUE.

Declarações

Vejamos agora uma relação das declarações existentes na linguagem JavaScript que são utilizadas na criação da estrutura lógica de um programa.

Normalmente estas declarações são atribuídas às tomadas de decisões, laços repetitivos e funções.

Operador new

Este operador irá permitir que o usuário crie uma nova instância de um objeto definido. Veja sua sintaxe:

```
NomeObjeto=new Tipo(parâmetros)
```

Palavra-chave this

Esta palavra-chave é utilizado para fazer referência ao objeto corrente. Veja sua sintaxe:

```
this.propriedade
```

Break

Esta instrução desvia o JavaScript de uma estrutura controlada e continua sua execução na primeira linha após o bloco da instrução onde foi encontrado.

Esta instrução pode ser utilizada em estruturas baseadas nas seguintes intruções:

```
for
for...in
while
```

Utilização de comentários

Assim como qualquer outra linguagem de programação, a linguagem JavaScript faz o uso de comentários que irão permitir ao programador inserir anotações referentes ao seu desenvolvimento ou explicar determinada operação de seu script.

Estes comentários na execução do script, são ignorados pelo interpretador (browser). Veja a sintaxe do uso de comentários na linguagem JavaScript:

```
// Comentário de uma linha de texto.
/* Comentário de várias linhas de texto,
continuação do comentário de várias linhas */
```

Conforme visto no exemplo anterior, quando o comentário for um pequeno texto que irá ocupar uma linha o usuário fará o uso da instrução "//" caso o mesmo irá compor mais linhas de texto no início do comentário utiliza-se a instrução "/*", e após a última linha de texto encerra-se com a instrução "*/".

Além destes comentários é recomendável que utilize antes de iniciar um script o Tag de comentário da Linguagem HTML, que irá permitir que navegadores já ultrapassados no sentido de não reconhecer as instruções JavaScript, possam ignorar estas instruções evitando erros futuros.

A sintaxe de utilização do Tag de comentário em um script é formada da seguinte forma:

```
<!--Inicio do JavaScript
Instruções
//Término do JavaScript -->
```

Observe que no final do script, foi definido um comentário de uma linha de texto no JavaScript, encerrando-se com o Tag de Fechamento da Linguagem HTML.

O comentário do JavaScript somente foi necessário em razão de haver um texto de comentário, caso contrário, bastaria o Tag de Comentário do HTML.

Arquivos externos de JavaScript

Usando a tag <script> como discutido na seção anterior permite que você adicione o JavaScript para uma única página. Mas muitas vezes você vai criar scripts que você quer compartilhar com todas as páginas do seu site. Por exemplo, você pode usar um programa em JavaScript para adicionar animações, menus de navegação drop-down para uma página web. Você vai querer que tenha uma barra de navegação em cada página de seu site, mas copiando e colando o JavaScript com o mesmo código em cada página do seu site é uma péssima idéia por várias razões.

Primeiro, é um grande trabalho de copiar e colar o mesmo código várias vezes, especialmente se você tem um site com centenas de páginas.

Segundo, se você decidir que quer mudar ou melhorar o código JavaScript, você precisará localizar todas as páginas que foi usado o seu JavaScript e atualizar o código um a um.

Finalmente, uma vez que todo o código para o programa de JavaScript estaria localizada em cada página da web, cada página será muito maior e mais lento para download.

Uma abordagem melhor é usar um arquivo JavaScript externo. Se você já usou arquivos de CSS externo para suas páginas web, esta técnica deve lhe parecer familiar. Um arquivo de JavaScript externo é simplesmente um arquivo de texto que termina com a extensão .js, por exemplo. O arquivo inclui apenas o código JavaScript e está ligada a uma página web usando a tag <script>. Por exemplo, para adicionar um arquivo JavaScript nomeado arquivo.js para sua página home, você pode escrever o seguinte:

```
<!doctype html>
<html>
   <head>
   <meta charset="UTF-8">
   <title>Minha Página</title>
   <script src="arquivo.js"></script>
</head>
```

O atributo src da tag <script> funciona exatamente como o atributo src de uma tag img,

ou o atributo href de uma tag <a>. Em outras palavras, ele aponta para um arquivo ou aponta para seu site ou para um outro site.

***Dica**: Ao adicionar o atributo src e vincular a um arquivo JavaScript externo, não se deve adicionar nenhum código JavaScript entre a abertura e fechamento de tags <script>.*

Se você deseja vincular a um arquivo JavaScript externo e adicionar código personalizado JavaScript para uma página, use um segundo conjunto de tags <script>. Por exemplo:

```
<script src="arquivo.js"></script>
<script >
alert('Bem vindo!');
</script>
```

Você pode (e muitas vezes vai) anexar vários arquivos JavaScript externos para uma única página. Por exemplo, você pode ter criado um arquivo JavaScript externo que controla uma barra de navegação drop-down, e outro que permite que você adicione um slideshow bacana para um página de fotos. Em sua página galeria de fotos, você gostaria de ter os dois programas de JavaScript, assim que você anexar os dois arquivos.

Além disso, você pode anexar arquivos JavaScript externos e adicionar um programa JavaScript para mesma página como abaixo:

```
<!doctype html>
<html>
<head>
   <meta charset="UTF-8">
   <title>Minha página</title>
    <script src="navegacao.js"></script>
    <script src="galeria.js"></script>
    <script>
      alert('Bem vindo!');
    </script>
</head>
```

Basta lembrar que você deve usar um conjunto de abrir e fechar tags <script> para cada arquivo JavaScript externo. Você vai criar um arquivo JavaScript externo em um próximo exemplo.

Você pode manter arquivos JavaScript externos em qualquer lugar dentro da pasta raiz do seu site (ou qualquer subpasta dentro da raiz). Muitos desenvolvedores web criam um diretório especial para arquivos JavaScript externos na pasta raiz do site: nomes comuns são js (significado JavaScript) ou libs (ou seja, bibliotecas).

***Nota**: Às vezes, a ordem em que você anexa arquivos externos JavaScript podem ocasionar falhas de execução, como você verá mais adiante neste livro.*

Desenvolvimento de scripts

As instruções da linguagem JavaScript podem ser escritas em qualquer editor ASCII, como por exemplo, o Bloco de Notas do Windows e até mesmo o Edit do MS-DOS, sendo que seu arquivo deverá ser salvo com a extensão HTML ou .JS. Para visualizar a execução deste script, basta acessá-lo através do browser.

Quando se desenvolve scripts em uma página HTML, é necessário que o usuário os delimite através do Tag <SCRIPT> ou utilize-os como manipuladores de eventos através de alguns Tag's HTML. Outra maneira é criar um arquivo externo para ser chamado à partir de uma página HTML. Este arquivo separado deverá possuir a extensão .JS.

Desenvolvendo scripts com o tag <script>

Com o Tag <SCRIPT> é possível ao usuário incorporar seus scripts dentro de uma página HTML. Veja a sintaxe de utilização deste Tag:

```
<script>
 //instruções do JavaScript...
</script>
```

Em alguns casos é possível observar o tag SCRIPT com o seguinte atributo:

```
<script language="javascript">
//instruções do JavaScript...
</script>
```

Desenvolvendo scripts através de um arquivo externo

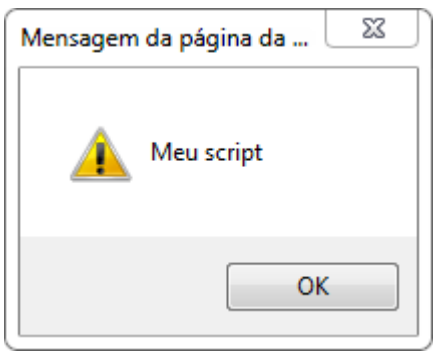
As instruções da linguagem JavaScript podem ser executadas de um arquivo externo. Com isto, o usuário não precisará repetir instruções várias vezes, isto, facilita a manutenção do código desenvolvido e a reutilização do mesmo.

Para isto, o usuário deverá criar o código em qualquer editor ASCII da mesma forma que se cria uma página HTML, e ao salvá-lo, o usuário deverá atribuir ao seu nome a extensão .JS.

Neste arquivo o usuário não precisará utilizar o Tag HTML para delimitar suas instruções.

Conforme dito anteriormente, a linguagem JavaScript é interpretada pelo browser e que seu código é embutido dentro do código HTML entre os tag's <script> e </script> ou através de um arquivo externo que possua a extensão .JS.

Observe o uso de algumas ações que o JavaScript pode desenvolver através da figura a seguir:



Caixa de diálogo criada por uma instrução da Linguagem JavaScript.

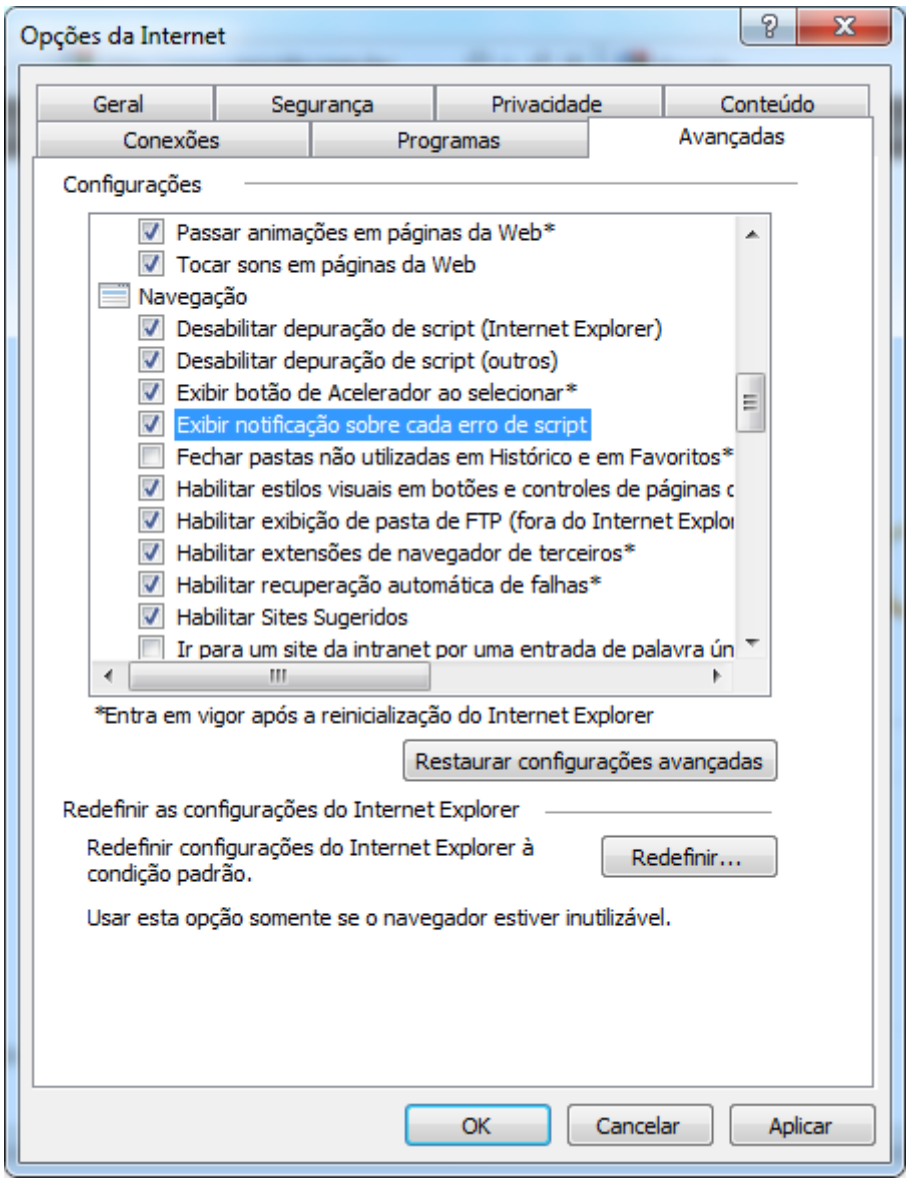
Em outro capitulo aprenderemos como gerar o código desta janela.

Notificação de erros

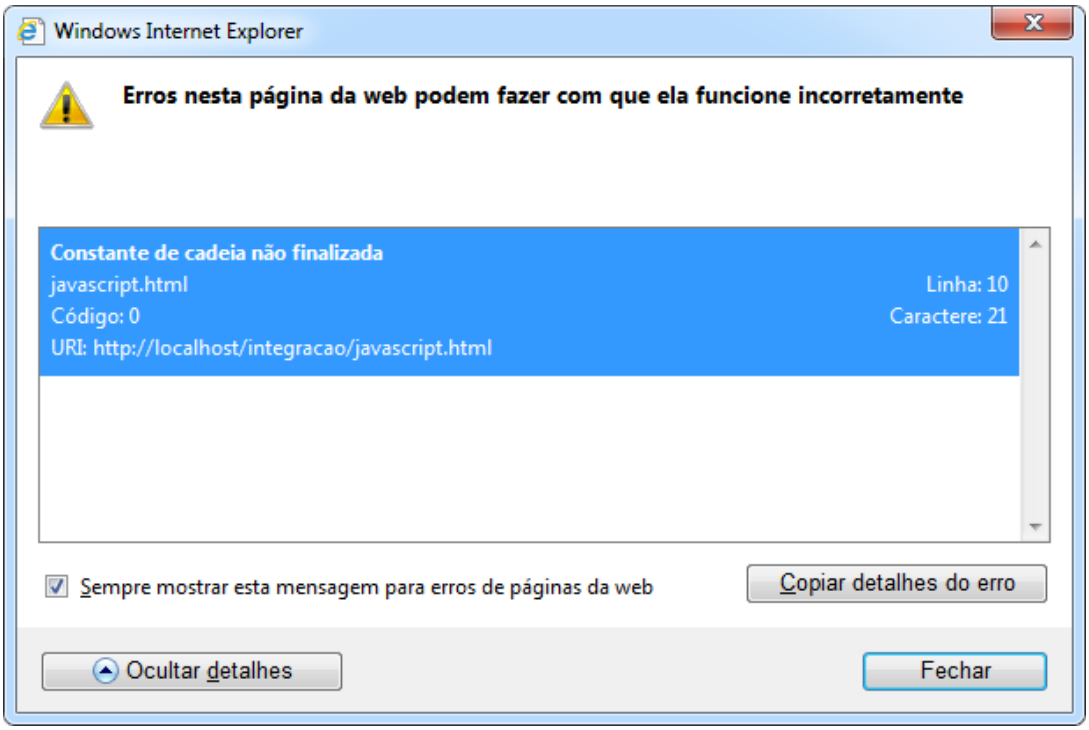
Além dos comentários, que irão evitar que os navegadores mais antigos exibam algum código JavaScript que não é reconhecido, durante o desenvolvimento e execução do código o programador precisará saber a origem de qualquer erro existente no seu programa.

Para isto, é possível configurar o browser para exibir uma notificação de erro de script durante seus testes.

Utilizando o Internet Explorer o usuário poderá acessar o menu Ferramentas > Opções da Internet > Avançadas e selecionar a opção Exibir Notificação sobre cada erro de script conforme mostrado na figura a seguir:



Feito isto, qualquer erro existente em seu programa será notificado pelo browser de acordo com a figura abaixo:



Analisando a origem dos erros

É importante que o usuário na reparação de erros encontrados em seus scripts, isole suas funções e rotinas de modo que possa analisar etapa por etapa de seu programa.

Verificar minuciosamente cada trecho das linhas do script de maneira que encontre possíveis erros.

Muitos erros lógicos são causados por valores incorretos atribuídos as variáveis.

Para facilitar a localização destes erros, basta que o usuário defina áreas de observação em vários locais de seu script.

Estas áreas na maioria das vezes podem ser caixas de alerta para determinar se a execução do programa está chegando àquele ponto que se encontra.

Logo após a depuração de todo o script, o usuário terá apenas que remover estas áreas de observação.

Outros programadores, preferem ter em mãos uma cópia de todo o seu código para que se possa fazer as devidas correções. Muitas das vezes, o usuário enxerga no papel o que ele não vê na tela.

Outros erros comuns

1. Deixar de utilizar aspas em strings.
2. Má utilização das aspas e apóstrofos (aspas simples).
3. Sinal de igualdade individual para expressões de comparação.
4. Utilizar o objeto de formulário pertencente ao Documento e não a Janela.
5. Utilização dos métodos com outros objetos que não os possuem.
6. Criação de laços infinitos.
7. Falta da instrução return em uma função.

Eventos

Eventos são quaisquer ações iniciadas pelo usuário, isso significa, que qualquer alteração feita pelo usuário pode ser considerado como um evento, um clique em um botão, uma alteração em um objeto da janela, como você pode notar exitem várias formas de interação do usuário com eventos.

Sua aplicação é simples, você deve indicar estes eventos dentro das tags da linguagem HTML.

Veja exemplo:

```
<TAG nomeEvento="Funções">
```

TAG é qualquer comando da linguagem HTML.

nomeEvento é um dos comandos conhecidos da linguagem JavaScript.

O texto "**Funções**" indica quais são as funções ou alterações de propriedades que o JavaScript fará na página.

***Nota**: Quando quiser adicionar mais de um comando de JavaScript no mesmo evento, adicione ponto e vírgula (;) para separar as instruções, veja abaixo:*

```
<TAG nomeEvento="Função1;Função2;Função3">
```

TAG é qualquer comando da linguagem HTML.

nomeEvento é um dos comandos conhecidos da linguagem JavaScript.

O texto "**Função1;Função2;Função3**" indica quais são as funções ou alterações de propriedades que o JavaScript fará na página.

Utilização de Eventos

Evento onblur

Este evento ocorre quando o usuário tira o foco de um campo de formulário.

Veja exemplo:

1. Abra o editor de texto, insira o código abaixo e salve seu arquivo como **onblur.html**

```
<!doctype html>
<html>
   <head>
   <meta charset="UTF-8">
   <head>
   <title>Evento onBlur</title>
   </head>
   <body>
   <form name=”form1”>
   Digite seu nome:
   <input type=”text” name=”nome” onBlur=”alert('Saiu da
caixa de texto')”>
   </form>
   </body>
</html>
```

2. Teste seu arquivo no navegador para ver o resultado.

Varemos outro teste agora, que se o usuário deixar o campo em branco ele será aviso através de uma mensagem de alerta.

1. Altere o arquivo **onblur.html** deixando-o como abaixo:

```
<!doctype html>
<html>
<head>
   <meta charset="UTF-8">
   <title>Evento onBlur</title>
</head>
<body>
   <script>
   function aviso(){
    if(form1.nome.value == “”){
     alert(“Campo obrigatório”);
    }
   }
   </script>
   <pre>
   <form name=”form1”>
     Digite seu nome:
     <input type=”text” name=”nome” onBlur=”aviso()”>
   </form>
</body>
</html>
```

2. Teste seu arquivo no navegador para ver o resultado.

Evento onchange

Verifica se houve alterações no conteúdo de caixa de texto ou objetos de seleção, o evento só é chamado se alguma coisa for realmente alterada, este comando é semelhando ao comando onBlur.

Veja o exemplo:

```
<input type="text" name="nome"
onChange="alert('alterou!')">
```

Evento onclick

Um dos eventos mais utilizados em JavaScript, muito comum para criar interações com o usuário, funciona com a grande maioria dos componentes HTML.

Veja o exemplo:

```
<input type="button" name="botao" value="Enviar"
onClick="alert('Dados enviados!')">
```

Evento onfocus

Este evento é executado quando o usuário clica em caixas de texto ou em objetos de seleção. Este comando funciona de forma contrária ao do evento onBlur.

Veja exemplo:

```
Digite seu nome:
<input type="text" name="nome" onFocus="alert('Digite seu
nome!')"><br />
Digite seu endereço:
<input type="text" name="nome" onChange="alert('alterando')
onFocus="this.value='Rua Curso de JavaScript'">
```

Você conhecerá mais evento em outro capitulo.

Questionário

1. O que são expressões?

_____
_____
_____
_____

2. O que são operadores?

_____
_____
_____
_____

3. Qual a tarefa do operador incremental?

_____
_____
_____
_____

4. Para que são utilizadas as declarações no JavaScript?

_____
_____
_____
_____

5. Para que serve o uso de comentários na programação?

_____
_____
_____
_____

Aula 4 - Formulários e Controles

Tópicos da Aula

- Objeto form
- Manipuladores de evento para formulários
- Utilizando o objeto history
- Controles de decisão
- Comandos condicionais e repetição

Mais eventos

Evento onload

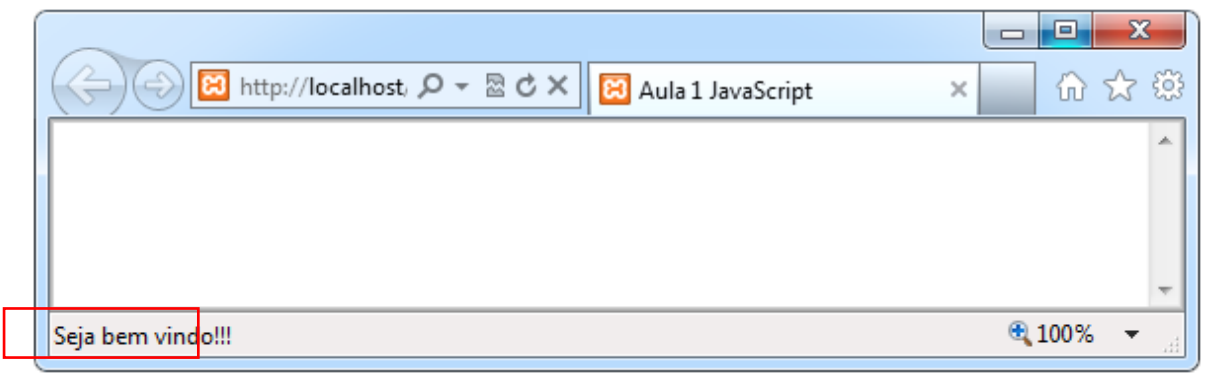
Este evento é executado quando a página HTML é carregada.

Veja o exemplo:

1. Abra seu editor de texto, adicione o código abaixo e salve como **carrega.html**

```
<!doctype html>
<html>
    <head>
    <meta charset="UTF-8">
    <head>
    <title>Aula de JavaScript</title>
    </head>
    <body onLoad="defaultStatus=('Seja bem vindo!')">
    </body>
</html>
```

2. Execute seu arquivo e veja o resultado no navegador



Este é um exemplo que pode ser utilizado para inúmeras interações com o usuário.

Evento onunload

Você também pode criar um evento que será chamado quando o usuário sair da página, isso é feito em diversos sites que tenham formulários gigantes, assim você consegue avisar o usuário que tentar fechar a página web sem querer.

Veja o exemplo abaixo:

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<head>
<title>Aula de JavaScript</title>
</head>
<body onLoad=”defaultStatus=(’Seja bem vindo!’)”
onUnLoad=”alert(‘Obrigado pela visita’)”>
</body>
</html>
```

Evento onmousemove

Com o evento onMouseMove o usuário criará uma ação que será acionada sempre o mouse for movimentado.

Veja exemplo:

```
<script>
var num = 0;
</script>
<body onMouseMove=”num++;defaultStatus=’Moveu : ‘ + num + ‘
vezes’”>
<h3>Olhe para a barra de Status</h3>
```

Evento onmouseover

Este evento ocorre quando o mouse passa sobre o objeto que vai receber o evento.

Veja exemplo:

```
<h3 onMouseOver="defaultStatus='Bem vindo a programação
JavaScript!'">Olhe para a barra de Status</h3>
```

Evento onmouseout

Este evento ocorre quando o mouse sai de cima do objeto que recebe o evento, note que no exemplo anterior a mensagem permanece na barra de status mesmo depois de tirar o mouse do link, vamos alterar um pouco nosso código para que fique um pouco mais interessante.

Veja exemplo:

```
<h3 onMouseOver="defaultStatus='Bem vindo a programação
JavaScript!'" onMouseOut=" defaultStatus='Vá até a barra de
Status!'">Olhe para a barra de Status</h3>
```

Evento onmousedown

Este evento é semelhante ao comando onclick, a principal diferença é que neste evento podemos controla se o usuário deixa o mouse clicado por bastante tempo, utilizado também em interações onde podemos arrastar objeto na tela.

Veja exemplo:

```
<h3 onMouseDown="alert('Clicou')">Clique aqui</h3>
<script>
function clicou() {
var total = 50;
for (i = 0; i < total; i++) {
      alert("Falei para não clicar");
}
}
</script>
<br /><h3 onMouseDown="clicou()">Não clique aqui</h3>
```

Evento onmouseup

Este evento funciona como inverso do comando onMouseDown, use sempre em conjunto para interações eficazes.

Evento onkeypress

Este evento é disparado quando o usuário pressiona um tecla do teclado, seja o físico ou virtual.

Evento onkeydown

Semelhante ao evento onKeyPress só que neste caso é possível verificar se o usuário deixa a tecla pressionada.

Evento onkeyup

Este evento ocorre quando o usuário para de pressionar a tecla.

Evento onselect

Este evento ocorre quando selecionamos alguma coisa em caixas de textos.

Veja exemplo:

```
<form name="form1">
Digite seu nome:
<input type="text" name="nome" onSelect="alert('O usuário
selecionou ' + this.value)">
```

Evento onsubmit

Este evento é utilizado para enviar dados através de formulários.

Veja exemplo:

```
<form name="form1" onSubmit="alert('Dados enviados')">
```

Este evento também é válido quando queremos fazer validações em todos os campos de um formulário.

Vamos criar um script simples que faz o controle do campo de texto verificando se o mesmo foi preenchido ou não.

Veja exemplo:

```
<script>
function teste() {
   if(form1.campo1.value==""){
     alert("Favor preencher");
     return false;
   } else {
     return true;
  }
}
</script>
<form name="form1" onSubmit="teste()">
Digite seu Nome: <input type="text" name="campo1">
```

Questionário

1. O que determina a propriedade method do objeto form?

_____
_____
_____
_____

2. O que determina a propriedade Port do objeto location?

_____
_____
_____
_____

3. O que é o objeto history do JavaScript?

_____
_____
_____
_____

4. Quais métodos representam os botões de avançar e voltar presentes no browser?

_____
_____
_____
_____

5. O que especifica a propriedade appVersion do objeto navigator?

_____
_____
_____
_____

Aula 5 - Funções e Objetos

Tópicos da Aula

- Funções da linguagem JavaScript
- Funções pré-programadas
- Objetos pré-contruídos
- Date
- String

Funções predefinidas

Funções predefinidas como o próprio nome descreve, são funções que acompanham a programação de JavaScript, com estas funções você pode permitir ao usuário realizar tarefas simples como fazer a impressão de uma página ou adicionar sua página aos favoritos.

Veja alguns exemplos.

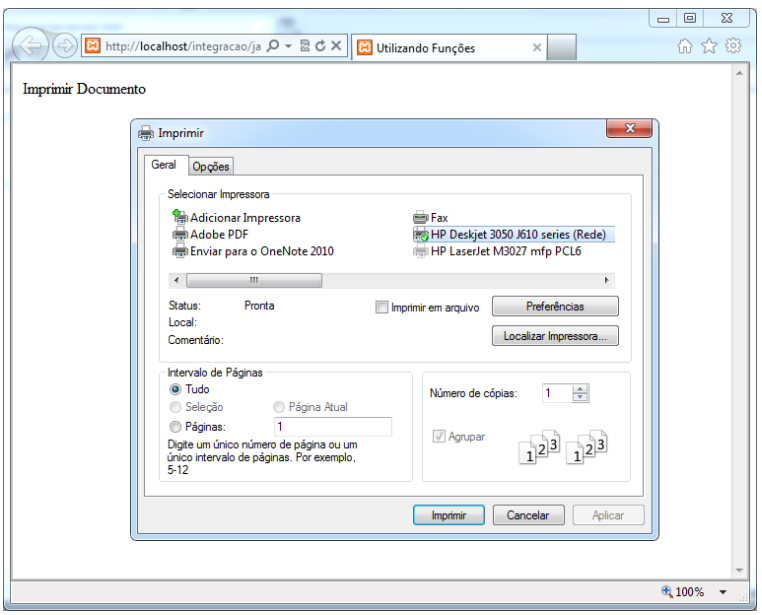
Função print()

Esta função possibilita que o usuário faça a impressão da página atual, sem ter que fazê-lo através de processos tradicionais como cliques em menus ou teclas de atalho, isso é um diferencial e facilitador para usuário com pouco conhecimento em informática.

Veja sintaxe básica.

```
<p onClick="self.print()">Imprimir Documento</p>
```

Ao clicar no texto que não é um link, o mesmo fará com que seja aberta a caixa de dialogo de impressão de arquivos, caso tenha um impressora instalada em seu computador será possível seleciona-lo e fazer a impressão clicando no botão de ok.



Adicionar ao favoritos

Segue a seguir, um exemplo de inserção de uma URL à pasta dos Favoritos.

Veja sua utilização e estude sua lógica, observe que é bem simples:

```
<script>
var url="http://www.google.com.br";
var titulo="Página do Google";
function Favoritos(){
   if(document.all)
     window.external.AddFavorite(url, titulo);
   }
}
</script>
<input type="button" value="Favoritos"
onClick="Favoritos()">
```

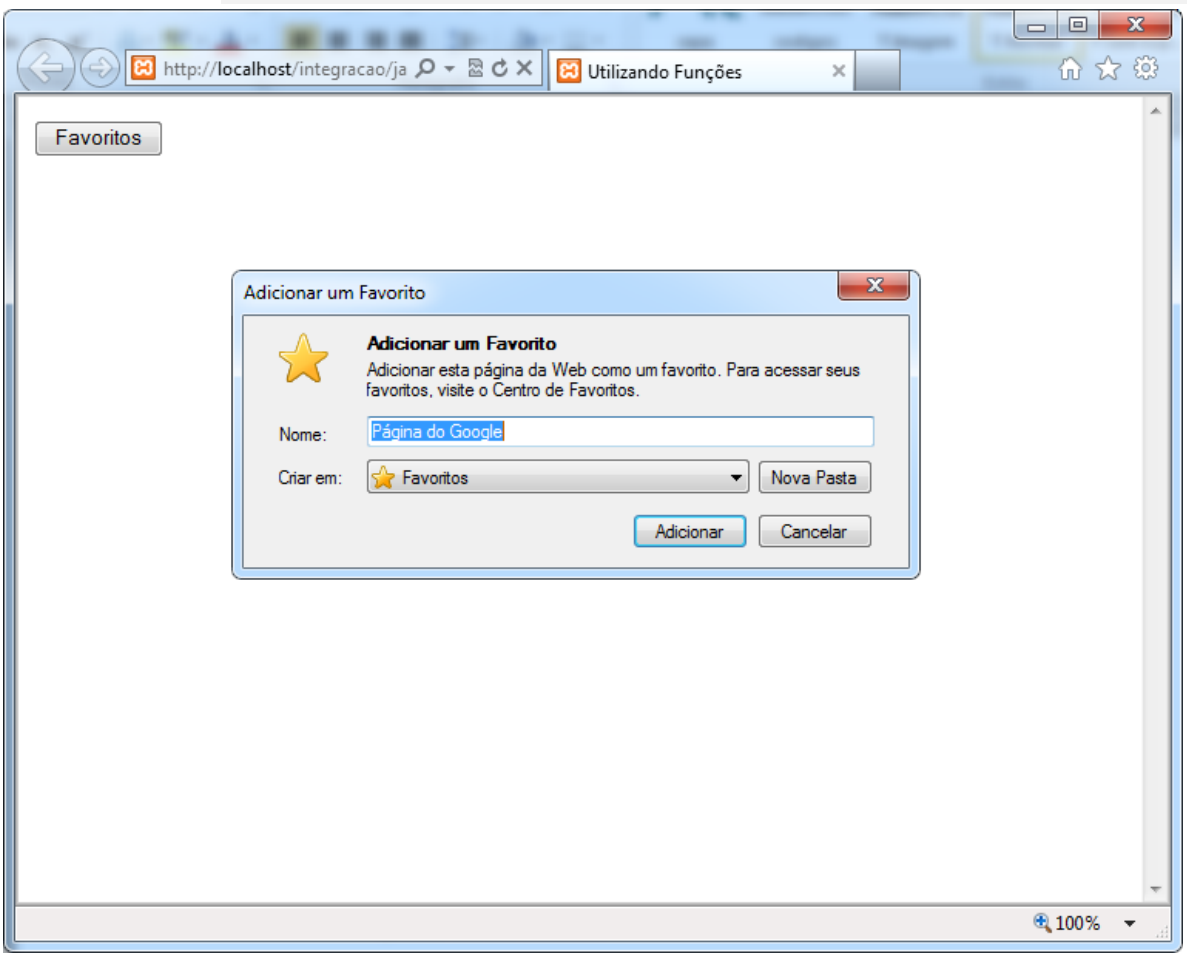


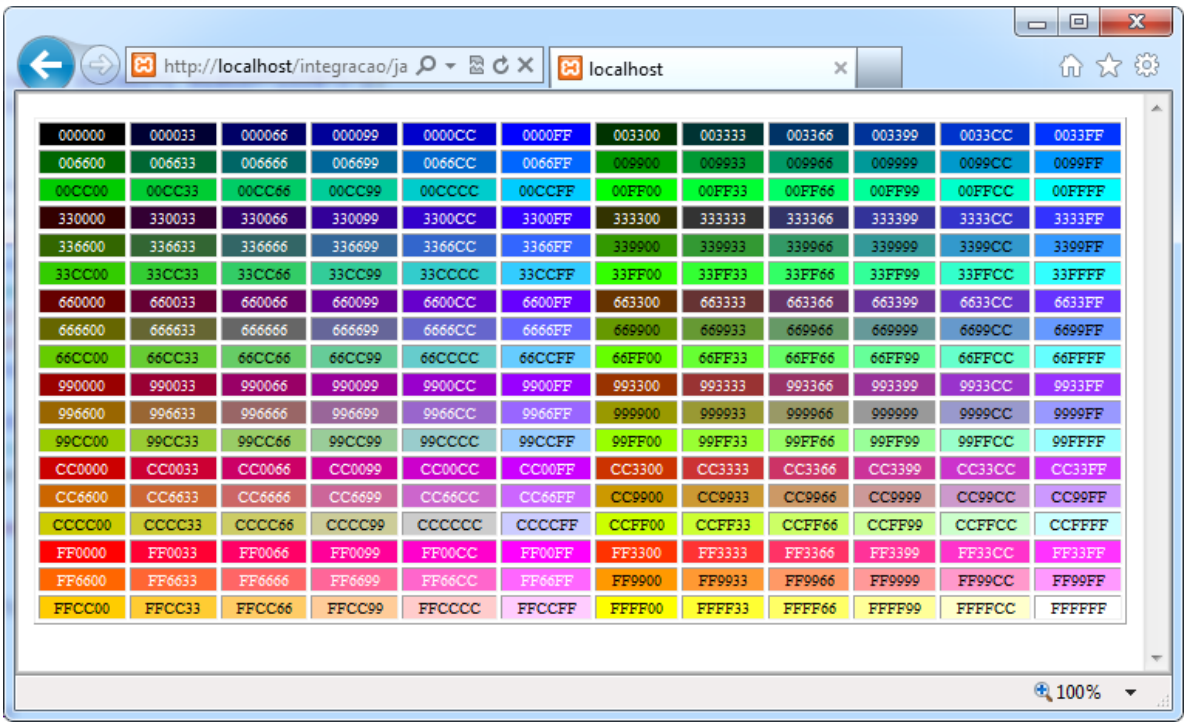
Tabela de cores

Neste simples cógido vamos criar uma tabela de cores para que você possa usá-la em aulas como HTML.

1. Abra seu editor de texto, adicione o código abaixo e salve como **tabeladecores.html**.

```
<script>
function geraTabela() {
document.write('<table border=1 width="100%">');
var valores = "00336699CCFF";
var r, g, b;
for (r=0;r<6;r++) {
for (g=0;g<6;g++) {
if (g%2==0) document.write("<tr>");
for (b=0;b<6;b++) {
cor = valores.substr(2*r, 2) +  valores.substr(2*g,2) +
valores.substr(2*b, 2);
document.write('<td align="center" bgcolor="#'+cor+'">');
if(g<3)
document.write('<font size="-2" color="white">');
else document.write('<font size="-2" color="black">');
document.write(cor+'</font></td>');
}
if(g%2==1) document.write('</tr>');
}
}
document.write('</table>');
}
</script>
</head>
<body onLoad="geraTabela()">
```

2. Teste seu arquivo em um navegador web.



Objetos pré-construídos

A linguagem JavaScript possui objetos padronizados para uso nos scripts.

Dentre eles, temos:

DATE Manipula datas e horas.

STRING Manipula caracteres.

MATH Realiza operações matemáticas.

Date

O objeto DATE permite que o usuário possa trabalhar com datas e horas. Para determinar um novo objeto de data, temos as seguintes sintaxes:

```
NomeObjeto = new Date();
NomeObjeto = new Date("mês dia, ano
horas:minutos:segundos");
```

```
NomeObjeto = new Date(ano, mês, dia);
NomeObjeto = new Date(ano, mês, dia, horas, minutos,
segundos);
```

Veja exemplos conforme sintaxe apresentada anteriomente:

```
Data = new Date(); // atribui a variável Data, a data e
hora corrente.
Data = new Date(1979, 05, 26); // atribui a variável Data,
a data 26 de maio de 1979.
```

Médotos do objeto Date

Os métodos getDate, getDay entre outros têm a finalidade de recuperar a data e hora de um objeto date. Ele poderá conter a data e hora atuais ou específicas. Após a especificação do objeto date, basta especificar qual o método veja pelos exemplos apresentados abaixo:

```
Data = new Date();
alert(Data.getDay()); // Retorna o dia atual.
alert(Data.getMonth()); // Retorna o mês atual.
alert(Data.getYear()); // Retorna o ano atual.
```

Métodos	**Descrição**
getDate	Dia da semana (0=Domingo).
getDay	Dia do mês.
getHours	Horas (0 a 23).
getMinutes	Minutos (0 a 59).
getMonth	Mês do ano (0=janeiro).
getSeconds	Segundos (0 a 59).
getTime	Milissegundos desde 1 de janeiro de 1990 (00:00:00).
getTimezone Offset	Diferença de fuso horário em minutos para o GMT.

getYear	2 digitos do ano até 1999. Após 2000, 4 dígitos.

Este objeto tem a função de armazenar a data e hora atuais no formato mm/dd/aa hh:mm:ss. Os valores do mês são contados de 0 até 11 e os dias da semana de 0 a 6 da seguinte forma:

Valor	Mês	Dia da Semana
0	Janeiro	Domingo
1	Fevereiro	segunda-feira
2	Março	terça-feira
3	Abril	quarta-feira
4	Maio	quinta-feira
5	Junho	sexta-feira
6	Julho	Sábado
7	Agosto	
8	Setembro	
9	Outubro	
10	Novembro	
11	Dezembro	

As horas são determinadas de 00:00 ás 23:00.

O objeto date pode definir data e hora a partir de 1 de janeiro de 1970. Após a criação do objeto date, o mesmo pode ser usado com qualquer método apresentado anteriormente.

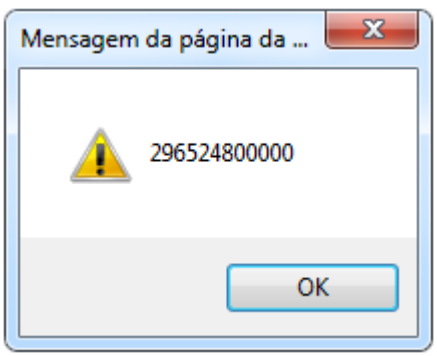
Método parse e utc

O método parse retorna o valor de milissegundos a partir de 1 de janeiro de 1970, 00:00:00, já o método UTC faz a mesma coisa, porém no fuso horário GMT.

Vejamos um exemplo da apresentação da quantidade milissegundos contados até o dia 26 de maio de 1979.

```
<script>
alert(Date.parse("May 26, 1979 GMT"));
</script>
```

Teste e veja se o resultado será 296524800000 milissegundos contados desde 1 de janeiro de 1970.



Veja outro exemplo de script apresentando na tela a hora atual, dia atual e as horas e minutos atuais.

```
<script>
hoje = new Date();
alert("a hora atual é " + hoje.getHours());
alert("o dia atual é " + hoje.getDay());
alert("agora são: " + hoje.getHours() + ":" + hoje.getMinutes());
</script>
```

Métodos set do objeto date

Os métodos setDate, SetDay entre outros, permitem ao usuário definir a data e a hora de um objeto date. Estes métodos são utilizados da mesma forma dos métodos get.

Vejamos a relação destes métodos:

Métodos	Descrição
setDate	Dia da semana (0 para Domingo).
setHours	Horas (0 a 23).
setMinutes	Minutos (0 a 59).
setMonth	Mês do ano.
setSeconds	Segundos (0 a 59).
setTime	Milissegundos de 1 de janeiro de 1990.
setYear	Ano.

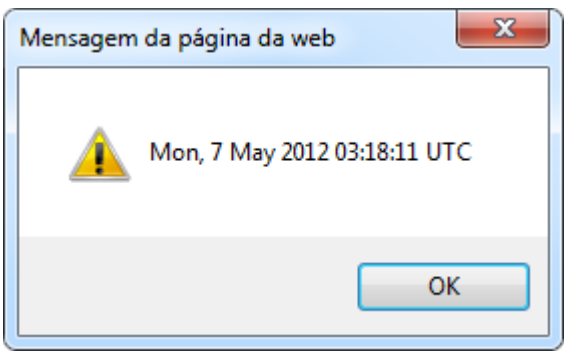
Método togmtscring

A definição de GMT é Greenwich Mean Time, que define o fuso horário internacional padrão para configuração de relógios. Este método faz a conversão de um objeto date para uma string usando convenções GMT.

Veja pelo exemplo a seguir, a conversão da hora atual em uma string no formato GMT. Certifique-se que o computador esteja com a definição de fusohorário correta.

```
alert(data.toGMTString());
```

O resultado, será a criação de uma string no formato:



Método tolocalestring

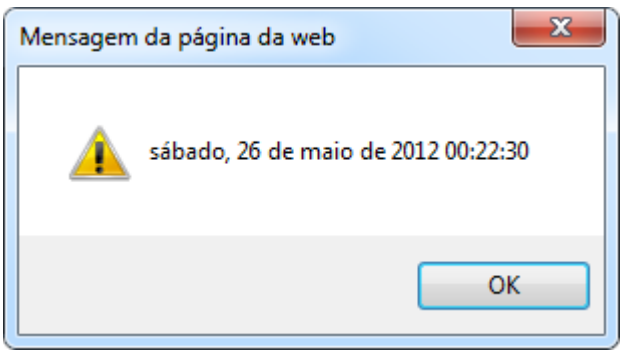
O método toLocaleString tem a função de formatar a data e a hora usando as convenções de string no computador local. Por exemplo: USA, Reino Unido, França, Alemanha, entre outros.

A idéia principal deste método é apresentar ao usuário a data e hora de forma que o mesmo possa interpretar de maneira simples na página, mesmo estando fora de sua localidade.

Veja o exemplo de utilização deste método:

```
<script>
data = new Date();
alert(data.toLocaleString());
</script>
```

O resultado esperado, será similar ao formato abaixo:



Vejamos agora um exemplo que irá apresentar um relógio digital na barra de status que fará a hora se atualizar de um em um segundo.

Analise o código apresentado abaixo:

Data e hora atualizada:

1. Abra seu editor de texto, adicione o código abaixo e salve como **relogio.html**.

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Relógio</title>
<script>
      function DataHora() {
      var data = new Date();
      tempo.innerHTML = data;
      setTimeout("DataHora()", 1000);
```

```
      }
</script>
</head>
<body onLoad="DataHora()">
<span id="tempo"></span>
</body></html>
```

String

Os objetos string são de nível superior.

SINTAXE

```
var versao = "10";
var curso = "JavaScript";
```

Veja a seguir a relação das propriedades do objeto String:

length Comprimento de uma string.

Os métodos do objeto string permitem a manipulação do objeto. O usuário poderá utilizar string literal ou de variáveis. Vejamos sua sintaxe abaixo:

```
"String literal".metodo();
TextoString="string de variável";
TextoString.metodo();
```

Método anchor

Este método tem a função de criar uma âncora a partir de uma string. Este método é similar à criação de uma âncora utilizando o tag HTML <a name="valor">, o mesmo ocorreria se definir string.anchor(valor).

Vejamos a sintaxe de utilização do método anchor:

```
String.anchor(nome);
```

Veja um exemplo de utilização deste método:

```
<script>
Ancora = "Início do Documento";
valor = Ancora.anchor("inicio");
document.write(valor);
</script>
```

Este script poderia ser utilizado pela linguagem HTML através do seguinte código:

```
<a name="inicio">Início do Documento</a>
```

Método big

Este método substitui o tag HTML <BIG>, que tem a função de aumentar a fonte e atribuir o estilo de negrito. Para utilizá-lo, siga a seguinte sintaxe:

String.big();

Veja o exemplo de utilização deste método:

```
<script>
texto="CURSO JAVASCRIPT";
document.write(texto,"<br>");
document.write(texto.big());
</script>
```

Método small

Este método substitui o tag HTML <SMALL> que tem a função de reduzir o tamanho da fonte. Para utilizá-lo, siga a seguinte sintaxe:

```
String.small();
```

Veja o exemplo de utilização deste método:

```
texto="CURSO JAVASCRIPT";
document.write(texto, "<br>");
document.write(texto.small());
```

Método bold

Referente ao tag HTML <B> que tem a função de atribuir o estilo de negrito sobre o texto.

Sua sintaxe segue o seguinte padrão:

```
String.bold();
```

Veja o exemplo de utilização deste método:

```
texto="CURSO JAVASCRIPT";
document.write(texto, "<br>");
document.write(texto.bold());
```

Método italics

Este método é referente ao tag HTML <I> que atribui o estilo de itálico em um texto. Sua sintaxe segue o mesmo padrão do método bold. Veja abaixo um exemplo da utilização do método italics.

```
texto="CURSO JAVASCRIPT";
document.write(texto, "<br>");
document.write(texto.italics());
```

Método strike

Este método tem a função de criar um texto tachado que exibe uma linha no meio do texto exibido. Este método tem a mesma função do tag HTML <STRIKE>.

Sua sintaxe básica segue o seguinte padrão:

```
texto="CURSO JAVASCRIPT";
document.write(texto, "<br>");
document.write(texto.strike());
```

Método fontcolor

Determina a cor da fonte em um texto de acordo com o tag HTML <FONT COLOR>.

```
String.fontcolor(cor);
```

Exemplo de utilização do método fontcolor:

```
texto="CURSO ";
document.write(texto.fontcolor("red"));
document.write("JAVASCRIPT".fontcolor("blue"));
```

O método fontcolor aceita nomes de cores sólidas, assim como, os valores hexadecimais da cor referente.

Método fontsize

Este método, determina o tamanho da fonte seguindo os padrões do tag HTML <FONT SIZE> que possui tamanhos que vão de 1 a 7, assim como a possibilidade de valores relativos através dos sinais de + e -. Sua sintaxe básica segue o seguinte padrão:

```
texto="CURSO JAVASCRIPT";
document.write(texto, "<br>");
document.write(texto.fontsize(7));
```

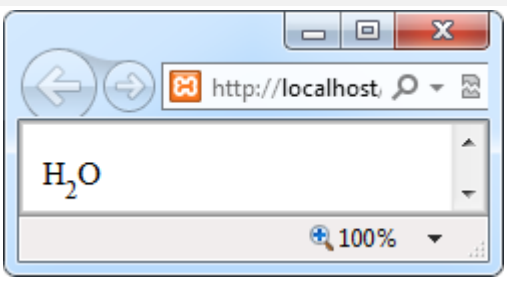
Método sub

Este método cria um texto subscrito tendo o mesmo efeito do tag HTML <SUB>. Sua sintaxe básica tem a seguinte formação:

```
String.sub();
```

Veja um exemplo para sua utilização:

```
texto="2";
document.write("H"+texto.sub()+"O");
```



Método sup

Este método cria um texto sobrescrito tendo o mesmo efeito do tag HTML <SUP>.

Sua sintaxe básica tem a seguinte formação:

```
String.sup();
```

Veja um exemplo para sua utilização:

```
texto="2";
```

```
document.write("16"+texto.sup);
```

Método charat

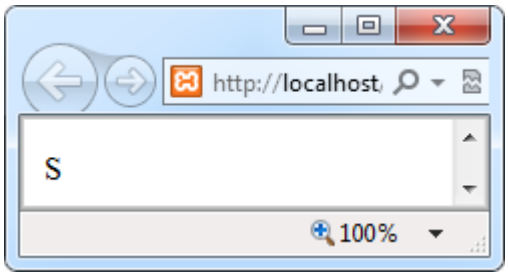
Com este método o usuário poderá retornar o caractere em uma determinada posição em uma string. Por exemplo, temos a string FABIO e a posição de referência é 1, com base nisto o caractere de retorno é "A". Estas posições são contadas à partir de 0 da esquerda para a direita.

Sintaxe:

```
String.charAt(valor);
```

Veja o exemplo de utilização do método charAt:

```
texto="JavaScript";
document.write(texto.charAt(4));
```



Método indexof

Com o método indexOf o usuário pode retornar a posição de um caractere dentro de uma string. Um exemplo claro do método indexOf, é a maneira de saber se determinada string possui algum caractere específico.

Caso a string não contiver o caractere específico, o método irá retornar o valor –1, caso haja a ocorrência do caractere procurado, será retornado o valor 0 ou superior, sendo que 0 é a posição do primeiro caractere da string, 1 a posição do segundo caractere e assim por diante. Caso exista duplicidade do caractere específico, o método irá retornar a posição do primeiro caractere encontrado.

Sua sintaxe segue o seguinte padrão:

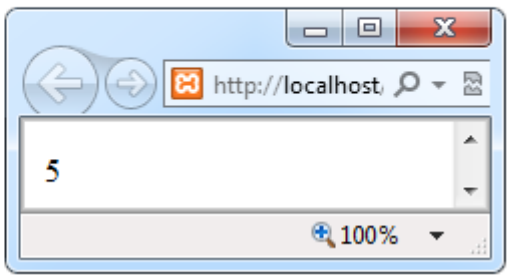
String.indexOf(valor)

Veja pelo exemplo a utilização do método indexOf:

```
texto="curso@seuemail.com";
document.write(texto.indexOf("@"));
```

```
<script>
texto="curso@seuemail.com";
document.write(texto.indexOf("@"));
</script>


</head>
<body>
</body>
</html>
```



Uma das práticas utilizações deste método, é determinar se determinado valor de uma string existe ou não, se o valor não existir é retornado o valor -1.

Método lastindexof

Com o método lastIndexOf o usuário poderá retornar a última posição de um determinado caractere em uma string. Um exemplo de utilização deste método é a de retornar a posição de um caractere barra (/) em uma string, para por exemplo utilizar com URL's.

Sua sintaxe básica, segue o seguinte exemplo:

String.lastIndexOf(valor, compensar);

Onde valor, é o caractere que deseja procurar dentro da string.

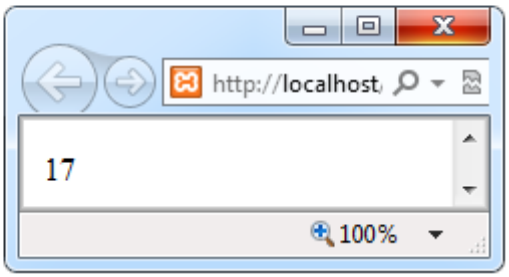
Onde compensar, é a posição na string a partir de onde o usuário deseja começar a pesquisa.

Veja abaixo um exemplo que localiza a última ocorrência da letra "M" na string curso@seuemail.com utilizada como exemplo.

```
texto="curso@seuemail.com";
document.write(texto.lastIndexOf("m"));
```

```
<script>
texto="curso@seuemail.com";
document.write(texto.lastIndexOf("m"));
</script>


</head>
<body>
</body>
</html>
```



O resultado será 17. É bom lembrar que as strings sempre começam a contagem em 0).

Questionário

1. O que são Funções em JavaScript?

_____
_____
_____
_____

2. Qual finalidade na instrução return?

_____
_____
_____
_____

3. O que avalia a função eval?

_____
_____
_____
_____

4. Qual função permite ao usuário executar a função de imprimir?

_____
_____
_____
_____

5. Dentre os objetos pré-construídos quais temos?

_____
_____
_____
_____

Aula 6 - Matemática e Array

Tópicos da Aula

- Objeto link
- O Objeto Math
- Metodos do Objeto Math
- Objeto Image
- Array
- Resumo geral de objetos JavaScript
- Resumo geral de métodos JavaScript

Objeto link

Propriedades do objeto links

Propriedades	Descrição
hash	Especifica o texto seguido da simbologia "#" em um URL.
host	Contém o hostname:port de um URL.
hostname	Especifica o host e o domínio (endereço IP) de um URL.
href	Especifica o URL inteiro.
length	Determina o número de âncoras de um documento.
pathname	O path atual do URL.
port	Especifica a porta lógica de comunicação obtida da URL.
protocol	Especifica o protocolo da URL.
search	Especifica a porção search da URL.
target	Determina o link de destino.

Método link

Este método é similar ao tag HTML <A HREF> que tem a função de criar hiperlinks em uma página. Sua sintaxe básica tem a seguinte formação:

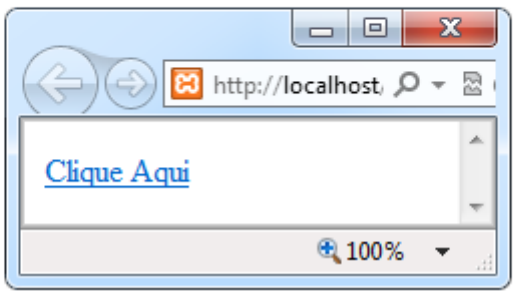
```
String.link(href);
```

Onde é href é a referência de vínculo do hiperlink.

Vejamos um exemplo:

```
<html><head>
<title>JavaScript</title>
<script>
texto = “Clique Aqui”;
document.write(texto.link(“http://www.google.com”));
</script>
</head>
<body>
</body>
</html>
```

```
<html><head>
<title>JavaScript</title>
<script>
texto="Clique Aqui";
document.write(texto.link("http://www.google.com"));
</script>
</head>
<body>
</body>
</html>
```



Método replace

Este método tem a função de trocar valores dentro de uma string. Sua sintaxe básica tem a seguinte formação:

```
String.replace(s1,s2);
```

Onde s1 é o caractere procurado dentro de uma string.

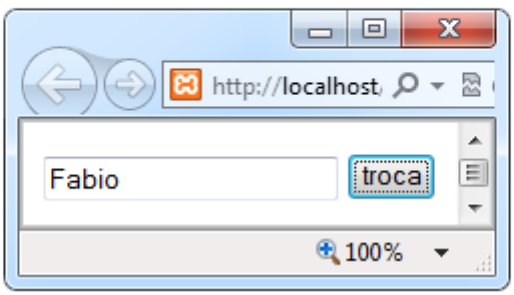
Onde s2 é o novo caractere que será trocado por s1.

Vejamos um exemplo simples que ao ser digitado um nome com acento agudo na letra A, ao clicar sobre o um botão é trocado a letra sem acento.

```
<!doctype html>
<head>
```

```
<script>
function troca() {
      texto = document.form1.nome.value;
      document.form1.nome.value = texto.replace("á","a");
      }
</script>
<body>
<form name="form1">
<input type="button" onClick="troca()" value="troca">
</form>
</body>
</html>
```

```
<form name="form1">
<input type="text" name="nome" value="Fábio" />
<input type="button" onClick="troca()" value="troca">
</form>
</body>
</html>
```



Método substring

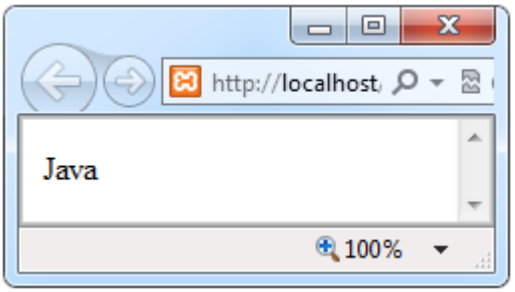
Este método retorna uma parte de uma string. O usuário poderá especificar o início e o final da parte que deseja extrair indicando a posição inicial como 0, já a posição final é determinada com a instrução string.length-1, isto é, um a menos do que o comprimento da string. Sua sintaxe básica tem a seguinte

Formação:

```
String.substring(inicio, fim);
```

Vejamos um exemplo da utilização do método substring:

```
<script>
texto="JavaScript";
document.write(texto.substring(0, 4));
</script>
```

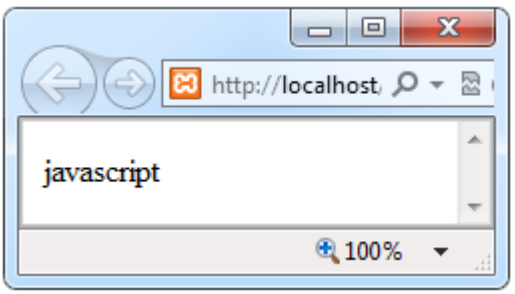


Valor retornado: Java.

Método tolowercase

Com o método toLowerCase o usuário poderá converter uma string em letras minúsculas. Sua sintaxe básica segue o seguinte padrão:

```
<script>
texto=”JAVAScript”;
document.write(texto.toLowerCase());
</script>
```

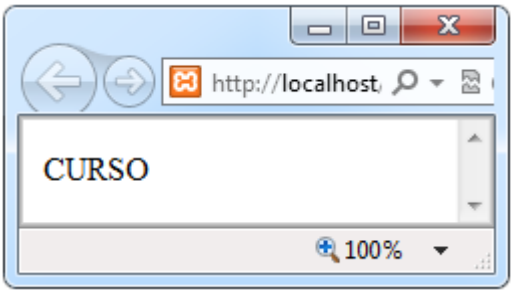


Veja que o conteúdo da variável texto está em letras maiúsculas, com o uso do método toLowerCase, este texto será apresentado no documento em letras minúsculas.

Método touppercase

Com o método toUpperCase, o usuário poderá converter uma string em letras maiúsculas. Sua sintaxe básica segue o seguinte padrão:

```
<script>
texto=”curso”;
document.write(texto.toUpperCase());
</script>
```



O objeto math

Este objeto é utilizado para realizar operações matemáticas. Estas operações podem ser aritméticas, funções trigonométricas, funções de arredondamento e comparação. A sintaxe de utilização dos métodos deste objeto seguem a seguinte sintaxe:

```
Math.método(valor)
```

Ou

with (Math) {

método (valor);

}

Propriedades de cálculo do objeto math

Veja na tabela abaixo a relação das propriedades do objeto Math:

Propriedades	Descrição
E	Constante de Euler e a base dos logaritmos naturais (próximo de 2,118).
LN2	Logaritmo natural de 2.
LN10	Logaritmo natural de 10.
LOG2E	Logaritmo na base 2 de E
LOG10E	Logaritmo na base 10 de E

PI	Equivalente numérico de PI arrendondado para 3,14.
SQRT1_2	Raiz quadrada de um meio
SQRT2	Raiz quadrada de 2

No exemplo que foi utilizado a estrutura with, permite ao usuário utilizar uma série de métodos math sem a necessidade de acrescentar varios Math.Objeto, facilitando todo um trabalho.

Instrução with

Esta instrução faz com que um objeto se torne default para uma série de opções existentes. Normalmente esta instrução é utilizada com o objeto Math, uma vez que ele exige que o usuário especifique o nome do objeto quando acessar qualquer uma de suas propriedades.

Veja sua sintaxe:

```
with (objeto) {
      intruções
}
```

Vejamos alguns exemplos de sua utilização:

```
<script>
      alert(Math.PI);
      alert(Math.round(1234.5678));
</script>
```

Utilizando a instrução with o usuário irá determinar os valores que deseja economizando tempo na aplicação. Observe como ficaria estas instruções aplicadas com a instrução with:

```
<script>
with (Math) {
      alert(PI);
      alert(round(1234.5678));
</script>
```

Veja pelo exemplo anterior, que o usuário não necessitou utilizar o objeto Math várias vezes. Outra questão, é que a instrução with não é utilizada somente com o objeto Math. Ela poderá ser usada com a maioria dos outros objetos da linguagem JavaScript.

Métodos do objeto math

Abs

Este método tem a função de retornar o valor absoluto de um número. Isto significa que o valor será sempre positivo. Caso seja utilizado um valor negativo à este método. Ele será retornado como positivo. Por exemplo, caso seja definido o valor –123, ele será convertido para 123.

Veja o exemplo abaixo:

```
<script>
valor=Math.abs(-123);
alert(valor);
</script>
```

Neste exemplo foi definido à variável valor o método abs do objeto Math que possui o valor negativo –123, em seguida foi solicitado através de uma caixa de alerta a exibição do conteúdo da variável valor que foi convertido em número positivo.

Acos

Este método irá retornar o arco co-seno (em radianos) de um número.

Vejamos o exemplo a seguir:

```
<script>
valor=Math.acos(0.12);
alert(valor);
</script>
```

O script acima irá retornar o resultado: 1.4505064444001085

Asin

O método asin retorna o arco seno (em radianos) de um valor.

Veja o exemplo a seguir:

```
<script>
valor=Math.asin(0.12);
document.write(valor);
</script>
```

O script acima irá retornar o resultado: 0.12028988239478806

Ceil

Este método retorna um inteiro maior que ou igual a um número. O resultado deste método é equivalente ao arredondamento de um número.

Claro que a lógica do arredondamento de um número é que se um número é um valor positivo como por exemplo 12,6 – o resultado é 13, quando o número for um valor negativo, por exemplo –12,6 – o resultado é –12. Vejamos o exemplo do uso do método ceil.

```
<script>
valor=Math.ceil(12.6);
valor2=Math.ceil(-12.6);
alert(valor);
alert(valor2);
</script>
Os resultados retornados serão: 13 e -12.
```

Cos

Este método irá retornar o co-seno (em radianos) de um número. Vejamos o exemplo a seguir:

```
<script>
valor=Math.cos(0.12);
alert(valor);
</script>
```

O resultado obtido será: 0.9928086358538662

Exp

Este método irá retornar o logaritmo do número na base E.

Vejamos um exemplo:

```
<script>
valor=Math.exp(0.0009);
alert(valor);
</script>
```

O resultado obtido será: 1.0009004051215273

Floor

Retorna o maior inteiro menor ou igual a um número. Vejamos o exemplo:

```
<script>
valor=Math.floor(101.25);
valor2=Math.floor(-101.25);
alert(valor);
alert(valor2);
</script>
```

Com isto teremos o seguinte resultado: 101 e –102.

Log

Retorna o logaritmo natural de um número (base E). Vejamos o exemplo a seguir:

```
<script>
valor=Math.log(1.1);
alert(valor);
</script>
```

Resultado: 0.09531017980432493

Max

Retorna o maior valor entre dois números. Vejamos o exemplo a seguir:

```
<script>
valor=Math.max(5,10);
alert(valor);
</script>
```

Resultado: 10.

Min

Retorna o menor valor entre dois números. Vejamos o exemplo a seguir:

```
<script>
```

```
valor=Math.min(5,10);
alert(valor);
</script>
```

Resultado: 5.

POW (base,expoente)

Retorna a base elevada à potência do expoente, por exemplo, 2 elevado à décima potência é 1024. Com o método pow apresenta-se os argumentos de base e de expoente.

Vejamos este exemplo o seu resultado:

```
<script>
valor=Math.pow(1024,2);
alert(valor);
</script>
```

Resultado: 1.048.576.

Random

Retorna um número aleatório entre 0 e 1 com até 15 dígitos. Este número aleatório é definido através do relógio do computador. Veja pelo script a seguir a apresentação de um número aleatório:

```
<script>
alert(Math.random());
</script>
```

Round

Com o método round é possível arredondar um valor. O arredondamento segue a regra de arredondamento mostrada anteriormente.

Vejamos o exemplo:

```
<script>
valor=Math.round(125.6);
alert(valor);
</script>
```

Sin

Este método retorna o seno de um número. Veja o exemplo a seguir:

```
<script>
valor=Math.sin(1.6);
alert(valor);
</script>
```

Resultado: 0.9995736030415051.

Sqrt

Retorna a raiz quadrada de um número.

```
<script>
valor=Math.sqrt(49);
alert(valor);
</script>
```

Resultado: 7.

Tan

Retorna a tangente de um número.

```
<script>
valor=Math.tan(1.5);
alert(valor);
</script>
```

Resultado: 14.101419947171718.

Objeto image

Na linguagem JavaScript as imagens que são inseridas através da linguagem HTML são consideradas cada uma um objeto do tipo IMAGE. Com isto, podemos concluir que as imagens possuem propriedades e métodos assim como os outros objetos já existentes. Através deste objeto é possível que o usuário possa interagir melhor e dinamicamente as imagens utilizadas em suas páginas.

Vejamos pelo exemplo a seguir a instrução HTML que insere uma imagem em uma página.

```
<img src=”foto1.jpg” />
```

Até aqui tudo bem, mas note que fora atribuído uma variável nesta imagem através do atributo name. Esta variável serve para fazer referência à imagem atualmente inserida na página no código JavaScript que será desenvolvido.

Vamos agora inserir um botão de formulário que será responsável pelo evento que iremos desenvolver, logo nosso código ficará da seguinte forma:

```
<img src="foto1.jpg" name="foto" /><br>
<input type="button" value="Muda Figura">
```

No código a seguir, foi utilizado o evento onClick que determinará a ação para o botão. Esta ação foi designada para trocar a imagem atualmente inserida por outra imagem. Note que foi feita uma referência para inserir à nova imagem no local da imagem atual.

```
<img src="foto1.jpg" name="foto" /><br>
<input type="button" value="Muda Figura"
onClick="document.foto.src='foto2.jpg'">
```

Em análise sobre este código simples, foi determinado que no documento atual, especificamente no objeto chamado "foto" que trata a figura atualmente inserida, será originada outra imagem "foto2.jpg" em seu local atual.

Resultando na troca da imagem. Veja agora o mesmo código fazendo alternação entre as duas imagens de acordo com a opção escolhida, observe:

```
<img src="foto1.jpg" name="foto" /><br>
<input type="button" value="Muda Figura"
onClick="document.foto.src='foto2.jpg'">
<input type="button" value="Volta Figura"
onClick="document.foto.src='foto1.jpg'">
```

Observe agora a criação de uma função que fará com que quando o usuário mover o mouse sobre a imagem a mesma será ampliada e ao movimentar para fora da imagem, a mesma retornará ao seu tamanho original:

```
<html><head>
<title>JavaScript</title>
</head>
<body>
<script>
function figura(valor) {
      document.foto.width=valor;
```

```
}
</script>
<img src=”foto1.jpg” name=”foto” onMouseOver=”figura(150)”
onMouseOut=”figura(70)” />
</body>
</html>
```

Array

Utilizando arrays

Primeiramente, saiba que um ARRAY é um grupo de itens que são tratados como uma única unidade. Um exemplo disto, é o grupo de meses do ano estarem dentro de um array chamado meses.

Os elementos de um array podem ser strings, números, objetos ou outros tipos de dados.

Para que se possa declarar um array, use a seguinte sintaxe:

```
NomeArray = new Array(numElementos);
```

Veja como declarar um array chamado meses e seus elementos.

```
Meses = new Array(12);
```

Outra maneira, é declarar os valores para o novo array criado. Observe a sintaxe abaixo:

```
Meses = new Array("janeiro", "fevereiro", "março", "abril",
"maio", "junho", "julho", "agosto", "setembro", "outubro",
"novembro", dezembro");
```

Quando atribuído o número de elementos no array, é necessário declarar os elementos que farão parte do mesmo.

Utilize a seguinte sintaxe:

```
NomeArray[numElementos]
Meses[0]=janeiro;
Meses[1]=fevereiro;
Meses[2]=março;
```

E assim por diante...

Veja pelo exemplo do script abaixo a apresentação da data atual presente no navegador:

```
<html><head>
<script>
// Array com os dias da semana
hoje=new Date();
samana = new Array("Domingo", "Segunda", "Terça", "Quarta", "Quinta",
"Sexta", "Sábado");
// Array com os meses do ano
meses = new Array("Janeiro", "Fevereiro", "Março", "Abril", "Maio", "Junho",
"Julho", "Agosto", "Setembro", "Outubro", "Novembro", "Dezembro");
document.write("Hoje é: ", semana[hoje.getDay()],", ",
meses[hoje.getMonth()], " de ", hoje.getYear());
</script>
</body>
</head>
```

Neste exemplo foi declarado uma variável chamada hoje que define seu conteúdo como valores de data e criados dois arrays, o primeiro chamado semana e o outro chamado meses. Cada um dos arrays possui como conteúdo os dias da semana e os meses do ano.

Finalizando, foi utilizado o objeto document.write que apresentará a variável hoje com o array correspondente da variável semana de acordo com seu método getDay() que apresenta o valor especificado do dia da semana. Ocorrendo o mesmo com a variável meses para os meses do ano.

Veja agora outro exemplo da utilização dos arrays fazendo com que seja criado vários campos de formulário de acordo com a quantidade definida pelo usuário. Neste exemplo quando o usuário carrega a página é solicitado a quantidade de campos que deseja criar, para isto foi definido o seguinte código:

```
<form name="form1">
<script>
nome = prompt("digite a quantidade","");
</script>
```

Em seguida foi criado um laço for que caso o valor da variável i for menor que a quantidade referenciada na variável nome, será incrementado o valor de nome dentro da variável i. analise o código a seguir:

```
for(i=0;i<nome;i++){
      document.write("<br>Nome ",[i],":<input type=text nome=campo", [i],
">");
}
```

Para a execução do laço foi definido que será criado no documento atual um campo de formulário do tipo texto e a variável de cada campo criado que aqui chamada de campo, receberá cada uma o valor de i cada vez que o laço se repete. Com isto serão criados os campos cada um nomeado da seguinte forma:

Se o usuário informar 5 campos, serão criados cinco campos cada um chamado de: campo0, campo1, campo2, campo3, campo4. Lembre-se que um array sempre inicia-se a partir de 0. faça um teste e veja o resultado obtido.

Criaremos agora fora do script um botão de formulário que ao clicar sobre ele, será exibido em um caixa de alerta o valor que o usuário digitou em um determinado campo.

Analise o código a seguir:

```
<input type="button" value="ok"
onClick="alert(form1.campo3.value)">
```

Veja a seguir o código completo:

```
<html><head>
<title>JavaScript</title>
</head>
<form name="form1">
nome = prompt("digite a quantidade","");
for(i = 0; i < nome; i++){
  document.write("<br>Nome ", [i], ":<input type=text   name=campo",[i],">");
}
</script>
<input type="button" value="ok" onClick="alert(form1.campo3.value)">
</body>
</html>
```

Array anchors[]

Este array lista todas as âncoras existentes em um documento. Este objeto possui a propriedade length e é uma propriedade do objeto document.

Sintaxe:

```
document.anchors.length
```

Veja um exemplo de um script que informará a quantidade de âncoras existentes no documento.

```
<body>
<a name=1>primeira âncora</a>
<a name=2>segunda âncora</a>
<a name=3>terceira âncora</a>
<a name=4>quarta âncora</a>
<script>
ancoras = document.anchors.length;
alert("Esta página possui " + ancoras);
</script>
```

No script apresentado, foi definido na página quatro âncoras a partir do tag HTML <a name> e em seguida já no script, foi criada a variável ancoras que contará a quantidade de âncoras existentes na página através da propriedade length e logo depois, é executado a instrução alert que tem a função de exibir uma mensagem na tela informando o conteúdo da variável ancoras.

Array elements[]

O array elements[] tem a finalidade de listar todos os controles de um formulário. Sua sintaxe tem a seguinte formação:

```
document.NomeForm.elements[x].propriedade;
document.NomeForm.elements.length;
```

x é o número de elementos presentes dentro do formulário também iniciado com zero.

No exemplo a seguir, foi criado um código que tem a função de selecionar uma lista de caixas de verificação de um formulário quando é acionado um botão.

Observe o código:

```
<script>
function seleciona() {
      itens=document.form1.elements;
      for(i=0;i<itens.lenght;i++) {
            document.form1.elements[i].checked=true;
      }
}
function tira() {
      itens=document.form1.elements;
      for(i=0;i<itens.lenght;i++) {
            document.form1.elements[i].checked=false;
      }
}
```

Neste exemplo, foi criado uma função chamada seleciona() que cria uma variável que receberá os elementos do formulário form1. Em seguida, foi criado um laço for que somará a variável i a quantidade de elementos presentes no formulário, onde cada elemento deverá receber para sua propriedade checked o valor verdadeiro, ou seja, selecionar a caixa de verificação.

Logo mais, foi criada uma função chamada tira() que tem a função contrária da função seleciona().

Faça um teste e veja o que acontece.

No script abaixo é apresentado uma função que exibe o dia da semana mais as horas sendo atualizadas de um em um segundo:

```
<html><head>
<body>
<script>
function relogio()
tempo=new Date();
dia= new Array("Domingo", "Segunda", "Terça", "Quarta", "Quinta", "Sexta",
"Sábado");
hora=tempo.getHours();
min=tempo.getMinutes();
sec=tempo.getSeconds();
if(sec<10)
```

```
      sec="0"+sec;
}
defaultStatus=dia[tempo.getDay()]+", " + hora + ":" + min + ":" + sec;
setTimeout("relogio()", "1000");
}
</script><body onLoad="relogio()">
</body>
</html>
```

Resumo geral de objetos JavaScript

Objeto	Descrição
anchor (âncora)	Link de hipertexto para a mesma página. As âncoras dependem do objeto document (documento).
anchors[] (âncoras)	Array que contém todas as âncoras no documento.
button	Cria um botão para um formulário. Este objeto pertence ao objeto form (formulário).
checkbox	Uma caixa de verificação de um formulário. Este objeto pertence ao objeto form.
Date	Define a Data/Hora atuais. Este é um objeto de nível superior.
document	Visualiza outros objetos no corpo de uma página. Este objeto pertence ao objeto window.
elements[] (elementos)	Array que apresenta todos os elementos de um formulário. Todos os elementos pertencem ao objeto form.
form (formulário)	Contém qualquer objeto criado em um formulário. Este objeto pertence ao objeto document.
forms[] (formulários)	Array que contém todos os formulários em documento.
frame	Determina as páginas divididas no navegador. Cada frame possui um objeto document. Pertencente ao objeto window.
frames[]	Array de frames do objeto window.

hidden (oculto)	Elemento de formulário que cria uma caixa de texto oculta. Pertencente ao objeto form.
history	Histórico de todas as páginas visitadas. Pertence ao objeto document.
link	Link de hipertexto. Pertence ao objeto document.
links[]	Array dos links de um documento.
location (localização)	URL (endereço) do documento atual. Pertence ao objeto document.
Math	Utilizado na execução de cálculos matemáticos. Objeto de nível superior.
navigator	Informações sobre o browser. Objeto de nível superior.
options[] (opções)	Array de itens de uma lista de seleção (select) em um formulário. Pertencente ao objeto form.
password (senha)	Elemento de um formulário que cria uma caixa de texto do tipo senha. Pertencente ao objeto form.
radio (botão de opção)	Elemento de um formulário que cria cria botões de opção. Pertencente ao objeto form.
reset	Elemento de um formulário que cria um botão que limpa os campos do formulário. Pertencente ao objeto form.
select (lista de opções)	Lista de seleção de um formulário. Pertencente ao objeto form.

string	Variáveis do tipo alfanumérico. Pertencente ao documento que se encontram.
submit	Elemento de um formulário que cria um botão de envio de dados em um formulário. Pertencente ao objeto form.
text (caixa de texto)	Elemento do formulário que cria um campo de texto. Pertencente ao objeto form.
textarea (área de texto)	Elemento do formulário que cria uma área de digitação de texto. Pertencente ao objeto form.
window (janela)	Representa a janela do browser. Objeto de nível superior.

Resumo geral de métodos JavaScript

Métodos do objeto document

Método	Descrição
clear	Limpa a janela (window).
close	Fecha o documento atual.
write	Permite a inclusão de texto no corpo do documento.
writeln	Permite a inclusão de texto no corpo do documento com um caractere de nova linha anexado.

Métodos do objeto form

Método	Descrição
blur()	Quando remove o foco de um campo do tipo text, textarea e campos de senha password.

click()	Quando é dado um clique sobre um elemento de botão em formulário ou campos do tipo radio e checkbox.
focus()	Quando é dado o foco sobre um campo do tipo text, textarea e campos de senha password.
select()	Quando é selecionado o conteúdo de um campo do tipo text, textarea ou password.
submit()	Quando o formulário é enviado ao servidor.

Métodos do objeto date

Método	Descrição
getDate	Retorna o dia do mês da data especificada a partir de um objeto date.
getDay	Retorna o dia da semana da data especificada. O valor retornado é um valor inteiro correspondente ao dia da semana, sendo 0 para Domingo, 1 para Segunda-feira e assim por diante
getFullYear	Retorna o ano composto de 4 dígitos.
getHours	Retorna a hora para a data especificada. O valor retornado corresponde a um número inteiro entre 0 e 23.
getMinutes	Retorna os minutos na hora especificada. O valor retornado corresponde a um número inteiro entre

	0 e 59.
getMonth	Retorna o mês da data especificada. O valor retornado corresponde a um número inteiro entre 0 a 11, sendo 0 para janeiro, 1 para fevereiro e assim por diante.
getSeconds	Retorna os segundos da data especificada. O valor retornado corresponde a um número inteiro entre 0 e 59.
getTime	Retorna o número de segundos entre 1 de janeiro de 1970 e uma data específica.
getTimezoneOffset	Retorna a diferença de fuso horário em minutos para a localidade atual.
getYear	Retorna o ano de uma data específica.
parse	Retorna o número de milessegundos de uma data a partir de 1 de janeiro de 1970, 00:00:00
setDate	Estabelece o dia do mês para uma data especificada sendo um inteiro entre 1 ou 31.
setHours	Estabelece a hora para uma data especificada, sendo um inteiro entre 0 e 23, representando a hora dia.
setMinutes	Estabelece os minutos para a data especificada, sendo um inteiro entre 0 e 59.
setMonth	Estabelece o mês para a data especificada, sendo um inteiro entre 0 e 11 para o mês.

setSeconds	Estabelece o número de segundos para data especificada, sendo um inteiro entre 0 e 59.
setTime	Estabelece o valor do objeto date, sendo um inteiro representando o número de milisegundos desde 1 de janeiro de 1970.
setYear	Define o ano de uma data específica
toGMTString	Converte a data para string usando as convenções da GMT.
toLocaleString	Converte uma data para string, usando as convenções locais.
toString	Retorna uma string representando a data especificada
UTC	Converte uma data delimitada por vírgulas para o número de segundos a partir de 1 de janeiro de 1970.

Métodos do objeto history

Evento	Descrição
Back	Representa a URL visitada anteriormente.
Forward	Representa a URL que estava antes de voltar.
Go	Representa uma URL que esteja na relação de URL's visitadas.

Métodos do objeto math

Métodos	Descrição
Abs	Retorna o valor absoluto de um número.
Acos	Retorna o arco cosseno de um número, em radianos.
Asin	Retorna o arco seno de um número, radianos.
Atan	Retorna o arco tangente de um número, em radianos.
Ceil	Retorna o menor inteiro maior ou igual a um número.
Cos	Retorna o cosseno de um número.
Eval	Calcula o conteúdo de uma expressão. EVAL é uma função.
Exp1	Retorna o logarítmo do múmero na base
Floor	Retorna o maior inteiro menor ou igual ao número.
isNAN	Determina se um valor é um número ou não.
Log	Retorna o logarítmo natural de um número (base E).
Max(num1,num2)	Retorna o maior valor de dois números.
Min(num1,num2)	Retorna o menor valor de dois números.

Pow(base,expoente)	Retorna a base elevada ao expoente.
Random()	Retorna um número aleatório entre 0 e 1.
Round	Retorna o valor arrendondado de um inteiro.
Sin	Retorna o seno de um número.
Sqrt	Retorna a raiz quadrada de um número.
Tan	Retorna a tangente de um número.

Questionário

1. O que é o método link?

_____
_____
_____
_____

2. O que é o objeto math?

_____
_____
_____
_____

3. O que faz o método round do objeto math?

_____
_____
_____
_____

4. O que é Array?

_____
_____
_____
_____

5. O que faz o elemento submit de um formulário?

Aula 7- Revisão / Parte 1

Exercício 1 – Introduzindo o JavaScript na página HTML.

ex_javascript_1.html

```
<html>
<head>
  <title>Ajax – exercício 1 – Introdução do JavaScript na pagina html.</title>
</head>

<script language="javascript">

Alert(“Ola Microcamp”);
</script>

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

Exercício 2 – Arquivo externo.

ex_javasrcipt_2.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<script type="text/javascript" src="exercicio_2.js"></script>
<title>Ajax – exercício 2 - Arquivo JavaScript externo</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>

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

Arquivo javascript externo – exercicio_2.js

```
// ActionScript Document
```

```
alert("Esta é uma mensagem gerada por um arquivo externo de javascript.");
document.write("Este texto foi gerado por um arquivo esterno javascript.");
```

Exercício 3 – Variáveis.

ex_javascript_3.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script language="javascript">
// variáveis globais
var nome, endereco, bairro, cidade, estado;

//Atribuindo valores as variáveis locais.
nome="Marcos de Melo";
endereco="Rua Violeta, 301";
bairro="Centro";
cidade="Campinas";
estado="SP";
cep="13-183.456";

document.write("::::::::::Variáveis Globais::::::::::<br>");
document.write("Nome: " + nome + "<br>");
document.write("Bairro: " + bairro + "<br>");
document.write("cidade: " + cidade + "<br>");
document.write("Estado: " + estado + "<br>");
document.write("CEP: " + cep + "<br><br>");

// variáveis locais, só funcionam dentro da função
function minhaFuncao(){
var nome="Juliana Melo", endereco="Rua Fernando Candido, 50", bairro="Centro",
cidade="Sumare", estado="SP",cep="13-183.456";
document.write("::::::::::Variáveis locais::::::::::<br>");
document.write("Nome: " + nome + "<br>");
document.write("Bairro: " + bairro + "<br>");
document.write("cidade: " + cidade + "<br>");
document.write("Estado: " + estado + "<br>");
document.write("CEP: " + cep + "<br>");
}
```

```
minhaFuncao();

</script>
<title>Ajax - exercício 3 - Variáveis</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" /></head>
<body>
</body>
</html>
```

Exercício 4 – Vetores

ex_javascript_4.html

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

var vetor = new Array(1,100,200,300,"Quatrocentos"); // atribuição de um Array na
varial vetor com 5 valores, as posições começão em 0 sendo elas: 0,1,2,3,4.

document.write(vetor[2]+ "<br>") ;// Escreve o valor da posição 3 do vetor que é o valor
200.

document.write(vetor[4]);// Escreve o valor da posição 5

</script>

<title>Ajax - exercício 4 - Vetores</title>

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

Exercício 5 - JavaScript – Vetores.

ex_javascript_5.html

```
<html>
<head>
<script language="javascript">
// Método join => mostra todos os valores do vetor.
// Método sort => mostra todos os valores do vetor em ordem alfabética.

var vetor = new Array("Marcos de Melo","Nicoly Melo","Pedro Junqueira","Bruna Zapata","Anderson Distro");
document.write("O método join mostrou todos os valores do vetor: " + vetor.join() + "<br>");
document.write("O método sort mostrou os valores do vetor em ordem alfabética: " + vetor.sort());

</script>

<title>Ajax - exercício 4 - Vetores</title>
<body>
</body>
</html>
```

Aual 8 – Revisão Parte 2

Exercício 6 - JavaScript – Operadores.

ex_javascript_6.html

```
<html>
<head>
 <title>Ajax – Javascript - Operadores</title>

<script language="javascript">
```

```
var valor1 = 10;
var valor2 = 200;
var soma = valor1 + valor2 + "<br>"; // se for números será feito calculo.
document.write(soma);

var texto1 = "Marcos";
var texto2 = "de Melo";
var concatena = texto1 + " " + texto2; // se for texto será concatenado os valores.
document.write(concatena);

</script>

</head>
<body>
</body>
</html>
```

Exercício 7 - JavaScript – Operadores.

ex_javascript_7.html

```
<html>
<head>
<title>Ajax - Javascript - Operadores </title>

<script language="javascript">

var valor1 = 5;
var valor2 = 2;
var restoDivisao = valor1 % valor2;
document.write("O resto da divisão de 5 por 2 é: " + restoDivisao);

</script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
</body>
</html>
```

Exercício 8 - JavaScript – Estruturas de controle – If,else.

ex_javascript_8.html

```
<html>
<head>
<title>Ajax - Javascript - Estruturas de controle - IF e ELSE </title>

<script language="javascript">

var codigo=1;
if(codigo==2){ // estrutura condicional que verifica se o código é igual á 2.
alert("O código " + codigo + " esta correto.");
} else{ // se a estrutura for falsa, o código na estrutura else é executado.
        alert("O código " + codigo + "não esta correto.");
}
</script>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
</body>
</html>
```

Exercício 9 - JavaScript – Estruturas de controle – while.

ex_javascript_9.html

```
<html>
<head>
<title>Ajax - Javascript - Estruturas de controle - WHILE </title>
```

```
<script language="javascript">
var codigo=1;

while(codigo<=10) // Enquanto o código for menor ou igual a 10, o código na estrutura
abaixo sera executado
{
        alert("O código é: " + codigo);
        codigo++;
}
</script>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
</body>
</html>
```

Exercício 10 - JavaScript – Estruturas de controle – do... while.

ex_javascript_10.html

```
<html>
<head>
<title>Ajax - Javascript - Estruturas de controle – DO …WHILE </title>
<script language="javascript">
var codigo=1;
do
{
        alert("O código é: " + codigo);
        codigo++;
}
while(codigo<=10) // Enquanto o código for menor ou igual a 10, o código na estrutura
acima sera executada
```

```
</script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
</body>
</html>
```

Aula 9 - Revisão Parte 3

Exercício 11 - JavaScript – Operadores lógicos – for.

ex_javascript_11.html

```
<html>
<head>
<title>Ajax - Javascript - Estrutura de controle - WHILE </title>
<script language="javascript">

document.write("<table border=\"1\">");
for(i=1;i<=10;i++)
{
document.write("<tr><td>");
document.write("Celula " + i );
document.write("</td></tr>");
}
document.write("</table>");
</script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
</body>
</html>
```

Exercício 12 - JavaScript – Funções.

ex_javascript_12.html

```
<html>
<head>
<title>Ajax – JavaScript – Funções </title>

<script language="javascript">
function Quadrado(valor){
        var q = valor*valor;
        return q;
}
document.write("O quadrado de 20 é: " + Quadrado(20));
</script>
</head>
<body>
</body>
</html>
```

Exercício 13 - JavaScript – Funções.

ex_javascript_13.html

```
<html>
<head>
<title>Ajax -</title>
```

```
<script language="javascript">
	function MostraTabela (cor) {
		document.write("<table bgcolor=\"" + cor + "\" border=\"1\">");
		document.write("<tr><td>");
		if(cor=='blue')
			document.write("O fundo desta tabela é azul!");
		else if(cor=='red')
			document.write("O fundo desta tabela é vermelho!");
		document.write("</td></tr>");
		document.write("</table>");
	}
</script>
</head>
<body>
 <p><a href="#" onClick="MostraTabela('blue');">Tabela azul</a></p>
 <p><a href="#" onClick="MostraTabela('red');">Tabela vermelha</a></p>
</body>
</html>
```

Exercício 14 - JavaScript – Funções.

ex_javascript_14.html

```
<html>
<head>
	<title>Ajax –JavaScript – Funções </title>
<script language="javascript">
	function JanelaAviso () {
		janela = window.open('', 'Cadastro',
'width=200,height=200,toolbar=no,location=no,status=no,scrollbars=no,resizable=no');
		janela.document.write("<p>Para acessar o site você precisa se
cadastrar!</p>");
		janela.document.write("<p><a
href='javascript:window.close();'>Fechar</a></p>");
	}
</script>
</head>
```

```
<body>
 <p><a href="#" onClick="JanelaAviso();">Acessar o site</a></p>
</body>
</html>
```

Exercício 15 - JavaScript – Funções.

ex_javascript_15.html

```
<html>
<head>
	<title>Ajax -</title>
<script language="javascript">
	function Incrementa () {
		document.formulario.numero.value++;
	}
</script>
</head>
<body>
 <form name="formulario">
	<input type="text" name="numero" value="1">
	<input type="button" value="Incrementar" onClick="Incrementa();">
 </form>
</body>
</html>
```

Exercício 16 - JavaScript – Funções.

ex_javascript_16.html

```
<html>
<head>
```

```
        <title>Ajax -</title>
<script language="javascript">
        function Verificar () {
                if(document.formulario.aceitar.checked==false)
                        alert("Você deve aceitar os termos do contrato!");
                else
                        alert("Concluído!");
        }
</script>
</head>
<body>
 <form name="formulario">
        Aqui vai o texto do contrato...<br>
        <input type="checkbox" name="aceitar" value="sim"> Aceito o contrato<br>
        <input type="button" value="Prosseguir" onClick="Verificar();">
 </form>
</body>
</html>
```