Workshop de Javascript

Autores do manual

Este manual foi criado pelos seguintes colaboradores de Criarweb.com:

Miguel Angel Alvarez - Tradução de JML
(54 capítulos)

Carlos Cuenca Díaz
Consultoria Informática
(5 capítulos)

Ignacio Rodriguez
(1 capítulo)

Fabio Núñez Iturriaga
http://www.nedial.net
(2 capítulos)

Juan Carlos Gámez
http://www.espaciolatino.com
(1 capítulo)

Bruno Suárez Laffargue
http://www.helloworldsolutions.com/
(1 capítulo)

Juan Carlos Montejo
http://www.adrformacion.com/
(1 capítulo)

Agustin Jareño
(1 capítulo)

João Coelho
(1 capítulo)

César Pietri
http://www.tips-web.com
(1 capítulo)

José Antonio Jiménez Garelli
(2 capítulos)

Javier Bernal Lérida
(1 capítulo)

Alex Sancho
http://alexsancho.name/
(1 capítulo)

Ismael Zori
http://www.telefonica.net/web2/izori/
(1 capítulo)

Gema Maria Molina Prados
(1 capítulo)

Efeitos rápidos com Javascript

Antes de aprofundarmos na matéria, podemos ver uma série de efeitos rápidos que se podem programar com Javascript. Isto, pode nos dar uma idéia más clara e exata das capacidades e da potência da linguagem na hora de percorrer os próximos capítulos.

Abrir uma janela secundária

Primeiro vamos ver que com uma linha de Javascript podemos fazer coisas bastante atrativas. Por exemplo podemos ver como abrir uma janela secundária sem barras de menus que mostre o buscador Google. O código seria o seguinte:

```
<script>
window.open("http://www.google.com","","width=550,height=420,menubar=no")
</script>
```

Podemos ver o exemplo em funcionamento aqui.

Uma mensagem de boas vindas

Podemos mostrar uma caixa de texto emergente ao terminar de carregar o portal de nosso site web, que poderia dar as boas vindas aos visitantes.

```
<script>
window.alert("Bem-vindo ao meu site web. Obrigado...")
</script>
```

Podemos ver o exemplo em uma página a parte.

Data atual

Vejamos agora um simples script para mostrar a data de hoje. Às vezes é muito interessante mostrá-la nas webs para dar um efeito de que a página está "ao dia", ou seja, está atualizada.

```
<script> document.write(new Date()) </script>
```

Estas linhas deveriam ser introduzidas dentro do corpo da página no lugar onde quisermos que apareça a data da última atualização. Podemos ver o exemplo em funcionamento aqui.

> **Nota:** Um detalhe a destacar é que a data aparece em um formato um pouco raro, indicando também a hora e outros atributos da mesma, mas já aprenderemos a obter exatamente o que desejarmos no formato correto.

Botão de voltar

Outro exemplo rápido pode ser visto a seguir. Trata-se de um botão para voltar para trás, como o que temos na barra de ferramentas do navegador. Agora veremos uma linha de código que mistura HTML e Javascript para criar este botão que mostra a página anterior no histórico, se é que havia.

```
<input type=button value=Atrás onclick="history.go(-1)">
```

O botão será parecido ao seguinte. Podemos clicá-lo para ver seu funcionamento (deveria nos levar à página anterior).

Atrás

Como diferença com os exemplos anteriores, há que destacar que neste caso a instrução Javascript se encontra dentro de um atributo de HTML, onclick, que indica que essa instrução tem de ser executada como resposta ao clicar no botão.

É possível comprovar a facilidade com a qual se podem realizar algumas ações interessantes, e existiriam muitas outras mostras, mas que reservamos para capítulos posteriores.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Abertura e configuração de popups com Javascript

Clique aqui para ver o que é uma janela secundária

Em determinadas ocasiões é muito útil abrir um link em uma janela secundária, ou seja, uma janela a parte que se abre para mostrar uma informação específica. Algumas vantagens de abrir um link em uma janela secundária seriam:

- O usuário não sai da página onde estava o link.
- A janela secundária pode ser configurada livremente com o qual podem ser feitas janela maiores ou menores e com mais ou menos menus.
- Em geral, o grau de controle da janela secundária utilizando Javascript aumenta.

Para abrir uma janela secundária podemos fazer de duas maneiras, com HTML e com Javascript. Vejamos cada uma delas:

Abrir uma janela com HTML

Pode-se conseguir abrir uma janela secundária muito facilmente simplesmente com HTML. Para isso podemos utilizar o atributo TARGET das etiquetas HREF. Se colocamos target="_blank" no link, a página se abrirá numa janela secundária. Também podemos colocar target="xxx" para que o link se apresente na janela chamada xxx ou no frame xxx.

O link teria que ter esta forma:

```
<a href="minhapagina.html" target="_blank">
```

O problema de abrir uma página secundária com HTML consiste em que não podemos definir a forma desta e nem poderemos exercer maior controle sobre ela tal como comentávamos entre as vantagens de abrir uma janela secundária com Javascript. A janela que se abre sempre será como o usuário tenha definido por padrão em seu navegador.

Abrir uma janela com Javascript

Para abrir uma janela com Javascript podemos utilizar a sentença window.open(). Não tem problema se você não conhece Javascript, visto que é muito simples utiliza-lo para este caso. Veremos passo a passo como abrir uma janela secundária utilizando Javascript.

1. Sentença Javascript para abrir uma janela

A sentença é simplesmente a função window.open(), o mais complicado é saber como utilizar esta função, mas agora veremos que não requer nenhuma complicação.

A função window.open() recebe três parâmetros, que se colocam dentro dos parênteses, deste modo:

window.open(URL,nome_da_janela,forma_da_janela)

Vejamos rapidamente cada um destes parâmetros separadamente.

URL: representa o URL que desejamos abrir na janela secundária, por exemplo http://www.criarweb.com
nome_da_janela: é o nome que se atribui a esta janela para dirigir links com o atributo target do HTML
forma_da_janela: se indica o aspecto que vai ter a janela secundáaria. Por exemplo, pode-se definir sua altura, largura, se têm barras de deslocamento, etc

Vejamos um exemplo de sentença Javascript completa para abrir uma janela secundária:

```
window.open("http://www.criarweb.com" , "janela1" , "width=120,height=300,scrollbars=NO")
```

Isto quer dizer que abra a página inicial de criarweb.com em uma janela secundária a qual vamos chamar janela1. Ademais, a janela será de 120 pixels de largura, 300 de altura e não terá barras de deslocamento.

Um esclarecimento adicional, se depois de abrir essa janela colocarmos outro link na página que abria a janela cujo atributo target está dirigido para o nome_da_janela (neste caso janela1), este link será mostrado na janela secundária.

2. Função que abre uma janela

Os mais cômodo para abrir uma janela é colocar uma função Javascript que se encarregue das tarefas de abri-la e que receba por parâmetro a URL que se deseja abrir.

O script é simples, vejamos a seguir:

```
<script language=javascript>
function janelaSecundaria (URL){
  window.open(URL,"janela1","width=120,height=300,scrollbars=NO")
}
</script>
```

3. Colocamos um link

Este link não deve estar dirigido diretamente à página que quisermos abrir, e sim, à sentença Javascript necessária para abrir a janela secundária. Para executar uma sentença Javascript com o clique de um link, fazemos assim:

```
<a href="javascript:sentenca_javascript_para_abrir_a_janela">
```

4. O link chama à função que abre a janela

Agora Vejamos como ficaria todo esse link na página.

```
<a href="javascript:janelaSecundaria('http://www.criarweb.com')"> Clique neste link para abrir a janela secundaria</a>
```

Que dá como resulado:

Clique neste link para abrir a janela secundária

(Na página que formos colocar este link deveríamos ter o script que fizemos anteriormente que continha a função para abrir a janela.)

Há que observar que as aspas simples são as que são colocadas para definir o URL que se passa como parâmetro da função janelaSecundaria(). São aspas simples porque o href do link já tem umas aspas duplas, e dentro das aspas duplas sempre se deve utilizar aspas simples a não ser que desejemos fechar as aspas duplas.

Parâmetros para dar forma a uma janela

Estes atributos podem ser utilizados na função window.open() para definir a forma que desejar que tenha sua janela secundária.

Width	Ajusta a largura da janela. Em pixels
Height	Ajusta a altura da janela
Top	Indica a posição da janela. Na verdade é a distancia em pixels que existe entre a borda superior da tela e a borda superior da janela.
Left	Indica a posição da janela. Em concreto é a distancia em pixels que existe entre a borda esquerda da tela e a borda da janela.
Scrollbars	Para definir de forma exata se saem ou não as barras de deslocamento. scrollbars=NO fazem com que nunca saiam. Scrollbars=YES faz com que sempre saiam (sempre em ie e somente se forem necessárias em NTS).
Resizable	Establece se se pode ou não modificar o tamanho da janela. Com resizable=YES pode-se modificar o tamanho e com resizable=NO consegue-se um tamanho fixo.
Directories (barra diretorios)	A partir de aqui se enumeram outra série de propriedades que serven para mostrar ou não um elemento da barra de navegação que tem os navegadores mais populares, como poderia ser a barra de menus ou a barra de estado.
Location (barra endereços)	Quando colocamos o atributo=YES estamos forçando que esse elemento seja visto. Quando colocamos atributo=NO o que fazemos é evitar que esse elemento seja visto.
Menubar	

(barra de menus)

Status
(barra de estado)

Titlebar
(a barra do título)

Toolbar
(barra de ferramentas)

Abrir uma janela sem um link

Em outras ocasiões desejaremos abrir uma janela secundária automaticamente, ou seja, sem a necessidade de que o usuário clique sobre nenhum link. Neste caso, o código da função janelaSecundaria nos serve também e teremos que acrescentar uma linha de código Javascript em seguida da função janelaSecundaria. Esta linha a acrescentar simplesmente será uma chamada à função que se executará Segundo esteja carregando a página. Vejamos como ficaria este código:

```
<script language=javascript>
function janelaSecundaria (URL){
  window.open(URL,"janela1","width=120,height=300,scrollbars=NO")
}

janelaSecundaria("http://www.criarweb.com");
</script>
```

Fica em negrito o que seria a chamada à função que abre a janela secundária, como está fora de uma função se executa segundo estiver carregando a página.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Acessso por senha Javascript

Lamentavelmente, javascript não é uma linguagem com a qual se possa realizar um método interessante para fazer com que algumas páginas de nosso site somente seja acessíveis se se introduz uma senha correta. Mesmo assim, existe um mecanismo para poder realizar isto, que não é muito avançado nem muito seguro, mas pode dar um efeito em nossas páginas que estamos desejando.

Trata-se de colocar páginas web em nosso espaço de links, para que ninguém possa acessa-las. Esta é toda a segurança que podemos dar a nossas páginas: como não existem links dirigidos para elas, ninguém poderá acessa-las. A única maneira de acessar as páginas seria conhecendo o nome de arquivo e escrever a URL do mesmo, mas como também não vamos publicar o nome do arquivo, poderemos estar quase certos de que ninguém acertará construir

o endereço da página que queremos ocultar. Logo, criaremos um formulário muito simples, que incluirá um campo de texto e um botão. No campo de texto teremos que escrever o nome do arquivo que se deseja ver e ao clicar o botão javascript seremos conduzidos para a página que tiver esse nome de arquivo. Nesse ponto pode acontecer duas coisas:

1. Primeiro, que o nome de arquivo seja incorreto, ou seja, inventamos a senha mas não acertamos com o nome da página escondida. Neste caso, se mostraria uma página de erro típica, dessas que o servidor mostra quando tentamos acessar a uma página que não existe.
2. Segundo, que o nome da página seja correto, ou seja, que a senha que introduzimos seja igual ao nome do arquivo queremos acessar. Neste caso, javascript nos conduzirá ao lugar correto e poderemos ver a página oculta.

Vejamos passo a passo como construir este sistema de acesso por senha:

1.- As páginas para desenvolver

Temos que trabalhar com 2 páginas web pelo menos, uma para colocar o formulário e outra que seria a página oculta. Teremos estas páginas colocadas no mesmo diretório, com o qual simplificaremos um pouco o problema.

2.- Formulário para a senha

Na página que quisermos colocar o acesso por senha, devemos colocar o seguinte formulário:

```
<FORM name=formsenha>
<INPUT type=password name=senha>
<INPUT type=button value=Acessar>
</FORM>
```

3.- Função que nos envia à página oculta

Como a página oculta tem como nome de arquivo o que tivermos escrito no campo de texto, poderemos acessa-la da seguinte forma:

```
<SCRIPT>
function acesso(){
   window.location = document.formsenha.senha.value + ".html"
}
</SCRIPT>
```

A função é muito simples, o único que faz é concatenar o nome que foi escrito no campo de texto com ".html" e nos manda, utilizando window.location, diretamente à página cujo nome acaba de ser construido.

Como concatenamos com ".html" o nome do arquivo escrito no formulário, o nome que escrevermos deverá ir sem ".html".

4.- Incluir no botão a chamada à função

Com o objetivo de que ao clicar o botão o navegador nos leve à página oculta, temos que fazer com que ao clica-lo, seja chamada a função acesso definida no ponto anterior. Isto se

consegue mediante o atributo onclick, que devemos inserir na etiqueta do botão.

```
<INPUT type=button value=Acessar onclick="acesso()">
```

5.- Código inteiro da página

Podemos ver a seguir o código da página inteira. Somente mostramos o código da página que tem o formulário, porque a página oculta poderá ser como cada um desejar.

```
<html>
<head>
   <title>senha acesso</title>
</head>

<body>
<SCRIPT>
function acesso(){
   window.location = document.formsenha.senha.value + ".html"
}
</SCRIPT>

<FORM name=formsenha>
<INPUT type=password name=senha>
<INPUT type=button value=Acessar onclick="acesso()">
</FORM>

</body>
</html>
```

6.- Últimos anotações

Uma coisa importante na hora de conseguir que o script seja mais confiável consiste em criar páginas com um nome de arquivo difíicil de se inventar. Como o nome da página é a senha com a qual vai ser acessada a página necessitaremos que tal nome seja um pouco complexo, como por exemplo, fks12dmxc53.html. Se a página senha se chamasse por exemplo, index.html qualquer um com um pouco de imaginação poderia inventa-la.

Antes de terminar, cabe repetir que este não é o método mais seguro que existe para criar scripts para realizar acessos restringidos, é somente uma pequena astúcia que "funciona". Para realizar este objetivo com melhores resultados temos linguagens como ASP, PHP ou CGI. Também podemos restringir o acesso às páginas utilizando o própio sistema operativo e a autentificação que este implementa, tal vez seja a opção mais cômoda, embora não seja cem porcento provável que o nosso provedor de hospedagem nos permita.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Rollover com Javascript

Fazer com que mude uma imagem ao passar o mouse por cima, como convidando a clicar, chamamos de rollover. É um dos efeitos mais vistosos que podemos incluir em uma página web con poucas linhas de Javascript, e sem necessidade de saber programar.

Exemplo de rollover

Vamos ver a técnica pela **prática e com uma elemental receita:**

1. Nomeamos a imagem

Colocamos na página a imagem que tem o desenho apagado. Além disso, lhe atribuímos um nome, para poder nos referirmos a ela mediante JavaScript.

```
<img src="desenho_apagado.gif" name="imagen1">
```

2. Colocamos um link na imagem

Agora colocamos o link cujo queremos navegar no momento no qual o usuário clique nele.

```
<a href="ir_a.html">
```

3. Começamos com JavaScript

Devemos colocar dois atributos HTML ao link que vai nos servir para realizar o efeito buscado. **OnMouseOver**, com ele indicaremos, mediante JavaScirpt, a ação a realizar quando pousarmos o mouse em cima da imagem.
OnMouseOut nos serve para definir o evento de retirar o mouse da imagem,

```
<a href="ir_a.html" onmouseover="-Código JavaScript-" onmouseout="-Código JavaScript-">
```

4. Pegamos as imagens com JavaScript

Vamos declarar duas variáveis com JavaScript para salvar as imagens iluminada e apagada. Para isso, vamos utilizar a etiqueta <SCRIPT> de HTML. Podemos colocar o script em qualquer lugar, mas seria mais adequado colocar antes da imagem.
Os números que vocês verão correspondem com a largura e com a altura da imagem que estão pegando.

```
<script language=javascript>
iluminada = new Image(84,34)
iluminada.src = "desenho_iluminado.gif"
apagada = new Image(84,34)
apagada.src = "desenho_apagado.gif"
</script>
```

5. Escrevemos o código dos eventos

Para acessar um elemento de JavaScript utilizamos a hierarquia de objetos de JavaScript. Pode ser complicada, mas nosso objetivo é simples, assim faremos:

```
window.document['nome_da_imagem'].src = variavel_imagem_javascript.src
```

Teoricamente, os atributos HTML dos eventos **onmouseover** e **onmouseout** ficarão assim:

onmouseover="window.document['imagem1'].src= iluminada.src

onmouseout="window.document['imagem1'].src = apagada.src

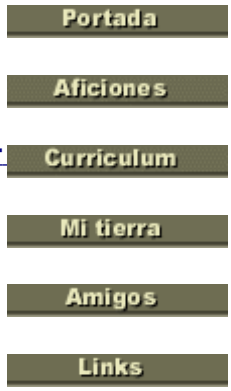
5. Este é o código completo

```
<script language=javascript>
    iluminada = new Image(84,34)
   iluminada.src = "desenhoiluminado.gif"
   apagada = new Image(84,34)
   apagada.src = "desenhoapagado.gif"
</script>
<a href="ir_a.html"
onmouseover="window.document['imagem1'].src = iluminada.src"
onmouseout="window.document['imagem1'].src = apagada.src">
<img src="images/eugimlogo.gif" border=0 name="imagem1" width=84 height=34>
</a>
```

Isto é o único que vocês deve ser feito para iluminar uma imagem, é um primeiro passo, agora vocês podem provar com um grupo de imagens para criar uma barra de navegação dinâmica com Javascript!

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Navegador dinâmico Javascript



Vamos ver como fazer, com Javascript e umas imagens, **uma barra de navegação para uma página web**, que mude quando o mouse passe por cima.

Esta ajuda técnica está pensada para ser lida em seguida do informe **Rollover com Javascript**, publicado em CriarWeb, pois contém as bases sobre o que vamos trabalhar agora.

1. Construimos as imagens

Temos que construir duas versões de barra de navegação, umas que esteja iluminada, por assim dizer, e outra que esteja um pouco apagada. Ao passar o mouse mudaremos de uma a outra.

Aqui estão as imagens que propomos para este exercício:

2. Criamos a barra com HTML

Com uma tabela de HTML vamos fazer a barra de navegação para a página, ainda sem movimento. Cm estes detalhes:

- À princípio colocamos as imagens apagadas
- Cada imagem tem um link à página correspondente
- Demos um nome diferente a cada imagem com o atributo **name**. desde imagem1 até a imagem6.
- Nossa tabela tem cellpadding e cellspacing 0 para que não fique separação entre as imagens. Por esta última razão, também não devemos deixar espaço em branco no código HTML entre os TD e as imagens.

```
<table cellspacing="0" cellpadding="0" border="0">
<tr>
   <td><a href="portada.html"><img src="portada1.gif" width=110 height=16 alt="Portada" border="0"
name="imagem1"></a></td>
</tr>
<tr>
   <td><a href="aficciones.html"><img src="aficciones1.gif" width=110 height=16 alt="Aficciones" border="0"
name="imagem2"></a></td>
</tr>
<tr>
   <td><a href="curriculum.html"><img src="curriculum1.gif" width=110 height=16 alt="Curriculum" border="0"
name="imagem3"></a></td>
</tr>
<tr>
   <td><a href="mitierra.html"><img src="mitierra1.gif" width=110 height=16 alt="Mi tierra" border="0"
name="imagem4"></a></td>
</tr>
<tr>
   <td><a href="amigos.html"><img src="amigos1.gif" width=110 height=16 alt="Amigos" border="0"
name="imagem5"></a></td>
</tr>
<tr>
   <td><a href="links.html"><img src="links1.gif" width=110 height=16 alt="Links" border="0"
name="imagem6"></a></td>
</tr>
</table>
```

3. Pré-carregamos as imagens

Para ter as imagens já em nosso explorador antes de que sejam utilizadas, devemos pré-carregar usando Javascript, assim conseguiremos que o efeito de rollover seja rápido, e mudem as imagens velozmente segundo se passe o mouse.

Utilizaremos este código, que se coloca no cabeçalho do documento HTML:

```
<script>


  iluminada1 = new Image(110,16)
 iluminada1.src = "portada2.gif"
 apagada1 = new Image(110,16)
 apagada1.src = "portada1.gif"

 iluminada2 = new Image(110,16)
 iluminada2.src = "aficciones2.gif"
 apagada2 = new Image(110,16)
 apagada2.src = "aficciones1.gif"

 iluminada3 = new Image(110,16)
 iluminada3.src = "curriculum2.gif"
 apagada3 = new Image(110,16)
 apagada3.src = "curriculum1.gif"

 iluminada4 = new Image(110,16)
 iluminada4.src = "mitierra2.gif"
 apagada4 = new Image(110,16)
 apagada4.src = "mitierra1.gif"

 iluminada5 = new Image(110,16)
 iluminada5.src = "amigos2.gif"
 apagada5 = new Image(110,16)
 apagada5.src = "amigos1.gif"

 iluminada6 = new Image(110,16)
 iluminada6.src = "links2.gif"
 apagada6 = new Image(110,16)
 apagada6.src = "links1.gif"

</script>
```

4. Os eventos

Temos que definir os eventos **onmouseover** e **onmouseout** para cada um dos links, indicando a mudança da imagem à iluminada e à apagada respectivamente.

```
onmouseover="window.document['imagem1'].src =iluminada1.src"
 onmouseout="window.document['imagem1'].src = apagada1.src"

onmouseover="window.document['imagem2'].src =iluminada2.src"
 onmouseout="window.document['imagem2'].src = apagada2.src"

onmouseover="window.document['imagem3'].src =iluminada3.src"
 onmouseout="window.document['imagem3'].src = apagada3.src"

onmouseover="window.document['imagem4'].src =iluminada4.src"
 onmouseout="window.document['imagem4'].src = apagada4.src"

onmouseover="window.document['imagem5'].src =iluminada5.src"
 onmouseout="window.document['imagem5'].src = apagada5.src"

onmouseover="window.document['imagem6'].src =iluminada6.src"
```

```
onmouseout="window.document['imagem6'].src = apagada6.src"
```

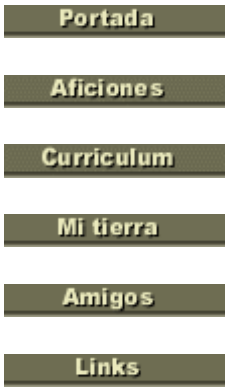
5. Código inteiro

Isso é tudo que se necessita. A seguir podemos ver o código inteiro deste exemplo:

```
<html>
<head>
 <title>Navegador</title>
<script>


   iluminada1 = new Image(110,16)
  iluminada1.src = "portada2.gif"
  apagada1 = new Image(110,16)
  apagada1.src = "portada1.gif"

  iluminada2 = new Image(110,16)
  iluminada2.src = "aficciones2.gif"
  apagada2 = new Image(110,16)
  apagada2.src = "aficciones1.gif"

  iluminada3 = new Image(110,16)
  iluminada3.src = "curriculum2.gif"
  apagada3 = new Image(110,16)
  apagada3.src = "curriculum1.gif"

  iluminada4 = new Image(110,16)
  iluminada4.src = "mitierra2.gif"
  apagada4 = new Image(110,16)
  apagada4.src = "mitierra1.gif"

  iluminada5 = new Image(110,16)
  iluminada5.src = "amigos2.gif"
  apagada5 = new Image(110,16)
  apagada5.src = "amigos1.gif"

  iluminada6 = new Image(110,16)
  iluminada6.src = "links2.gif"
  apagada6 = new Image(110,16)
  apagada6.src = "links1.gif"

</script>
</head>


<body bgcolor="#6e6d52">
<table cellspacing="0" cellpadding="0" border="0">
<tr>
   <td><a href="portada.html" onmouseover="window.document['imagem1'].src =iluminada1.src"
onmouseout="window.document['imagem1'].src = apagada1.src"><img src="portada1.gif" width=110 height=16
alt="Portada" border="0" name="imagen1"></a></td>
</tr>
<tr>
   <td><a href="aficciones.html" onmouseover="window.document['imagem2'].src =iluminada2.src"
onmouseout="window.document['imagem2'].src = apagada2.src"><img src="aficciones1.gif" width=110
height=16 alt="Aficciones" border="0" name="imagen2"></a></td>
</tr>
<tr>
   <td><a href="curriculum.html" onmouseover="window.document['imagem3'].src =iluminada3.src"
onmouseout="window.document['imagem3'].src = apagada3.src"><img src="curriculum1.gif" width=110
height=16 alt="Curriculum" border="0" name="imagen3"></a></td>
</tr>
<tr>
   <td><a href="mitierra.html" onmouseover="window.document['imagem4'].src =iluminada4.src"
onmouseout="window.document['imagem4'].src = apagada4.src"><img src="mitierra1.gif" width=110 height=16
alt="Mi tierra" border="0" name="imagen4"></a></td>
</tr>
<tr>
   <td><a href="amigos.html" onmouseover="window.document['imagem5'].src =iluminada5.src"
onmouseout="window.document['imagem5'].src = apagada5.src"><img src="amigos1.gif" width=110 height=16
```

```
alt="Amigos" border="0" name="imagen5"></a></td>
</tr>
<tr>
   <td><a href="links.html" onmouseover="window.document['imagem6'].src =iluminada6.src"
onmouseout="window.document['imagem6'].src = apagada6.src"><img src="links1.gif" width=110 height=16
alt="Links" border="0" name="imagen6"></a></td>
</tr>
</table>


</body>
</html>
```

Exemplo funcionando



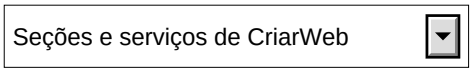
Evidentemente, existem muitas outras maneiras de fazer uma barra de navegação, mas esta é uma boa forma. Com um pouco de criatividade você pode criar umas imagens bonitas que façam uns efeitos legais ao passar o mouse por cima.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Navegador desdobrável

Está na moda incluir um navegador de website consistente em um campo desdobrável de formulário ou de caixa de seleção que, ao selecionar um de seus elementos, que se correspondem com seções do site, nos leve à seção que pretendemos visitar de maneira automática.

Pode ser visto a seguir um exemplo.



Este tipo de controle possui várias vantagens, entre as quais podemos destacar:

- Ocupa pouco espaço na página
- Pode-se colocar quantas opções se desejar

- É uma ferramenta muito visual e prática
- É muito fácil de implementar

Passos a realizar:

Para conseguir um navegador assim em nossa página web devemos enfrentar uma tarefa que poderia ser dividida em duas partes. Por um lado, devemos criar um formulário que contenha a caixa de seleção desdobrável e por outro, um simples script que dê vida à caixa, ou seja, que prepare a caixa para que o navegador se dirija à página selecionada. Vamos ver separadamente cada um destes elementos.

Formulário

Custará da etiqueta <form> para abrir e fechar o formulário. Daremos um nome ("navegador") ao formulário para poder acessa-lo mais tarde usando Javascript.

Dentro do formulário colocaremos um único elemento: o campo de seleção múltipla desdobrável. Daremos um nome também ao campo ("secoes") para poder acessá-lo usando Javascript. Este campo conterá as diferentes opções que queremos que sejam apresentadas no menu desdobrável. Estas poderão ser seções de sua web ou também, páginas que estejam fora do site.

Vamos ver o código do formulário antes de continuar com a explicação:

```
<form name=navegador>
<select name="secoes">
<option value="no">Seções e serviços de CriarWeb
<option value="no">-------------------------------------
<option value="http://www.criarweb.com/">Portal
<option value="http://www.criarweb.com/manuais.asp">Manuais
<option value="http://www.criarweb.com/divulgacao">Divulgação de p&aacute;ginas
<option value="http://www.criarweb.com/contribue/">Colaborar
<option value="no">-------------------------------------
<option value="http://www.criarweb.com/contato/">Entrar em contato
<option value="http://www.criarweb.com/livro_visitas/">Livro de visitas
</select>
</form>
```

Se você observar, cada opção corresponde a uma das seções ou serviços de criarweb e está composta por duas partes importantes, a primeira corresponde com o atributo **value** da etiqueta <option>, que é igual à URL absoluta da página ao qual queremos nos dirigir. A segunda parte destacável corresponde com o nome que desejarmos que seja visto na caixa de seleção.

É também importante destacar que incluímos alguma opção que não quisemos que nos direcione a nenhum lugar. São opções para separar as seções dos serviços ou a primeira opção, para indicar que este menu desdobrável contém as seções e serviços de criarweb. As opções que não quisermos que nos levem a outra página são marcadas com um "no" em seu atributo value.

O script

Agora vamos ver qual é o script que utilizamos para animar este menu desdobrável. É muito simples, podemos vê-lo a seguir:

```
<script language=javascript>

function destino(){
```

```
url = document.navegador.secoes.options[document.navegador.secoes.selectedIndex].value
if (url != " no") window.location = url;
}
</script>
```

Basicamente é uma função que recupera o *value* da opção selecionada na caixa de seleção desdobrável e o armazena em uma variável chamada url. Posteriormente, se a variável url não contém a palavra "no", leva ao navegador à posição selecionada, ou seja, ao url que havíamos buscado. Lembrem que se marcávamos o value de uma opção a "no" é que não desejávamos que o navegador viajasse a outro endereço.

O evento OnChange

Isto não funcionaria se não vinculássemos o evento onchange da caixa de seleção à função destino que vimos agora pouco. O evento onchange se dispara quando se muda o valor selecionado dentro do menu desdobrável e devemos fazer com que chame a função destino. Para isso, na etiqueta <select> devemos incluir o seguinte atributo onchange="destino()". A etiqueta ficaria assim:

```
<select name="secoes" onchange="destino()">
```

Com tudo isto junto você já terá um bonito navegador desdobrável e, esperamos, a capacidade para personalizá-lo a seu gosto.

Mais uma coisa: se o seu site tem frames deverá ser feito algumas mudanças no script para que tudo funcione corretamente.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Navegador desdobrável com frames

Este é um artigo que debe ser lido como continuação do artigo Como fazer um navegador desdobrável, publicado em criarweb.com. Nesse artigo mostrávamos como criar um navegador desdobrável com um elemento SELECT de um formulário.

Muitos visitantes já utilizaram o script com sucesso, mas alguns escreveram com dúvidas de sua utilização em uma página realizada com frames. A dúvida consiste em que o navegador somente nos atualiza o frame no qual está, e o interessante para eles seria que atualizasse um frame diferente. É um problema muito lógico visto que muitas vezes o navegador se coloca de modo que esteja sempre visível, em um frame onde temos os controles de navegação e a área que desejarmos que se atualize é a correspondente ao frame principal.

Mudanças no script

O único lugar onde vamos ter que fazer mudanças é no script que contem a função a qual chamamos destino(). Há que adaptar essa função para que possamos mudar a página de um frame distinto ao que estamos.

Em nosso exemplo anterior fazíamos window.location = url para mudar o conteúdo do frame

onde estava o navegador. Agora devemos mudar o window.location de um frame distinto a este e para acessar a location de um frame distinto se consegue através desta inter-ligações de objetos:

```
window.parent.frames[].window.location
```

frames[] é um vetor de frames onde o primeiro frame do FRAMESET seria frames[0], o segundo seria frames[1] e assim sucessivamente. Caso não tenha ficado claro, vejamos com um exemplo.

Temos este FRAMESET

```
<frameset rows="*,40">
<frame name="principal" src="index.html" marginwidth="10" marginheight="10" scrolling="auto"
frameborder="no">
<frame name="menudesdobravel" src="desdob.html" marginwidth="10" marginheight="10" scrolling="auto"
frameborder="no">
</frameset>
```

No segundo frame temos a página que contem menu desdobrável. Como é o segundo frame acessaríamos a sua location desta maneira:

```
window.parent.frames[1].window.location = url
```

O script inteiro ficaria assim:

```
<script language=javascript>
function destino(){
    url = document.navegador.secoes.options[document.navegador.secoes.selectedIndex].value
    if (url != "no") window.parent.frames[0].window.location = url;
}
</script>
```

Isso é tudo, já não faz falta mudar mais coisas para cumprir nossos objetivos.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Texto em movimento na barra de estado

Vamos ver neste artigo como fazer para que se mova um texto pela barra de estado de nosso navegador. É um script bastante simples e corrente. Sem dúvida, pode ser fácil pegar o script, copiar em nossa página e fazê-lo funcionar, mas nós vamos procurar uma explicação para que tudo fique mais claro e entendamos um pouco o que estamos fazendo.

Este script vai criar um texto que se moverá da direita à esquerda pela barra de estado. Pode-se ver na **página de exemplo**. Agora vejamos os distintos passos.

1. Defino umas variáveis iniciais

O script pode mover o texto que nós desejarmos e para configurá-lo há que criar uma variável que chamamos texto_estado onde introduziremos nosso texto.

```
var texto_estado = "          Bem-vindo a minha página web"
```

Podemos notar que foram introduzidos uns espaços antes que o texto. São para que se crie um espaço na barra de estado entre a saída do texto e a entrada deste pelo outro lado. O número de espaços em branco pode ser modificado livremente, assim como o texto que se mostra.

Também será necessário criar uma variável chamada posição onde vamos salvar a posiç/ao do texto na barra de estado.

```
var posicao = 0
```

2. Função para mover o texto

Agora vamos ver a função, a qual chamaremos move_texto(), que se encarrega de mover o texto pela barra de estado. Entender esta função pode ser um pouco complexo se não se conhece um pouco a linguagem Javascript. De qualquer forma, podemos também copia-la e cola-la em nossas páginas embora não consigamos entender. Realiza quatro ações básicas:

- Move a posição atual, atualizando a variável posição. Se chegarmos ao final da cadeia, volta-se ao princípio
  ```
  if (posicao < texto_estado.length)
     posicao ++;
  else
     posicao = 1;
  ```
- Cria uma cadeia a partir do texto original. A cadeia criada contém o texto que existe desde a posição atual até o final concatenado com o que há desde o princípio até a posição atual. Este é o passo que realmente gera o movimento, porque vai mudando a cadeia que logo escreveremos à medida que a posição também muda.
  ```
  string_atual = texto_estado.substring(posicao) + texto_estado.substring(0,posicao)
  ```
- Escreve a cadeia resultante da operação anterior na barra de estado.
  ```
  window.status = string_atual
  ```
- A função chama-se a si mesma para continuar o movimento. Para isso, utiliza-se uma função muito socorrida, setTimeout(), que serve para executar uma sentença com atraso de tempo. A função recebe a sentença para executar (que neste caso, é uma chamada a mesma função) e o número de milésimos de segundos que tem que esperar para executa-la, neste caso 50.
  ```
  setTimeout("move_texto()",50)
  ```

A função inteira tem este código:

```
function move_texto(){
   if (posicao < texto_estado.length)
      posicao ++;
   else
      posicao = 1;
   string_actual = texto_estado.substring(posicao) + texto_estado.substring(0,posicao)
   window.status = string_atual
   setTimeout("move_texto()",50)
}
```

3. Chamamos à função

Para começar a mover o texto pela página temos que realizar uma chamada à função que se encarrega disso. Será mais claro o código da página se colocarmos a chamada à função depois que tiver sido definida, embora não seja obrigatório.

```
move_texto()
```

4. Tudo junto

Para acabar, podemos observar a seguir o código inteiro de uma página web que move texto pela sua barra de estado. É um página bastante simples depois de tudo.

```
<html>
<head>
  <title>Texto ena barra de estado</title>
  <script language="javascript">
  //texto da mensagem
  var texto_estado = "        Bem-vindos a minha página web"
  var posicao = 0

   //funcao para mover o texto da barra de estado
  function move_texto(){
    if (posicao < texto_estado.length)
      posicao ++;
    else
      posicao = 1;
    string_atual = texto_estado.substring(posicao) + texto_estado.substring(0,posicao)
    window.status = string_atual
    setTimeout("move_texto()",50)
  }
  move_texto()
  </script>
</head>

<body>
<h1>Exemplo de script com um texto na barra de estado</h1>

</body>
</html>
```

Pode-se **ver o exemplo em funcionamento**.

5. Outro exemplo

Dependendo do script que utilizarmos para mover o texto da barra de estado conseguiremos uns efeitos ou outros. A seguir podemos ver um **segundo exemplo sobre como mover um texto pela barra de estado** utilizando um efeito de movimento distinto.

Não vamos comentar este segundo exemplo porque já se encontra comentado no próprio código fonte, mas poderemos ver que é muito parecido ao anterior.

```
<html>
<head>
  <title>Segundo exemplo de texto em movimento</title>
</head>

<body>
<h1>Texto em movimento na barra de estado</h1>
<h2>Exemplo 2</h2>

<script language="javascript">

//variável com o texto a mostrar
var texto = "Bem-vindos a minha página web!!!"
//variavel coma osicao no texto. Colocar sempre a 0
var pos = 0

//crio uma funcao para mudar o texto da barra de estado
function textoEstado(){
```

```
  //incremento a posicao em 1 e extraio o texto a mostrar neste momento.
  pos = pos + 1
  textoAtual = texto.substring(0,pos)
  //coloco o texto que quero mostrar na barra de estado do navegador
  window.status = textoAtual
  //Chamamos outra vez a esta funcao para que continue    mostrando texto
  if (pos == texto.length){
    //se chegamos ao final, volto ao principio e faco um atraso superior
    pos = 0
    setTimeout("textoEstado()",1500)
  } else{
    //se nao chegamos ao final, continuo com a funcao um atraso minimo.
    setTimeout("textoEstado()",100)
  }
}

//chamo à função para colocar o texto em movimento
textoEstado()
</script>


</body>
</html>
```

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Marcar ou desmarcar todos os checkboxes de um formulário com Javascript

O exercício que vamos relatar neste artigo é bastante típico de trabalho com formulários em Javascript. Trata-se de fazer um link para que se possa marcar todos os campos checkbox que houver em um formulário de uma só vez, ou seja, sem ter que clica-los um por um para marcar todos. Também faremos a função que permite desmarcar todos os campos checkbox do formulário de uma só vez.

O exercício é simples de entender, mas podemos vê-lo em uma página a parte para termos uma idéia exata de nossas intenções.

O formulário HTML

Temos um formulário criado com HTML que é onde estarão os checkboxes que serão marcados e desmarcados automaticamente. O formulário é muito simples. Vemos a seguir:

```
<form name="f1">
Nome: <input type="text" name="nome">
<br>
<input type="checkbox" name="ch1"> Opcao 1
<br>
<input type="checkbox" name="ch2"> Opcao 2
<br>
<input type="checkbox" name="ch3"> Opcao 3
<br>
<input type="checkbox" name="ch4"> Opcao 4
<br>
//Outro campo de formulario:
<select name=outro>
<option value="1">Selecao 1
<option value="2">Selecao 2
</select>
<br>
```

```
<input type="submit">
<br>
<br>
<a href="javascript:selecionar_todo()">Marcar todos</a> |
<a href="javascript:deselecionar_todo()">Marcar nenhum</a>
</form>
```

O único que devemos observar é que colocamos diversos tipos de elementos no formulário. Na verdade só vamos trabalhar com o estado dos checkbox, mas não incluímos outros elementos porque o habitual em um formulário é que tenham elementos de vários tipos.

No final do formulário temos dois links para marcar ou desmarcar todos os checkboxes de uma só vez. Estes links chamam a duas funções Javascript que veremos agora.

Funções de Javascript

```
function selecionar_tudo(){
  for (i=0;i<document.f1.elements.length;i++)
    if(document.f1.elements[i].type == "checkbox")
      document.f1.elements[i].checked=1
}
```

A função selecionar_tudo() realiza um percorrido por todos os elementos do formulário. Para fazer um percorrido por todos os campos utiliza-se o array "elements", que salva uma referência com cada elemento que houver dentro do formulário.

No percorrido comprova se o elemento atual é de tipo "checkbox" (lembrar que o array elements contem todos os elementos, mas somente desejamos operar com os que sejam checkbox) e nesse caso, simplesmente atualiza-se o atributo "checked" ao valor 1, com o qual o chekbox se marcará.

```
function deselecionar_tudo(){
  for (i=0;i<document.f1.elements.length;i++)
    if(document.f1.elements[i].type == "checkbox")
      document.f1.elements[i].checked=0
}
```

A função deselecionar_tudo() é quase igual que a anterior. Realiza um percorrido a todos os elementos e no caso que sejam checkbox, se fixa a zero o atributo "checked" para que a caixa de seleção fique desmarcada.

O exemplo não tem mais mistério. Pode-se ver em funcionamento em uma página a parte.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Desabilitar o menu contextual do navegador com Javascript

Neste artigo vamos mostrar um método para desabilitar o menu contextual do navegador, que aparece clicando com o botão direito em qualquer área da página. Assim, podemos evitar que o usuário tenha acesso a algumas das opções do navegador, como ver o código fonte.

A primeira coisa que queremos destacar é que este exemplo não protege o código das páginas web que estamos publicando. Simplesmente coloca algumas travas para ver como fizemos a página, mas qualquer usuário espertinho poderá acessar ao código da página se realmente

quiser.

Para começar, na barra de menus do navegador, em "exibir - código fonte", pode-se acessar também ao código fonte das páginas. Portanto, se desejarmos que não seja visto nosso código, teremos que mostrar a página em uma nova janela do navegador, que devemos abrir mediante Javascript para que não inclua as barras de menus.

Mesmo se conseguíssemos evitar mostrar a barra de menus e o menu contextual, um usuário ainda poderia desabilitar Javascript para tentar ver o menu contextual sem que a página o empeça.

Porém, devemos saber que quando se envia uma página a um visitante, o arquivo HTML é salvo no disco rígido local desse usuário, então em último caso, a pessoa interessada simplesmente tem que acessar seus arquivos temporários da Internet para localizar a página que tem o código que deseja visualizar. Como o arquivo está fisicamente em seu computador, poderá fazer o que desejar com ele: abrir, modificar, salvar com outro nome, etc. Portanto nossos códigos nunca estarão totalmente seguros.

> **Nota:** A melhor solução para proteger um código é escrevê-lo no lado do servidor, com linguagens como ASP ou PHP. Ao estar no lado do servidor, os scripts se executarão no servidor e o visitante só receberá o código gerado dessa execução, no próprio código ASP ou PHP.

Portanto, não pode ser feito nada definitivo para ocultar um código que se executa no cliente, portanto esta solução proposta é só um detalhe que pode entorpecer a captura de um código, mas não serve para protege-lo definitivamente.

Sendo assim, o código que vamos a propor é muito mais simples do que se pode pensar. Simplesmente utilizaremos um evento de Javascript que se chama "oncontextmenu" e depende de "document". Atribuiremos uma função a este evento, que se executará no momento em que o usuário clique o botão direito para visualizar o menu contextual.

A função que vamos atribuir a este evento é a seguinte:

```
function desabilitar(){
   alert ("Esta função está desabilitada.\n\nDesculpem as moléstias.")
   return false
}
```

A função mostra uma mensagem de advertência, mas observemos o return false: é necessário para que não chegue a mostrar o menu contextual, porque se não colocarmos, se mostraria a mensagem de alerta mas em seguida se mostraria também o menu contextual, com o qual não serviria para nada o script.

Para atribuir esta função ao evento oncontextmenu, realizamos este código:

```
document.oncontextmenu=desabilitar
```

Tão simples como isso. O script completo, que colocaríamos entre <head> e </head> ficaria assim:

```
<script language=JavaScript>
<!--

function desabilitar(){
   alert ("Esta função está desabilitada.\n\nDesculpem as moléstias.")
   return false
}

document.oncontextmenu=desabilitar

// -->
</script>
```

Podemos ver o exemplo em funcionamento em uma página a parte.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Relógio em Javascript

Vamos ver um workshop prático sobre Javascript com o qual poderemos incluir um relógio em nossa página web. É um simples script, que poderemos colocar simplesmente copiando e colando, embora procuraremos explica-lo um pouco para os que estejam em condições de entender Javascript.

Para começar, vamos ver o reloginho que pretendemos criar:

03 : 30 : 27

É um relógio muito simples de maneira intencionada, para que possamos entender bem o exercício. Logo, o complicamos um pouco mais para lhe dar algo mais de atração.

Construir o formulário

Começamos colocando o campo de texto onde será mostrado o relógio. Devemos colocar um pequeno formulário onde somente teremos um campo de texto.

```
<form name="form_relogio">
<input type="text" name="relogio" size="10">
</form>
```

Não deveria haver nenhum problema nessas linhas de HTML. Só colocamos as etiquetas de início e final do formulário e dentro um campo de texto. Atribuímos um nome tanto ao formulário como ao campo de texto para logo acessarmos por esse nome mediante Javascript.

Função para atualizar o valor do relógio

Temos que construir uma função que busque a hora do sistema e a mostre no campo de texto.

Para pegar a hora vamos fazer uso do objeto Date de Javascript.

momentoAtual = new Date()

Se criarmos uma nova instância do objeto Date sem lhe passar parâmetros, inicia-se à data e hora atuais.

Logo temos que obter dessa instância de Date o que nos interessa, que é a hora, os minutos e segundos. Fazemos isto utilizando os correspondentes métodos do objeto Date.

```
hora = momentoAtual.getHours()
minuto = momentoAtual.getMinutes()
segundo = momentoAtual.getSeconds()
```

Agora combinamos os resultados para construir a hora com o formato que desejarmos.

```
horaImprimivel = hora + " : " + minuto + " : " + segundo
```

Por último colocamos no campo de texto dol formulário o valor atual da hora.

```
document.form_reloj.reloj.value = horaImprimible
```

Por agora a função fica desta maneira:

```
function moveRelogio(){
   momentoAtual = new Date()
   hora = momentoAtual.getHours()
   minuto = momentoAtual.getMinutes()
   segundo = momentoAtual.getSeconds()

   horaImprimivel = hora + " : " + minuto + " : " + segundo

   document.form_relogio.relogio.value = horaImprimivel
}
```

A função deve charmar a si mesma

Com estas linhas de código obtemos a hora e a atualizamos em seu campo de texto, mas ainda não terminou, necessitamos que essa função se chame constantemente e cada segundo para que atualize o valor de nosso campo de texto constantemente também.

Para isso, o melhor é que a função se encarregue também de chamar a si mesma a cada segundo, assim voltará a fazer todo o processo de obtenção e atualização da hora e por último chamará a si mesma ao final de um segundo. Este processo se repetirá sempre até sairmos da página.

A linha de código para chamar a si mesma, que colocaremos na última linha da função é a seguinte:

```
setTimeout("moveRelogio()",1000)
```

A função setTimeout serve para fazer o adiantamento antes de executar a sentença. A sentença é uma simples chamada à função e o adiantamento é de 1000 milésimos de segundos (um segundo).

Colocar o relógio em funcionamento

Finalmente necessitamos colocar o relógio em funcionamento. Isto pode ser feito uma vez terminado o carregamento da página com o administrador de eventos onload, que coloca-se no <body>.

```
<body onload="moveRelogio()">
```

Isto quer dizer que quando termine de carregar a página se chamará à função moveRelogio(), que se encarregará de mover o relógio e chamar a si mesmo para fazer o processo de maneira contínua.

Código inteiro

O código da página fica desta maneira:

```
<html>
<head>
<title>Relogio com Javascript</title>
<script language="JavaScript">
function moveRelogio(){
   momentoAtual = new Date()
   hora = momentoAtual.getHours()
   minuto = momentoAtual.getMinutes()
   segundo = momentoAtual.getSeconds()

   horaImprimivel = hora + " : " + minuto + " : " + segundo

   document.form_relogio.relogio.value = horaImprimivel

   setTimeout("moveRelogio()",1000)
}
</script>
</head>

<body onload="moveRelogio()">

Vemos aqui o relógio funcionando...

<form name="form_relogio">
<input type="text" name="relogio" size="10">
</form>

</body>
</html>
```

Acréscimos para o relógio

Podemos fazer muitas coisas para melhorar os aspectos deste relógio. Por exemplo, dar um pouco de estilo ao campo de texto ou fazer co que ninguém possa pousar em cima do campo de texto para atualizar manualmente o valor da hora. Vamos ver alguma coisinha:

Estilo ao relógio

Com folhas de estilo podemos mudar a aparência do relógio para torná-lo um pouco mais especial. Este é um exemplo o qual poderíamos colocar.

```
<input type="text" name="relogio" size="10" style="background-color : Black; color : White; font-family : Verdana, Arial, Helvetica; font-size : 8pt; text-align : center;">
```

Evitar que acessem ao campo de texto

Assim ninguém poderá modificar o texto do relógio manualmente. Conseguimos isso com Javascript. O administrador de eventos onfocus se ativa quando o campo de texto adquire o foco da aplicação. Nesse momento noós tiraremos o foco da aplicação com o método blur(). O botão ficaria assim:

```
<input type="text" name="relogio" size="10" onfocus="window.document.form_relogio.relogio.blur()">
```

Fazer com que sempre tenhamos dois dígitos na hora, minutos e segundos.

Para conseguir que a hora tenha sempre um formato hh : mm : ss devemos brincar um pouco com os valores de tempo como se fossem strings. Para isso, o primeiro que teremos que fazer

é construir um string a partir do valor (hora, minuto ou segundo) que quisermos formatar. Logo, observaremos esse string para saber se temos que lhe acrescentar um dígito ao valor.

Vejamos como obtemos o string a partir do número de segundos obtido. Faremos para os segundos, logo, a hora e os minutos serão realizados de maneira similar.

```
str_segundo = new String (segundo)
```

Em seguida, observamos se temos só um caractere no string, porque se for assim, temos que concatenar um zero ("0") ao princípio.

```
if (str_segundo.length == 1)
segundo = "0" + segundo
```

Tudo junto

Vejamos outra vez nosso exemplo em uma só peça para ver como ficam todas estas melhoras:

```
<html>
<head>
<title>Relogio com Javascript</title>
<script language="JavaScript">
function moveRelogio(){
   momentoAtual = new Date()
   hora = momentoAtual.getHours()
   minuto = momentoAtual.getMinutes()
   segundo = momentoAtual.getSeconds()

   str_segundo = new String (segundo)
   if (str_segundo.length == 1)
      segundo = "0" + segundo

   str_minuto = new String (minuto)
   if (str_minuto.length == 1)
      minuto = "0" + minuto

   str_hora = new String (hora)
   if (str_hora.length == 1)
      hora = "0" + hora

   horaImprimible = hora + " : " + minuto + " : " + segundo

   document.form_relogio.relogio.value = horaImprimivel

   setTimeout("moveRelogio()",1000)
}
</script>
</head>

<body onload="moveRelogio()">

Vemos aqui o relógio funcionando...

<form name="form_relogio">
<input type="text" name="relogio" size="10" style="background-color : Black; color : White; font-family : Verdana,
Arial, Helvetica; font-size : 8pt; text-align : center;" onfocus="window.document.form_relogio.relogio.blur()">
</form>

</body>
</html>
```

Este segundo relógio pode ser visto a seguir:

03 : 30 : 27

Se desejarmos podemos abrir uma página web para visualizar:

- O relógio simples
- O relógio melhorado

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Scripts diferentes para cada navegador

Um dos maiores problemas que um programador encontra na hora de criar páginas com DHTML é lidar com os diferentes modelos de objetos que têm cada um dos navegadores, e cujas diferenças não só se dão com os navegadores das distintas companhias, como também entre a mesma companhia há diferenças na modalidade dos objetos dependendo das versões. Por exemplo, o modelo de objetos de Netscape 6 é diferente do modelo de objetos de Netscape 4.

Tudo isto nos cria o problema na hora de escrever scripts de ter que duplicar o código de cada uma das nossas funções dependendo do navegador que se utiliza. Devido à grande variedade de navegadores e versões disponíveis, isto faz com que nossas funções acabem sendo muito maiores e mais ilegíveis do que o esperado.

Uma solução para este problema é a criação de diferentes arquivos que contenham os scripts para cada um dos navegadores que existem, ligando no momento do carregamento da página com o arquivo de scripts do navegador e a versão que estamos utilizando. Desta forma, os scripts são muito mais simples já que um script se criará enfocado a uma só versão, e ademais, no caso de que saia no mercado um novo navegador com um modelo de objetos diferente, não teremos que mudar todos nossos scripts, simplesmente bastará modificar o script que detecta o navegador, e em seguida escrever um novo arquivo que contenha os mesmos scripts que os anteriores, mas com o novo modelo de objetos.

O seguinte código pretende ser um exemplo reduzido do anteriormente exposto.

Nosso objetivo será diferenciar entre Netscape e Internet Explorer, de forma que ao clicar sobre um único botão, apareça uma mensagem de alerta diferente dependendo do navegador que utilizarmos.

O primeiro passo é a criação dos diferentes arquivos que vão conter os scripts específicos para cada um dos navegadores. Em nosso caso serão dois arquivos, um que chamaremos Internet.js que conterá os scripts para Internet Explorer, e outro chamado Netscape.js que conterá os scripts para Netscape. O conteúdo de Internet.js será o seguinte:

```
function mostraAlerta(){
   alert("Estou utilizando IE")
}
```

O conteúdo de Nescape.js será o seguinte:

```
function mostraAlerta(){
   alert("Estou utilizando NS")
}
```

Por outro lado, no cabeçalho de cada uma das páginas, deveremos incluir um script que ligue a cada um dos navegadores, dependendo do que utiliza o usuário.

```
<script language="javascript">

ns=(document.layers)? true:false
ie=(document.all)? true:false

ponNs="<script language='javascript1.2' src='netscape.js'> <\/script>"
ponIe="<script language='javascript1.2' src='internet.js'> <\/script>"

if (ns) {document.write(colocarNs)}
if (ie) {document.write(colocarIe)}

</script>
```

*** Observem na contra-barra "\" colocada antes da barra "/" em </script> nos document.write().**

Por último incluímos no corpo da página um botão que chama à função que está definida dentro dos arquivos de scripts:

```
<input type="button" value="Clicar" onclick="mostraAlerta()">
```

De forma que a página fique com o seguinte aspecto:

```
<html>
<head>
   <title>Untitled</title>
<script language="javascript">

ns=(document.layers)? true:false
ie=(document.all)? true:false

ponNs="<script language='javascript1.2' src='netscape.js'> <\/script>"

ponIe="<script language='javascript1.2' src='internet.js'> <\/script>"

if (ns) {document.write(colocarNs)}

if (ie) {document.write(colocarIe)}

</script>

</head>

<body>

<form>
<input type="button" value="Clicar" onclick="mostraAlerta()">
</form>

</body>
</html>
```

*Artigo por **Carlos Cuenca Díaz***

Tamanho dos campos relativo ao navegador

Se criarmos um formulário com uns campos de texto em uma página web poderemos observar que Internet Explorer não coloca o mesmo tamanho aos campos de texto que Netscape. Na verdade, os campos que coloca Netscape são de um tamanho maior para um mesmo atributo size no campo.

Geralmente se soluciona colocando os campos de texto menores para que caibam no desenho em Netscape, mas em algumas ocasiões queremos que os campos sejam do tamanho justo e que no Internet Explorer não sejam tão pequenos e no Netscape tão grandes. Isto pode acontecer, por exemplo, em desenhos calculados muito à medida.

Concretizando, o tamanho dos campos se aplica com o atributo size da etiqueta <INPUT> e queremos fazer com que o size indicado dependa do navegador que estivermos usando.

Distinguir os navegadores

Para isso deveremos criar um script de Javascript que distinga entre os dois navegadores e escrever mediante javascript o atributo size com um valor distinto para cada caso.

Distinguir dinamicamente entre os navegadores é uma tarefa que pode ser bastante complicada e digna de outro artigo, mas poderíamos fazer de uma maneira parecida a esta:

```
if (document.layers)
//faco coisas para Netscape 4
else
// faco coisas para outros navegadores
```

Com este código discrimino somente entre a versão 4 de Netscape e todos os demais navegadores. É porque Netscape 4 é o único navegador cuja hierarquia de objetos inclui um objeto denominado layers. Ao introduzir esse nome de objeto na expressão a avaliar do if, obteremos casos positivos se o objeto existe (Netscape4) e negativos se não existirem (os demais).

Escrever coisas distintas em cada caso

Agora vamos colocar o código Javascript que escreveria um campo de texto com um valor de size distinto para cada navegador.

```
if (document.layers)
document.write("<INPUT type=text size=12>")
else
document.write("<INPUT type=text size=16>")
```

Colocamos o script em um formulário

É bem simples, só nos resta colocar esse código dentro de um formulário e teremos um campo de texto com sizes distintos, mas com o mesmo tamanho na tela.

```
<FORM>
```

```
<script language="JavaScript">
if (document.layers)
document.write("<INPUT type=text size=12>")
else
document.write("<INPUT type=text size=16>")
</script>
</FORM>
```

Pode-se ver este exemplo em uma página a parte.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Estilos diferentes para cada navegador

Graças a um e-mail de um usuário que queria colocar um fundo de tela diferente dependendo do tipo de navegador que nos fez pensar em dois scripts que podem servir para este Workshop de Javascript.

Trata-se de uns scripts que permitem mostrar estilos na página web dependendo do navegador com o qual se acessa à página, de modo que se um usuário entra com IExplorer veria um fundo, tipografias e outros elementos com formas ou estilos diferentes dos que veria outro usuário que tenha acessado com Netscape.

Começamos pelo exemplo exato que nos pedia o usuário do e-mail, para colocar somente fundos diferentes. É um script bastante específico que para muitos de vocês parecerá "inútil", mas pelo menos servirá para aprender alguma coisinha.

Com a etiqueta <BODY>

A primeira idéia que tive foi de escrever a etiqueta <BODY> inteira dentro de um bloco de script onde temos um if que nos permite distinguir entre navegadores.

O código ficaria algo assim:

```
<html>
<head>
   <title>Fundos diferentes para cada navegador</title>
</head>

<script language="JavaScript">
if (document.layers)
   document.write ("<body background=nts.jpg>")
else
   document.write ("<body background=ie.jpg>")
</script>

</body>
</html>
```

Vemos que para averiguar se temos Netscape ou Internet Explorer acessamos ao objeto layers do documento. Como só Netscape tem esse objeto, somente para Netscape essa avaliação será verdadeira. Isto é um truque rápido, embora deveria ser estudado à parte porque não funciona bem com todos os navegadores, por exemplo, Netscape 6 mostraria o fundo de

Explorer. Na verdade, só se distingue entre Netscape 4 e todos os demais navegadores, já que certamente, esse objeto só está disponível nesse navegador.

Voltando ao exemplo, dependendo se se utiliza Netscape 4 ou outro navegador se mostra uma etiqueta <BODY> com um atributo, o da imagem de fundo diferente.

Com estilos

Existe outra maneira de atribuir fundos diferentes dependendo do navegador e não só os fundos como também tipos de letra, tamanhos e em geral todo o que pode ser definido utilizando os estilos CSS.

Neste exemplo, supomos que sabe algo de CSS e da definição de estilos para todo um site incluído em um arquivo externo com sintaxe CSS. Para conhecer as CSS temos um excelente manual em CriarWeb.com.

A base do exemplo é a mesma, utilizamos um bloco script no qual averiguamos que navegador está sendo utilizando e dependendo do navegador mostra-se uma etiqueta com uns atributos ou outros. Na verdade, a etiqueta que colocamos dinamicamente em função do navegador é <LINK> que serve para incluir uma declaração de estilos externa.

Seria algo assim:

```
<html>
<head>
   <title>Linko com estilos dinamicamente</title>
<script LANGUAGE="JavaScript">
if (document.layers) {
   document.write("<LINK REL='stylesheet' HREF='estilo_nt.css' TYPE='text/css'>"); }
else {
   document.write("<LINK REL='stylesheet' HREF='estilo_ie.css' TYPE='text/css'>"); }
</script>
</head>

<body>

<h1>Bem-vindos a minha página</h1>

</body>
</html>
```

Como vemos, no caso de estar em Netscape 4 se carregará a folha de estilos chamada "estilo_nt.css" e se nosso navegador é outro qualquer se carregará "estilo_ie.css".

Este segundo caso pode ser utilizado para muitas coisas. Principalmente pode-se utilizar para evitar um efeito que existe no uso de diferentes navegadores, que consiste em que na mesma definição de fontes, os tamanhos das mesmas diferem em Netscape e Internet Explorer. (As mesmas fontes, em Netscape saem menores que em Explorer). É por isso que incluindo uma folha de estilos diferente pode-se colocar os tamanhos das fontes desejados para cada navegador.

Isso é tudo. Esperemos que estes scripts dêem idéias para resolver outros problemas.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Tabela de cores com Javascript

Em HTML construímos qualquer cor misturando o vermelho, verde e azul (RGB) nas proporções que desejarmos. Isto é um fato que deveria saber antes de ler este artigo. Explicamos com detalhes a construção de cores no artigo As cores e HTML. Será necessário que, aquele que não estiver familiarizado com este assunto, leia o artigo.

Além de como construir cores, o artigo As cores e HTML mostra também quais são as cores puras, que se vêem sem problemas em todas as profundidades de cor que possa ter a configuração dos computadores dos visitantes. Para a construção de cores puras misturamos as cores RGB sempre nestas quantidades 00, 33, 66, 99, CC, FF. Novamente, para quem não conhece isto deve ler a reportagem assinalada anteriormente.

O exemplo que pretendemos construir tem a ver com as cores puras em tyodas as definições. Trata-se de construir uma tabela em uma página web que contenha todos as cores puras, além do código RGB de cada cor. Esta tabela pode servir para selecionar uma cor que pretendemos utilizar em uma página web. Se nos limitarmos somente a utilizar as cores da tabela teremos a segurança que nossa paleta será respeitada em todos os casos.

Para gerar todos as cores puras vamos utilizar três arrays Javascript com os possíveis valores para cada uma das cores RGB. Portanto, teremos três arrays como os que podem ser vistos a seguir:

```
var r = new Array("00","33","66","99","CC","FF");
var g = new Array("00","33","66","99","CC","FF");
var b = new Array("00","33","66","99","CC","FF");
```

Para escrever a tabela na página web, faremos um percorrido a estes três arrays. Para isso, vamos utilizar loops aninhados, que são loops dentro de outros loops.

Vamos tratar de explicar porque necessitamos os loops aninhados; se fizermos as contas das cores que devemos ir gerando obteremos uma lista como a que segue:

#000000 #000033 #000066 #000099 #0000CC #0000FF #003300 #003333 #003366 #003399 #0033CC #0033FF #006600 #006633 #006666 #006699 #0066CC #0066FF...
Pode-se ver a conta completa aqui.

Então, vemos que no início os três valores de RGB valem 00 e como em sucessivas repetições se vai aumentando o valor de B (o valor atribuído ao azul) até chegarmos a FF. Para continuar, aumenta-se o valor de G e voltamos a realizar a conta com B. É como se contássemos e as unidades fossem os valores de RBG.

O caso é que realizamos a conta com o valor B, quando chegamos a FF aumentamos o valor de G e quando chegarmos a FF em G aumentaremos em um valor R. Assim, pode-se ver a estrutura em pseudocódigo como esta.

```
Contamos desde 00 até FF com o valor R{
   Contamos desde 00 até FF com o valor G{
      Contamos desde 00 até FF com o valor R{
        Imprimimos o valor atual de RGB
      }
   }
}
```

Esta estrutura imprime todos as cores puras, e já é próxima a nossa solução, embora ainda não esteja escrita em Javascript e falte colocar todas as etiquetas HTML que necessitamos para mostrar uma tabela em uma página web.

Como não importa ir um pouco mais devagar contanto que todo o mundo entenda o problema, vamos escrever em Javascript este loop para que simplesmente liste as cores puras, sem introduzi-las ainda em uma tabela. Será interessante para ver um pouco melhor o funcionamento de loops aninhados.

```
//criamos os arrays
var r = new Array("00","33","66","99","CC","FF");
var g = new Array("00","33","66","99","CC","FF");
var b = new Array("00","33","66","99","CC","FF");

//fazemos o loop aninhado
for (i=0;i<r.length;i++) {
   for (j=0;j<g.length;j++) {
      for (k=0;k<b.length;k++) {
         //creamos el color
         var nuevoc = "#" + r[i] + g[j] + b[k];
         //imprimimos a cor
         document.write(novoc + "<br>");
      }
   }
}
```

Para percorrer um array os loops se criam com um índice que servirá para acessar à posição atual do array. Os índices em arrays começam em 0, é por isso que nossos loops for contém uma iniciação a 0 da variável que vai servir de índice. Ademais o índice deve crescer de um em um até chegar à última posição do array, que se obtém acessando a sua propriedade length (que salva a longitude ou o número de elementos do array).

Colocando um loop dentro de outro poderemos realizar essa conta. O loop mais externo será o que menos vezes se executar, portanto com o loop exterior levaremos a conta de R. O loop do meio será para levar a conta de G e o mais interno (o que mais vezes se repetirá) para levar a conta de B, que é o valor que vai mudando constantemente.

Podemos vê-la em funcionamento neste link.

A tabela das cores completa

Para terminar, vamos ver o exemplo completo, com todas as linhas de código que incluem os textos HTML necessários para que a tabela saia convenientemente formatada e com as cores de fundo em cada uma das células iguais à cor atual.

Para isso, o primeiro que faremos é escrever o cabeçalho da tabela e a finalização da mesma, que ficam fora da estrutura de loops. Dentro dos loops realizaremos as sentenças para imprimir cada uma das filas e células da tabela.

Nossa tabela vai ter em cada fila um conjunto de cores, onde os valores RG não mudam e o valor B varia entre todos os possíveis. Na seguinte fila se incrementaria em um o valor G e faríamos outra vez a conta de valores B... assim até terminarmos com todos os valores R, G e B possíveis.

O código é o seguinte:

```
<table width="80%">
<script language="javascript">
var r = new Array("00","33","66","99","CC","FF");
var g = new Array("00","33","66","99","CC","FF");
var b = new Array("00","33","66","99","CC","FF");

for (i=0;i<r.length;i++){
   for (j=0;j<g.length;j++) {
      document.write("<tr>");
      for (k=0;k<b.length;k++) {
         var novoc = "#" + r[i] + g[j] + b[k];
         document.write("<td bgcolor=\"" + novoc + "\" align=center>");
         document.write(novoc);
         document.write("</td>");
      }
      document.write("</tr>");
   }
}
</script>
</table>
```

Vemos que antes de começar o loop mais interno, cria-se uma nova célula com <TR> e que quando se acaba, termina também a célula com </TR>. Ademais, dentro do loop mais interno se compõe primeiro a cor atual e logo se escreve uma célula com o atributo bgcolor atribuído a essa cor atual e dentro dela o texto da cor atual.

A tabela que nos gera este script pode ser vista aqui:

#000000	#000033	#000066	#000099	#0000CC	#0000FF
#003300	#003333	#003366	#003399	#0033CC	#0033FF
#006600	#006633	#006666	#006699	#0066CC	#0066FF
#009900	#009933	#009966	#009999	#0099CC	#0099FF
#00CC00	#00CC33	#00CC66	#00CC99	#00CCCC	#00CCFF
#00FF00	#00FF33	#00FF66	#00FF99	#00FFCC	#00FFFF
#330000	#330033	#330066	#330099	#3300CC	#3300FF
#333300	#333333	#333366	#333399	#3333CC	#3333FF
#336600	#336633	#336666	#336699	#3366CC	#3366FF
#339900	#339933	#339966	#339999	#3399CC	#3399FF
#33CC00	#33CC33	#33CC66	#33CC99	#33CCCC	#33CCFF
#33FF00	#33FF33	#33FF66	#33FF99	#33FFCC	#33FFFF

#660000	#660033	#660066	#660099	#6600CC	#6600FF
#663300	#663333	#663366	#663399	#6633CC	#6633FF
#666600	#666633	#666666	#666699	#6666CC	#6666FF
#669900	#669933	#669966	#669999	#6699CC	#6699FF
#66CC00	#66CC33	#66CC66	#66CC99	#66CCCC	#66CCFF
#66FF00	#66FF33	#66FF66	#66FF99	#66FFCC	#66FFFF
#990000	#990033	#990066	#990099	#9900CC	#9900FF
#993300	#993333	#993366	#993399	#9933CC	#9933FF
#996600	#996633	#996666	#996699	#9966CC	#9966FF
#999900	#999933	#999966	#999999	#9999CC	#9999FF
#99CC00	#99CC33	#99CC66	#99CC99	#99CCCC	#99CCFF
#99FF00	#99FF33	#99FF66	#99FF99	#99FFCC	#99FFFF
#CC0000	#CC0033	#CC0066	#CC0099	#CC00CC	#CC00FF
#CC3300	#CC3333	#CC3366	#CC3399	#CC33CC	#CC33FF
#CC6600	#CC6633	#CC6666	#CC6699	#CC66CC	#CC66FF
#CC9900	#CC9933	#CC9966	#CC9999	#CC99CC	#CC99FF
#CCCC00	#CCCC33	#CCCC66	#CCCC99	#CCCCCC	#CCCCFF
#CCFF00	#CCFF33	#CCFF66	#CCFF99	#CCFFCC	#CCFFFF
#FF0000	#FF0033	#FF0066	#FF0099	#FF00CC	#FF00FF
#FF3300	#FF3333	#FF3366	#FF3399	#FF33CC	#FF33FF
#FF6600	#FF6633	#FF6666	#FF6699	#FF66CC	#FF66FF

#FF9900	#FF9933	#FF9966	#FF9999	#FF99CC	#FF99FF
#FFCC00	#FFCC33	#FFCC66	#FFCC99	#FFCCCC	#FFCCFF
#FFFF00	#FFFF33	#FFFF66	#FFFF99	#FFFFCC	#FFFFFF

Podemos ver uma página web onde está somente esta tabela.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Sub-menu em outra janela

Vamos realizar um pequeno exemplo sobre como trabalhar com janelas secundárias ou popups. As janelas secundárias são essas janelinhas que se abrem além da janela principal do navegador e em geral, molestam um pouco em determinados sites gratuitos.

Para abrir essas janelas se utiliza a linguagem Javascript, mais concretamente, o método open() do objeto window. Não vamos explicá-lo aqui, porque já temos um artigo inteiro que explica detalhadamente a abertura de janelas secundárias e as características com as quais podemos abrir: Abrir janelas secundárias

Neste exemplo, iremos um pouco mais além, vamos criar uma janela secundária e a partir dela, vamos acessar às propriedades da janela pai, ou seja, a janela que a abriu. O exercício será o seguinte:

Teremos uma página com fundo branco, um campo de texto vazio e um botão. Quando se clique o botão, abriremos um popup que conterá uma tabela com as cores puras de HTML. O visitante poderá selecionar uma dessas cores e então, o fundo da página pai mudará para esta cor e a cor será escrita no campo de texto. É muito mais fácil ver o exemplo em funcionamento para compreender a explicação.

Página pai

A página original conterá um simples formulário com um botão e um campo de texto. Ademais, conterá o script Javascript necessário para abrir a janela secundária.

```
<html>
<head>
   <title>Sub-menu em janela a parte</title>
<script language="JavaScript">
function lancarSubmenu(){
  window.open("submenu_janela2.html","janela1","width=400,height=400,scrollbars=YES")
}
</script>
</head>
<body bgcolor="#ffffff">
<form>
```

```
<input type="text" name="colorin" size="12" maxlength="12">
<br>
<br>
<input type="button" value="Lançar sub-menu" onclick="lancarSubmenu()">
</form>
</body>
</html>
```

A função lancarSubmenu() é a que contém o script para abrir o popup. Para isso utiliza o método open() do objeto window, cujo uso foi descrito no artigo de Abrir janelas secundárias.

O formulário é dos mais normais. O único destacável é o atributo onclick do botão, que serve para definir as ações para serem executadas ao clicarmos o botão, neste caso uma chamada à função que abre a janela secundária.

Página secundária

A página secundária é um pouco mais complexa, mas a parte que nos interessa neste exercício é bastante simples. O importante da página é que tem que acessar à janela pai para modificar seu estado e para isso utiliza um objeto Javascript: opener.

O objeto opener está disponível nas páginas que são janelas secundárias e faz referência à janela que a abriu, ou seja, a janela pai. Em outras palavras, o objeto opener na janela popup é um sinônimo do objeto window na janela pai.

O script que acessa à janela pai é o seguinte:

```
<script language="JavaScript">
function atualizaPai(minhaCor){
   window.opener.document.bgColor = minhaCor
   window.opener.document.forms[0].colorin.value = minhaCor
}
</script>
```

A função atualizaPai() é a encarregada de realizar o trabalho. Recebe o código da cor sobre a qual foi clicada. Na primeira linha mudamos a cor da página web pai e na segunda linha colocamos o código RGB da cor recebida por parâmetro no campo de texto.

Como vemos, o objeto opener também depende do objeto window da página, como todos os demais objetos da hierarquia Javascript.

O resto da página é parecido a um artigo anterior de Javascript no qual escrevíamos com Javascript uma tabela de cores puras, com umas pequenas modificações para adapta-la às nossas necessidades. Pode-se ver a seguir:

```
<table width="80%" align="center" cellpadding="1" cellspacing="1">
<script language="javascript">
var r = new Array("00","33","66","99","CC","FF");
var g = new Array("00","33","66","99","CC","FF");
var b = new Array("00","33","66","99","CC","FF");
for (i=0;i<r.length;i++)
   for (j=0;j<g.length;j++) {
      document.write("<tr>");
      for (k=0;k<b.length;k++) {
         var nuevoc = "#" + r[i] + g[j] + b[k];
         document.write("<td bgcolor=\"" + novoc + "\" align=center><font size=1 face=verdana>");
         document.write("<a href=\"javascript:atualizaPai('" + novoc + "')\">");
```

```
        document.write(novoc);
      }
      document.write("</a></font></tr>");
    }
</script>
</table>
```

Visto a complicação deste script e dado que não vamos explicá-lo tudo outra vez, pode ser altamente recomendável a leitura do artigo anterior chamado Tabela de cores com Javascript.

O mais importante para nós agora é entender que este script cria uma tabela com todos as cores puras, colocadas cada uma em uma célula. Dentro de cada célula se escreve um link que chama à função atualizaPai() passando-lhe o código da cor que há que ser utilizada.

Podemos ver o exemplo em funcionamento.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Degradê de cor Javascript

Vamos ver um exemplo típico de efeito especial em uma página web: um degrade de cores para o fundo da página.

O degrade consiste em mudar a cor de fundo da página aos poucos para ir se aproximando à outra cor. Podemos fazer muitos degrades diferentes para páginas web com pouco esforço, poucos conhecimentos técnicos e bastante imaginação.

Pode-se ver um exemplo de degrade em uma página a parte, para ter uma idéia exata do que vamos explicar neste artigo. Ademais, no final do artigo, veremos um script que permite fazer uma ampla gama de degrades, como este exemplo de degrade mais complexo.

Complicação do degrade

A maior complicação que encontramos na hora de criar um script para fazer um degrade é que as cores se expressam em hexadecimal e em Javascript trabalhamos com números decimais. De modo que, para converter nossos números decimais em hexadecimais e poder assim utiliza-los para indicar cores criamos uma função especial:

```
function converteHexadecimal(num){
   var hexaDec = Math.floor(num/16)
   var hexaUni = num - (hexaDec * 16)
   return hexadecimal[hexaDec] + hexadecimal[hexaUni]
}
```

A função devolve o número passado por parâmetro em forma hexadecimal. Por exemplo, se recebe 140 devolve 8C. Se recebe 15 devolve 0F.

Para isso, temos a ajuda de um Array criado anteriormente que salva conversão de números decimais a hexadecimais das unidades básicas, ou seja, do 0 à letra F. Esta é a linha que cria o Array.

```
hexadecimal = new Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F")
```

A função primeiro calcula os grupos de 16 que podem ser feitos a partir do número que recebemos. Os números destes grupos seriam as "hexa-dezenas".

Logo, calculamos o resto da divisão anterior, ou dito de outra forma, quantas unidades nos restam depois de tomar estas "hexa-dezenas". Este segundo número, serão as unidades. Ao juntar as unidades com os grupos de 16 que obtivemos, convertendo ambos valores por meio do array de conversão decimal a hexadecimal, conseguimos a conversão buscada.

Criando a seqüência de cores para o degrade

Uma vez que pudermos converter do valor decimal (necessário para levar a conta em Javascript) ao hexadecimal (necessário para indicar um cor) nossa única tarefa será contar em decimal as cores e converte-las a hexadecimal antes de mudar a propriedade document.bgColor, que como sabemos é a propriedade que salva a cor de fundo da página.

Quisemos fazer um primeiro exemplo muito simples para que possa ser entendido mais facilmente. O exemplo foi mostrado anteriormente em funcionamento. Trata-se de uma escala de cinzas que começa em preto e termina em branco.

```
cor_decimal = 0
```

Primeiro, iniciamos a variável cor_decimal, que é a que salva a cor atual a mostrar, em formato decimal. Só salvamos um valor da cor pois, ao ser uma escala de cinzas, todos os valores RGB são o mesmo.

Teremos uma função chamada degrade que será a encarregada de realizar a maior parte do trabalho.

```
function degrade(){
   cor_decimal ++
   cor_hexadecimal = converteHexadecimal(cor_decimal)
   document.bgColor = cor_hexadecimal + cor_hexadecimal + cor_hexadecimal

   //chamo com um atraso
   if (cor_decimal < 255)
      setTimeout("degrade()",1)
}
```

A função se encarrega de incrementar a cor_decimal em uma unidade, converte-la em hexadecimal e coloca-la dentro de propriedade document.bgColor para atualizar o fundo da página.

Finalmente, e nisso se baseiam muitos efeitos especiais de Javascript, se chama a função a si mesma com um atraso. No exemplo podemos ver que se a cor_decimal for menor de 255 (que é o máximo que se pode alcançar em cores) se chama à função com setTimeout, que é a que nos cria o atraso.

O código deste exemplo simples pode ser visto inteiro a seguir:

```
<html>
<head>
   <title>Exemplo de degrade 1</title>
</head>
```

```
<body bgcolor=000000>

<h1 align="center">Degradando...</h1>
<h2>Exemplo 1</h2>
Nesta página podemos ver um exemplo de degrade de preto a branco, com uma só passada. Só se vê o texto quando o fundo for suficientemente branco para que contraste.

<script language="javascript">

hexadecimal = new Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F")

function converteHexadecimal(num){
   var hexaDec = Math.floor(num/16)
   var hexaUni = num - (hexaDec * 16)
   return hexadecimal[hexaDec] + hexadecimal[hexaUni]
}

cor_decimal = 0

function degrade(){
   cor_decimal ++
   cor_hexadecimal = converteHexadecimal(cor_decimal)
   document.bgColor = cor_hexadecimal + cor_hexadecimal + cor_hexadecimal

   //chamo com um atraso
   if (cor_decimal < 255)
      setTimeout("degrade()",1)
}

degrade()
</script>
</body>
</html>
```

Se quisermos, também podemos ver o exemplo em funcionamento em uma página a parte.

Mais degrades

Provavelmente este não seja o exemplo mais útil para o leitor que deseja implementar um degrade em sua página, pois é um pouco simples e específico. Para solucionar este assunto, criamos um sistema onde ode se configurar o tipo de degrade da página com uma série de variáveis. É um exemplo que se baseia no que acabamos de ver, embora tenha muitas variáveis para parametrizar o comportamento do degrade e que se adapte às necessidades do desenvolvedor.

Por falta de tempo não vamos a explicar todo o script, iremos observar somente as variáveis que permitem configura-lo.

cor_inicio = new Array(150,150,255)

Com cor_incio configuramos a cor que se mostra ao princípio na página. Indicamos com um array, onde em cada campo, colocamos o valor decimal de cada um das três cores RGB.

cor_fim = new Array(255,99,0)

Com cor_fim configuramos a cor a qual vai tender nossa degrade, de maneira idêntica a como fizemos em cor_inicio

passos = 100

é o número de passos que vamos utilizar para alcançar o valor da cor final, desde a cor de início.

comportamento = 1

Esta variável serve para definir o comportamento do script em quatro possíveis valores:

1. Realiza um loop infinito desde a cor de início à cor fim e da cor fim à cor de início, para voltar a começar. E irá repetindo sempre.
2. Realiza uma passada desde a cor de início à cor fim. Termina quando acaba a passada.
3. Realiza uma passada da cor de início ao fim e uma volta desde o fim ao início. Termina quando realiza a ida e a volta.
4. Um loop infinito com uma parada de 10 segundos entre quando fez a ida e a volta, antes de voltar a começar outra ida e volta.

Exemplos de degrade completo:

- Degrade infinito de lilás a laranja.
- Uma passada de azul a vermelho.
- De preto a amarelo e de amarelo a preto. Uma passada..
- Loop infinito com parada. De preto a branco..

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Validar inteiro no campo de formulário

Suponhamos que temos um campo de um formulário onde queremos que figure sempre um valor numérico inteiro. Um exemplo poderia ser um campo onde queremos salvar o número de um ano, onde, logicamente, não cabem decimais e nem letras.

Neste exercício vamos realizar um script que procure obter um número inteiro do valor que tiver escrito o usuário em um campo de texto. Se for um número inteiro ou puder converte-lo a um inteiro, coloque tal valor inteiro no campo. Se não puder obter um valor numérico inteiro apague o conteúdo do campo e deixe-o vazio. Faremos com Javascript, já que é a linguagem do lado do cliente -para operar no âmbito do navegador- mais difundida.

Para aclarar o funcionamento do exercício podemos ver o exemplo completo em uma página a parte.

Este exercício serve também para aprender a manejar as funções incorporadas de Javascript parseInt() e isNaN(). A primeira serve para converter um valor a número inteiro e a segunda para ver se um dado é um valor numérico. As duas podem conhecidas com mais profundidade nos primeiros capítulos do manual de Javascript II.

Outro tema importante que há que conhecer é a hierarquia de objetos do navegador, porém neste caso, faremos um esforço em explica-la para aquelas pessoas que não a conheçam.

Função validarInteiro()

Primeiro, vamos realizar uma função que fará a maior parte do trabalho, visto que é a encarregada de validar se um dado é um número inteiro. Esta função recebe um valor, que é o dado que desejamos validar e se for um inteiro devolverá tal qual. Se não for, tentará convertê-lo a inteiro e se conseguir devolverá esse valor. Finalmente, se a tentativa de

converte-lo não der resultado, devolverá uma cadeia vazia.

```
function validarInteiro(valor){
     //tento converter a inteiro.
    //se for um inteiro nao lhe afeta, se não for tenta convertelo
    valor = parseInt(valor)

     //Comprovo se é um valor numérico
     if (isNaN(valor)) {
          //entao (nao e numero) devuelvo el valor cadena vacia
          return ""
     }else{
          //No caso contrario (Se for um número) devolvo o valor
          return valor
     }
}
```

Formulário

Vemos o formulário que necessitaremos para colocar o campo de texto. É um formulário como outro qualquer, o único detalhe é que nos preocupamos por lhe dar nome tanto ao formulário em si como ao campo de texto. Posteriormente, utilizaremos esses nomes para referirmos aos elementos mediante Javascript.

Também temos um campo de formulário do tipo botão, que serve neste caso para indicar que quando se clique, chamará à função validarFormulario(). Para indicar isto se utiliza o atributo onclick do campo botão e entre aspas podemos ver o que queremos que se execute, neste caso a função indicada.

```
<form name=formul>
<input type=text name=texto>
<input type=button value=validar onclick="validarFormulario()">
</form>
```

Função validarFormulario()

Esta função extrai o dado do campo de texto e o passa à função validarInteiro(), que nos devolverá um valor que teremos que colocar de novo no campo de texto. Para acessar ao formulário utilizamos a hierarquia de objetos do navegador, que para quem não sabe, é um conjunto de objetos que fazem referência a todos os elementos da página.

O acesso aos elementos da página se realiza começando no objeto window, que é o primeiro da hierarquia. Logo, continua pelo objeto document -que salva todo o documento- e em nosso exemplo, baixaremos ao formulário para poder acessar definitivamente ao campo de texto, que era onde queríamos chegar. Com este esquema:

window.document.formul.texto

Observamos que se utilizam os nomes do formulário e o campo que colocamos no atributo name das etiquetas HTML destes elementos.

Todos os campos de texto têm uma propriedade value que é onde se salva o texto que leva escrito dentro. De modo que se quisermos acessar ao que tem escrito o campo de texto escreveremos isto:

window.document.formul.texto.value

Agora que sabemos tudo o que foi dito anteriormente deveríamos entender perfeitamente o código da função.

```
function validarFormulario(){
    //extraimos o valor do campo
    textoCampo = window.document.formul.texto.value
    //o validamos como inteiro
    textoCampo = validarInteiro(textoCampo)
    //colocamos o valor de novo
    window.document.formul.texto.value = textoCampo
}
```

Conclusão

Com tudo isto completamos o exercício. Podemos ver como funcionaria a página inteira para observar os resultados finais.

Por si só pode ser que não tenha um resultado muito produtivo este exemplo, porém pode ser um início para começar a validar formulários mais complexos. Com um pouco de imaginação e esforço podemos fazer funções que validem outros campos do formulário para ver se o que contém são textos, se são suficientemente longos ou curtos, validar números com decimais, etc. Tudo isso se faria de forma parecida como vimos antes, acrescentando o código à funão validarFormulario() e talvez construindo algumas funções acessórias adicionais como validarInteiro().

Esperamos que tenha bom proveito, a pesar de ser pouco.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Exemplos de funcionamento da classe String

Agora vamos ver uns exemplos sobre como se utilizam os métodos e propriedades do objeto String.

Exemplo de strings 1

Vamos escrever o conteúdo de um string com um caractere separador ("-") entre cada um dos caracteres do string.

```
var meuString = "Ola Amigos"
var result = ""

for (i=0;i<meuString.length-1;i++) {
   result += meuString.charAt(i)
   result += "-"
}
result += meuString.charAt(meuString.length - 1)

document.write(result)
```

Primeiro, criamos duas variáveis, uma com o string a percorrer e outra com um string vazio, onde salvaremos o resultado. Logo, fazemos um loop que percorre desde o primeiro até o penúltimo caractere do string -utilizamos a propriedade length para conhecer o número de caracteres do string- e em cada iteração colocamos um caractere do string seguido de um caractere separador "-". Como ainda nos resta o último caractere por colocar, o colocaremos

na seguinte linha depois do loop. Utilizamos a função charAt para acessar às posições do string. Finalmente imprimimos na página o resultado.

Podemos ver o exemplo em funcionamento em uma página a parte.

Exemplo de strings 2

Vamos fazer um script que rompa um string em duas metades e as imprima por tela. As metades serão iguais, sempre que o string tiver um número de caracteres par. No caso de que o número de caracteres seja ímpar não se poderá fazer a metade exata, porém partiremos o string o mais aproximado à metade.

```
var meuString = "0123456789"
var metade1,metade2

posicao_metade = meuString.length / 2

metade1 = meuString.substring(0,posicao_metade)
metade2 = meuString.substring(posicao_metade,meuString.length)

document.write(metade1 + "<br>" + metade2)
```

As duas primeiras linhas servem para declarar as variáveis que vamos utilizar e iniciar o string a partir. Na seguinte linha encontramos a posição da metade do string.

Nas duas seguintes linhas é onde realizamos o trabalho de colocar em uma variável a primeira metade do string e na outra a segunda. Para isso, utilizamos o método substring passando-lhe como início e fim no primeiro caso desde 0 até a metade e no segundo desde a metade até o final. Para finalizar imprimimos as duas metades com uma quebra de linha entre elas.

Podemos ver o exemplo em funcionamento em uma página a parte.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Exemplo de funcionamento de Date

Neste exemplo vamos criar duas datas, uma com o instante atual e outra com data do passado. Em seguida, imprimiremos as duas e extrairemos seu ano para imprimi-lo também. Logo, atualizaremos o ano de uma das datas e voltaremos a escreve-la com um formato mais legível.

```
//nestas linhas criamos as datas
minhaDataaAtual = new Date()
minhaDataPassada = new Date(1998,4,23)

//nestas linhas imprimimos as datas.
document.write (minhaDataAtual)
document.write ("<br>")
document.write (minhaDtaPassada)
```

```
//extraímos o ano das duas datas
anoAtual = minhaDataAtual.getFullYear()
anoPassado = minhaDataPassada.getFullYear()

//Escrevemos em ano na página
document.write("<br>O ano atual é: " + anoAtual)
document.write("<br>O ano passado é: " + anoPassado)

//mudamos o ano na data atual
minhaDataAtual.setFullYear(2005)

//extraímos o dia, mês e ano
dia = minhaDataAtual.getDate()
mes = parseInt(minhaDataAtual.getMonth()) + 1
ano = minhaDataAtual.getFullYear()

//escrevemos a data em um formato legível
document.write ("<br>")
document.write (dia + "/" + mes + "/" + ano)
```

Há que destacar um detalhe antes de terminar, é que o número do mês pode começar desde 0. Pelo menos no Netscape com o qual realizamos as provas começava o mês em 0. Por esta razão somamos um ao mês que devolve o método getMonth.

Existem mais detalhes para destacar, pois é que no Netscape o método getFullYear() devolve os anos transcorridos desde 1900, com o qual ao obter o ano de uma data de, por exemplo, 2005, indica que é o ano 105. Para obter o ano completo temos a nossa disposição o método getFullYear() que devolveria 2005 da mesma forma que em Netscape e Internet Explorer.

Muita atenção no trabalho com datas em distintas plataformas, visto que poderia ser problemático o fato de oferecerem distintas saídas aos métodos de manejo de datas, dependendo sempre da marca e versão de nosso navegador.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Link aleatório Javascript

Vamos criar um efeito típico em páginas web que consiste em um link que nos levará a um site escolhido de forma aleatória. Para termos um idéia exata podemos ver o exemplo em funcionamento.

Para isso vamos utilizar Javascript. Embora por algumas razões não seja a melhor linguagem para fazer este exercício, sim que vai ser extremamente simples e acreditamos que também será instrutivo para os leitores.

Para começar, vamos criar um array com os diferentes sites para onde poderia nos conduzir nosso link. Temos que definir este array, logicamente, dentro de um bloco <script> na própria página web. A razão pela qual Javascript não é a melhor linguagem para este exercício é justamente esta, que temos que escrever na página todos os possíveis endereços e um usuário avançado poderia ler o código da página e encontrar todas as opções escritas no array.

Esa declaração do array seria algo parecido a isso:

```
var enderecos = new Array("http://www.terra.com","http://www.google.com","http://www.yahoo.com")
```

Como pode ser visto, na mesma linha na qual se declara o array se introduzem os valores de cada um de seus campos, utilizando o método rápido de declaração e de preenchimento de arrays em Javascript. Quanto mais valores escrevermos, mais aleatório será o exercício, podendo colocar mais links sem ter que editar o resto do código do programa. Em nosso exemplo completo temos uma lista muito maior de links.

Continuamos colocando o link que se apresentará como "Link Aleatório", que nos levará a um site aleatório, dentro das possibilidades.

```
<a href="javascript:linkAleatorio()">Link Aleatorio</a>
```

Como vemos, o link se encarrega de chamar a uma função que será a que vai extrair uma URL do array anterior e nos transferir a este lugar. A função terá esta forma:

```
function linkAleatorio(){
   aleat = Math.random() * enderecos.length
   aleat = Math.floor(aleat)
   window.location=enderecos[aleat]
}
```

Como se pode ver, o primeiro que faz a função é obter um valor aleatório entre 0 e "enderecos.length", que é o número de URLs de nosso array. Se mudarmos o número de URLs do array este script continuará funcionando perfeitamente, porque os limites se extraem diretamente da propriedade length do array que contém os endereços.

Para obter esse número aleatório se utiliza o método random da classe Math, que devolve um número entre 0 e 1. Ao multiplica-lo pelo número de posições do array obtemos um número entre 0 e o número de posições do array. Porém este número está em um número flutuante, ou seja, é um número decimal, que não nos serve como índice de um array. Por isso lhe aplicamos o método floor, também do objeto Math, para obter a parte inteira deste número.

Por último se atualiza a propriedade location do objeto window com o valor do array na posição aleatória, o que faz com que o navegador se dirija à página aleatória, dentro das distintas possibilidades.

Exemplo completo

Para ver de maneira global este exercício, transcrevemos aqui todo o código utilizado.

```
<html>
<head>
   <title>Link Aleatorio</title>
<script>
   var enderecos = new Array("http://www.terra.com.br", "http://www.google.com.br", "http://jbonline.terra.com.br",
"http://www.lycos.com", "http://br.yahoo.com", "http://www.altavista.com", "http://www.hotbot.com",
"http://www.buscopio.com", "http://oglobo.globo.com", "http://www.excite.com", "http://br.cade.yahoo.com",
"http://www.mercadolivre.com.br", "http://br.weather.com", "http://www.buscape.com.br", "http://www.msn.com",
"http://www.astrolabio.net")
   function linkAleatorio(){
      aleat = Math.random() * enderecos.length
      aleat = Math.floor(aleat)
```

```
    window.location=enderecos[aleat]
  }
</script>
</head>

<body>
<a href="javascript:linkAleatorio()">LinkAleatorio</a>
</body>
</html>
```

Se quiser ver o exemplo em funcionamento, clique aqui.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Geração de números aleatórios Javascript

Em Javascript dispomos da classe Math, muito útil quando queremos fazer cálculos matemáticos de certa complexidade. Tal classe está explicada e documentada em um capítulo do manual de Javascript II. Para os que necessitarem, também temos explicações do que são as classes e os objetos.

Neste workshop de Javascript vamos construir uma simples função para criar um número aleatório, entre um mínimo e um máximo, que poderemos utilizar logo em outros scripts mais complexos.

Aqui temos o código que usa o método random da classe Math para obter um número aleatório com Javascript.

```
var aleatorio = Math.random()
```

Assim criamos uma variável aleatória à que atribuímos o resultado de executar o método random da classe Math. O número aleatório que obtemos sempre estará compreendido entre 0 e 1.

Se desejarmos obter um número aleatório em outra categoria, poderemos consegui-lo com um pouco de matemática e com a classe Math. Para ilustrar isso vamos fazer uma função que devolve um número aleatório compreendido em um intervalo. O intervalo o recebe como parâmetro com duas variáveis, uma para o limite pela parte inferior e outra para o limite pela parte superior.

```
function aleatorio(inferior,superior){
   numPossibilidades = superior - inferior
   aleat = Math.random() * numPossibilidades
   aleat = Math.floor(aleat)
   return parseInt(inferior) + aleat
}
```

A função que fizemos é muito simples, porém funciona perfeitamente para todos os tipos de intervalos que possamos passar, tanto com números positivos como negativos. O primeiro que fazemos é obter o número de possibilidades diminuindo ao limite superior o inferior. Logo, multiplicamos tal numero de possibilidades pelo número aleatório obtido (que está entre 0 e 1), com o qual obtemos um número aleatório entre 0 e o número de possibilidades.

O número tem um monte de decimais, e neste exemplo desejamos obter um número inteiro, sem decimais. Por isso, logo utilizamos o método round() da clase Math, que nos dá o inteiro mais próximo. Como o número ainda está entre 0 e o número de possibilidades temos que somar o limite inferior, com o que estará dentro da categoria que desejarmos. Este último valor será em que devolva a função.

Esta função pode-se ver em funcionamento em uma página a parte. No exemplo, construímos um pequeno formulário que podemos preencher com o mínimo e o máximo e quando apertamos sobre o botão se mostrará o valor aleatório no campo abaixo de tudo.

Um exemplo do que podemos fazer com um número aleatório pode ser criar um link aleatório em uma página web. Podemos vê-lo no exemplo Link aleatório em Javascript. Ademais, neste exemplo cria-se o número aleatório de maneira ligeiramente distinta a como vimos agora, o que pode ser interessante para aprender melhor a usar os métodos da classe Math.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Comprovar se as senhas são iguais

Em um formulário de registro de usuários é muito habitual que desejemos incluir um campo chave, que o usuário poderia utilizar para acessar aos serviços da web onde está se registrando ou para atualizar mais adiante a informação introduzida.

Para realizar um formulário onde se escreva a senha, o normal é apareçam dois campos para introduzir a mesma senha e que o usuário que a introduza esteja forçado a escrever a senha duas vezes, sendo as duas senhas iguais. Isto nos ajuda a que o usuário não se engane ao escrever a senha por uma falha na datilografia, já que é complicado que erre duas vezes ao escrever a senha.

Com Javascript podemos fazer uma comprovação -no cliente- para ver se foi introduzida a mesma senha nos dois campos e se for assim, mandar o formulário ao servidor ou fazer aquelas ações necessárias para continuar com o registro desse visitante. No caso de que as duas senhas sejam diferentes, o usuário deve ser informado da situação para que volte a introduzir a senha desejada corretamente.

Comprovar se dois campos de formulário são iguais

É uma ação bastante simples. Simplesmente devemos extrair os dois valores salvos nos campos do formulário onde foi escrita a senha e a repetição da senha.

Logo, com um enunciado if, podemos comprovar se estes dois dados são o mesmo ou não e fazer as sentenças necessárias em cada caso.

A função poderia ter uma forma como esta:

```
function comprovarSenha(){
   senha1 = document.f1.senha1.value
   senha2 = document.f1.senha2.value

   if (senha1 == senha2)
      alert("As duas senhas são iguais...\nRealizaríamos as ações do caso positivo")
```

```
   else
     alert("As duas senhas são diferentes...\nRealizaríamos as ações do caso negativo")
}
```

> **Nota:** Se desejarmos mandar o formulário ao servidor depois de ter visto que as duas senhas são iguais, ou seja, se depois da comprovação queremos submeter o formulário, poderíamos executar o método submit() do objeto form.
>
> ```
> document.f1.submit()
> ```

Observe que o formulário onde estamos acessando pela hierarquia de objetos do navegador chama-se "f1". Isso quer dizer que a etiqueta <FORM> do formulário onde estão os dois campos tem o atributo name="f1". Se nosso formulário se chama de outra maneira teríamos que mudar as linhas onde se acessa aos dois campos da senha, substituindo "f1" pelo nome do nosso formulário. Da mesma forma, os dois campos onde se escrevem as senhas em nosso exemplo se supõem que se chamam "senha1" e "senha2", se não for assim, também teríamos que mudar esse nome para o que foi colocado.

Para ver mais claramente, temos aqui o código do formulário onde se encontram os campos e seus nomes.

```
<form action="" name="f1">
Contraseha: <input type="password" name="senha1" size="20">
<br>
Repete contrasenha: <input type="password" name="senha2" size="20">
<br>
<input type="button" value="Comprovar se são iguais" onClick="comprovarSenha()">
</form>
```

Conclusão

Dada sua simplicidade, este exemplo não deveria supor muito problema. Resta-nos ver o código do exemplo completo e, se desejarmos, acessar a uma página onde se mostra o exercício em funcionamento.

```
<html>
<head>
<title>Validar se a senha e a repetição da senha são iguais</title>
<script>
function comprovarSenha(){
   senha1 = document.f1.senha1.value
   senha2 = document.f1.senha2.value

   if (senha1 == senha2)
     alert("As duas senhas são iguais...\nRealizaríamos as ações do caso positivo")
   else
     alert("As duas senhas são diferentes...\nRealizaríamos as ações do caso negativo")
}
</script>
</head>

<body>

<h1>Validar se a senha e a repetição da senha são iguais</h1>
<br>
Escreva uma senha duas vezes, uma em cada campo, e clique o botão. Javascript lhe dirá se as duas são iguais.

<br>
<form action="" name="f1">
```

```
Contrasenha: <input type="password" name="senha1" size="20">
<br>
Repita contrasenha: <input type="password" name="senha2" size="20">
<br>
<input type="button" value="Comprovar se são iguais" onClick="comprovarSenha()">

</form>

</body>
</html>
```

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Ex. de trabalho com formulários. Calculadora simples

Para ilustrar um pouco o trabalho com formulários, vamos realizar um exemplo prático. Pode ser que algumas coisas das quais vamos tratar, fiquem um pouco no ar por não terem sido explicadas detalhadamente antes, mas certamente nos serve para estarmos por dentro de como se trabalha com formulários e as possibilidades que temos.

Exemplo de calculadora simples

Neste exemplo vamos construir uma calculadora, embora bastante simples, que permita realizar as operações básicas. Para fazer a calculadora vamos realizar um formulário no qual vamos colocar três campos de texto, os dois primeiros para as parcelas e um terceiro para o resultado. Ademais haverá uns botões para fazer as operações básicas.

O formulário da calculadora pode ser visto aqui.

```
<form name="calc">
<input type="Text" name="parcela1" value="0" size="12">
<br>
<input type="Text" name="parcela2" value="0" size="12">
<br>
<input type="Button" name="" value=" + " onclick="calcula('+')">
<input type="Button" name="" value=" - " onclick="calcula('-')">
<input type="Button" name="" value=" X " onclick="calcula('*')">
<input type="Button" name="" value=" / " onclick="calcula('/')">
<br>
<input type="Text" name="resultado" value="0" size="12">
</form>
```

Mediante uma função vamos acessar aos campos do formulário para buscar as parcelas em duas variáveis. Os campos de texto têm uma propriedade chamada value que é onde podemos obter o que têm escrito nesse momento. Mais tarde recorreremos a função eval() para realizar a operação. Colocaremos por último o resultado no campo de texto criado em terceiro lugar, utilizando também a propriedade value do campo de texto.

Chamamos à função que realiza o cálculo (que podemos ver a seguir) apertando os botões de cada uma das operações. Tais botões podem ser vistos no formulário e contém um atributo onclick que serve para especificar as sentenças Javascript que desejarmos que se executem quando o usuário clicar sobre ele. Neste caso, a sentença a executar é uma chamada à função

calcula() passando como parâmetro o símbolo ou a parcela da operação que desejarmos realizar.

O script com a função calcula()

```
<script>
function calcula(operacao){
   var parcela1 = document.calc.parcela1.value
   var parcela2 = document.calc.parcela2.value
   var result = eval(parcela1 + operacao + parcela2)
   document.calc.resultado.value = result
}
</script>
```

A função eval(), lembramos, que recebia um string e o executava como uma sentença Javascript. Neste caso, irá receber um número que concatenado a uma operação e outro número será sempre uma expressão aritmética que eval() solucionará perfeitamente.

Podemos ver o exemplo da calculadora em funcionamento.

O acesso a outros elementos dos formulários se faz de maneira parecida a respeito da hierarquia de objetos, entretanto, como cada elemento tem suas particularidades as coisas que poderemos fazer com eles diferirão um pouco. Isto será visto mais adiante.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Enviar ao navegador a outra página se não tiver Javascript

Imaginemos uma página que, para se ver bem, necessite ter habilitada a possibilidade de executar scripts em Javascript e que, se não tiver habilitado Javascript, não funcionasse bem e não pudesse mostrar todos os conteúdos.

Em um caso como este seria muito útil dispor de uma função que detecte se está habilitado ou não Javascript para, no caso de que não seja assim, se envie ao navegador o outro endereço.

Pois esta função que detecta se está ou não habilitado Javascript não se pode fazer tão ricamente, pelo menos utilizando Javascript. Imagine que você não dispõe de Javascript, o navegador não poderia então executar essa função e nunca detectaria que não há Javascript.

Por sorte temos um enunciado <NOSCRIPT></NOSCRIPT> que nos serve para indicar ações a tomar no caso de que não esteja habilitado Javascript.

Utilizando essa etiqueta podemos colocar um link para que se veja só nos navegadores que não têm Javascript:

```
<NOSCRIPT>
Seu navegador não suporta Javascript. <a href="no_javas.html">Entre em uma página que não o utiliza</a>
</NOSCRIPT>
```

Podemos ainda ir um passo mais longe e utilizar a etiqueta META tipo "Refresh" para que o navegador se refresque automaticamente e se dirija a outra página que não inclua programação em Javascript.

É uma opção muito mais interessante, porque não temos que esperar que o visitante clique no link e assim nos certificamos que, apesar de alguém não encontrar o link, o navegador o encaminha corretamente.

```
<NOSCRIPT>
<META HTTP-EQUIV="Refresh" CONTENT="3;URL=no_javas.html">
</NOSCRIPT>
```

Obviamente, isto só funcionará se nosso navegador aceita este tipo de etiquetas de refresh automático, embora os navegadores mais habituais sim que as aceitam.

Aliás, a etiqueta de refresh deve ser colocada no cabeçalho (Dentro de <HEAD></HEAD>). O primeiro dado do valor de refresh é o tempo de espera antes de se atualizar em segundos, neste caso 3 segundos. O segundo dado é o endereço ao qual queremos enviar o navegador, neste caso no_javascript.html.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Confirmação de envio de formulário

Este artigo amplia uma dúvida de um usuário de CriarWeb.com. A pergunta em concreto que nos realizou o usuário era como fazer um formulário que, ao envia-lo, nos pergunte se realmente se deseja enviar.

A pergunta em concreto era a seguinte:

Estou fazendo um formulário e desejo que, ao envia-lo, me mostre uma janela de confirmação de envio do formulário, dessas que têm um botão de aceitar e outro de cancelar. Então, se se aceita o envio, se enviaria o formulário.... se não se aceita, o formulário não seja enviado.

Resposta

Isto tem muito a ver com o tema de tratamento de formulários. A resposta baseia sua maior técnica no fato de substituir o botão de submit por um botão normal. Com o botão normal não se envia o formulário diretamente e sim, chama a uma função que realiza a confirmação e, no caso positivo, envia o formulário.

O botão que colocaríamos no formulário em substituição do botão de submit seria o seguinte:

```
<input type=button onclick="pergunta()" value="Enviar">
```

Observemos que o botão tem definida uma ação no momento em que se clica. A ação em concreto faz com que se execute a função pergunta(), que será a que realize a confirmação e envie o formulário no caso positivo. Seu código pode ser visto a seguir.

```
<script language="JavaScript">
function pergunta(){
   if (confirm('Tem certeza que quer enviar este formulário?')){
      document.seuformulario.submit()
   }
}
</script>
```

A caixa confirm devolve true ou false dependendo de se se clica o botão de aceitar ou cancelar. Esse valor se utiliza em um enunciado if para decidir se se envia o formulário, com seu método submit(), ou não se faz nada.

O código completo de uma página que realiza esta tarefa em um formulário é o seguinte:

```
<html>
<head>
   <title>Confirmação de envio de formulário</title>
<script language="JavaScript">
function pergunta(){
   if (confirm('Tem certeza que quer enviar este formulário?')){
      document.seuformulario.submit()
   }
}
</script>
</head>

<body>
<form name=tuformulario action="http://www.criarweb.com">
<input type="text" name="qualquercampo">
<input type=button onclick="pergunta()" value="Enviar">
</form>

</body>
</html>
```

Pode-se ver o exercício em funcionamento aqui.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Javascript para se posicionar em um select

Trata-se de um script para se posicionar em um elemento de um select, ou seja, para conseguir que, clicando umas teclas do teclado que poderiam corresponder com as primeiras letras de um elemento do select, o elemento selecionado do tal select seja aquele que corresponda com as letras pulsadas.

É uma descrição um pouco longa, mas na verdade o efeito é simples. Nos select das páginas web, ao clicar uma tecla, o select se move ao primeiro elemento que tem como inicial essa tecla. Entretanto, se há muitos elementos no select, o usuário pode achar que essa ajuda fica um pouco curta, já que teria que, logo clicar a inicial do elemento buscado, repassar todos os elementos que começam por essa letra até encontrar o que busca. O presente exemplo melhora essa função de busca nos select, já que permite realizar o clique de várias teclas seguidas e vai mostrando aquele elemento que começa por todas as letras que vão sendo clicadas (uma atrás da outra) até que se aperta a tecla Enter, momento no qual se supõe que encontramos o elemento adequado e queremos continuar com o preenchimento de outros campos do formulário.

Neste exemplo foi criado um select com os nomes de diferentes países. Se, por exemplo, queremos buscar o país Estados Unidos, nos selects normais podemos clicar a E (inicial de Estados Unidos) e buscar entre todos os países até que aparece o que queremos. Porém, com a implementação deste script poderemos clicar a E, com o que se posicionará no primeiro país que comece por E (que não tem porque ser o que buscamos, na prática será Equador). Logo, podemos clicar a letra S, com o qual se mostrará Espanha, que não é o que buscamos. Mais

tarde clicaria-se a T, aparecendo Estônia e, por último, ao clicar a tecla A, já apareceria o elemento que buscávamos, ESTAdos Unidos.

O script

O script encontra-se comentado dentro do próprio código, para que possa ser compreendido facilmente, ou pelo menos suas bases. Basicamente, utiliza-se o evento de teclado onKeyPress no elemento select dos países, de modo que, quando se clique uma tecla, se tivermos o foco no select, se chamará a uma função que se encarregará de fazer o trabalho mais duro.

Tal trabalho consiste em pegar a tecla que foi clicada e salva-la em uma estrutura de dados, ademais de selecionar o elemento mais próximo ao valor atual da estrutura de dados. Por último, clica-se a tecla enter, deixa-se o select com o último valor selecionado e passa-se o foco ao seguinte elemento do formulário para que o usuário continua preenchendo-o.

```
<script language="JavaScript1.2">
var digitos=10 //quantidade de digitos buscados
var ponteiro=0
var buffer=new Array(digitos) //declaração do array Buffer
var cadeia=""

function buscar_op(obj,objfoco){
  var letra = String.fromCharCode(event.keyCode)
  if(ponteiro >= digitos){
     cadeia="";
     ponteiro=0;
   }
  //se se pressiona a tecla ENTER, apago o array de teclas pressionadas e salto a outro objeto...
  if (event.keyCode == 13){
     apagar_buffer();
     if(objfoco!=0) objfoco.focus(); //evita foco a outro objeto se objfoco=0
   }
  //senao busco a cadeia teclada dentro do combo...
  else{
     buffer[ponteiro]=letra;
     //salvo na posicao ponteiro a letra teclada
     cadeia=cadeia+buffer[ponteiro]; //armo uma cadeia com os dados que vao ingressando ao array
     ponteiro++;

     //barro todas as opcoes que contem o combo e comparo a cadeia...
     for (var opcombo=0;opcombo < obj.length;opcombo++){
       if(obj[opcombo].text.substr(0,ponteiro).toLowerCase()==cadeia.toLowerCase()){
       obj.selectedIndex=opcombo;
       }
     }
   }
  event.returnValue = false; //invalida a acao de clique de tecla para evitar busca do primeiro caractere
}

function apagar_buffer(){
  //inicia a cadeia buscada
   cadeia="";
   ponteiro=0;
}
</script>

<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF" text="#000000">
```

```
<table width="544" border="0" cellpadding="0" cellspacing="0">
<tr>
<td width="89" height="29"></td>
<td width="114"></td>
<td width="26"></td>
<td width="315"></td>
</tr>
<tr>
<td height="19"></td>
<td valign="top">
<select name="combo1" onKeypress=buscar_op(this,text2) onblur=borrar_buffer() onclick=borrar_buffer()>
   <option>Argentina</option>
   <option>Australia</option>
   <option>Bolivia</option>
   <option>Brasil</option>
   <option>Canada</option>
   <option>Colombia</option>
   <option>Dinamarca</option>
   <option>Estados Unidos</option>
   <option>Estonia</option>
   <option>Austria</option>
   <option>Bulgaria</option>
   <option>Chile</option>
   <option>Espanha</option>
   <option>China</option>
   <option>Costa Rica</option>
   <option>Croacia</option>
   <option>Equador</option>
</select>
</td>
<td></td>
<td></td>
</tr>
<tr>
<td height="18"></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td height="28"></td>
<td colspan="2" valign="top">
<input type="text" name="text2">
</td>
<td></td>
</tr>
<tr>
<td height="58"></td>
<td></td>
<td></td>
<td></td>
</tr>
</table>
</body>
</html>
```

Esperamos que vocês possam entender o script e utiliza-lo em suas páginas web.

*Artigo por **Ignacio Rodriguez***

Inibir um campo texto de formulário com Javascript

Desta vez faremos um workshop muito rápido e simples com Javascript para fazer com que um campo de formulário de tipo texto encontre-se inibido, ou seja, que não possamos clicar encima dele para editar seu conteúdo.

Focus e Blur

A maneira de fazê-lo requer o conhecimento de dois conceitos habituais de Javascript relacionados com o foco da aplicação.

O conceito focus, está relacionado com ganhar foco da aplicação. O método **focus()**, que têm os campos de texto e outros elementos de formulário, serve para outorgar o foco da aplicação a esse elemento. O administrador de evento **onfocus** aparece quando um elemento ganha o foco da aplicação.

O conceito blur, está associado a perder o foco da aplicação. O método **blur()** serve para que os elementos de formulário percam o foco e o administrador de eventos **onblur** se ativa quando o elemento a que o apliquemos perda o foco da aplicação.

O exercício

Para inibir um campo de formulário podemos fazer com que o usuário nunca possa clicar nesse elemento ou então, que se chegar a clicar seja repelido imediatamente. Para isto o único que temos que fazer é retirar o foco de um elemento quando tiver ganhado.

Nós utilizaremos o evento onfocus para detectar o instante no qual o elemento ganha o foco e nesse momento faremos uso do método blur() para retirar o foco.

O código é extremamente simples para tanta explicação:

```
<form>
<input type="text" value="122" onfocus="this.blur()">
</form>
```

O único detalhe que vale a pena assinalar é o uso da palavra this, que faz referência ao elemento onde se está utilizando, nesse caso o campo de texto. this.blur() seria uma simples chamada ao método blur() no elemento de formulário onde está colocada.

Pode ser visto em funcionamento aqui:

122

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Camadas com Internet Explorer 5, 6, Netscape 6, 7 e Opera

Todos os programadores tiveram alguma vez que enfrentar aos problemas de compatibilidade que apresentam os navegadores desenvolvidos por Netscape e Microsoft.

Devido a que tinham dois **DOM** (Modelo de objetos do documento) diferentes, a forma de acessar de um e outro eram diferentes, provocando a necessidade de utilizar diferentes truques para conseguir que os scripts fossem compatíveis com os dois navegadores.

Estes problemas se solucionaram em certa medida desde a adoção por parte das companhias do DOM definido pelo W3C, ao poder escrever o mesmo código e que fosse compatível com os dois navegadores. O único problema que apresenta é que o script não será compatível com as versões anteriores de tais navegadores, provocando no caso de necessitar compatibilidade para atrás, a necessidade de adicionar o código especifico para as versões anteriores que se desejem implementar.

Uma das funções chave para trabalhar com dois navegadores ao mesmo tempo, é a função **getElementById(elemento)** que recebe por parâmetro o nome de um elemento da página, e devolve o objeto correspondente. Através desta função, se poderão acessar a todas as propriedades do objeto. Esta função está definida no padrão do W3C.

Uma vez tendo o objeto, bastará acessar à propriedade style, para através dela, acessar a todos os estilos definidos no elemento.

A vantagem de utilizar este método, é que não necessitaremos distinguir que navegador está sendo utilizado, já que funciona para os dois.

*Artigo por **Carlos Cuenca Díaz***

Mostrar e ocultar camadas com IE 5,6 NS 6,7

O primeiro a fazer é definir as funções de mostrar e ocultar. Estas duas funções, receberão por parâmetro, o nome da camada que queremos mostrar ou ocultar, a seguir, mediante a função getElementById, acessaremos às propriedades da camada, e através de style aos estilos que nos definem se a camada está visível ou não (propriedade visibility), ativando-a ou desativando-a segundo a função.

> **Referência:** Falamos sobre a função getElementById() no artigo Camadas com Internet Explorer 5, 6, Netscape 6, 7 e Opera

```
<script language="Javascript">
function mostrar(nomeCamada){
document.getElementById(nomeCamada).style.visibility="visible";
}
function ocultar(nomeCamada){
document.getElementById(nomeCamada).style.visibility="hidden";
}
</script>
```

A seguir é necessário definir as camadas, e chamar em algum momento às funções definidas. Utilizaremos os eventos onMouseOver e onMouseOut da camada 1 para mostrar e ocultar a camada 2.

```
<div id="camada1" style="position:absolute;width:100;height:100;top:100;left:100;background-color:blue"
onmouseout="ocultar('camada2')" onmouseover="mostrar('camada2')">Camada 1</div>
<div id="camada2" style="position:absolute;width:100;height:100;top:100;left:200;background-
```

```
color:red;visibility:hidden">Camada 2</div>
```

Ao passar por cima da camada 1 se mostrará a 2, e ao sair da camada 1, se ocultará a dois.

Exemplo completo:

```
<html>
<head>
<title>Untitled</title>
<script language="Javascript">
function mostrar(nomeCamada){
document.getElementById(nomeCamada).style.visibility="visible";
}
function ocultar(nomeCamada){
document.getElementById(nomeCamada).style.visibility="hidden";
}
</script>
</head>
<body>
<div id="camada1" style="position:absolute;width:100;height:100;top:100;left:100;background-color:blue"
onmouseout="ocultar('camada2')" onmouseover="mostrar('camada2')">Camada 1</div>
<div id="camada2" style="position:absolute;width:100;height:100;top:100;left:200;background-
color:red;visibility:hidden">Camada 2</div>
</body>
</html>
```

Podemos ver o exemplo em funcionamento.

*Artigo por **Carlos Cuenca Díaz***

Movimento de Camadas com IE 5,6 NS 6,7

Para deslocar uma camada pela tela, é necessário modificar os atributos Top e Left.

> **Referência:** Falamos sobre a função getElementById(), necessária para entender a maneira de trabalhar neste artigo, no documento Camadas com Internet Explorer 5, 6, Netscape 6, 7 e Opera

Top define a posição vertical da camada desde a parte superior da tela

Left define a posición horizontal da camada desde a parte esquerda da tela.

Ao modificar os valores de Top e Left conseguiremos o movimento da camada.

Por último, temos que redesenhar os valores que armazenam uma cadeia de texto com o valor e as unidades, por isso será necessário converter os valores que se armazenam dentro das propriedades antes de poder utiliza-los.

Por exemplo, um valor armazenado em top o left poderá ser 140px.

Por este motivo é necessário utilizar a função parseInt para poder converter os valores a números.

A seguir mostra-se um exemplo, que desloca uma camada 5 pixels cada vez que se clica o

botão:

A função recebe por parâmetro o nome da camada que se deseja mover, acessa mediante a função getElementById à propriedade de top, mediante ParseInt a converte a um inteiro, soma 5 unidades, e a seguir escreve o novo valor no estilo da camada.

```
<script language="Javascript">
function mover(nomeCamada){
valor=document.getElementById(nomeCamada).style.top;
numero=parseInt(valor);
numero+=5;
document.getElementById(nomeCamada).style.top=numero;
}
</script>
```

Exemplo completo:

```
<html>
<head>
<title>Untitled</title>
<script language="Javascript">
function mover(nomeCamada){
valor=document.getElementById(nomeCamada).style.top;
numero=parseInt(valor);
numero+=5;
document.getElementById(nomeCamada).style.top=numero;
}
</script>
</head>
<body>
<div id="camada1" style="position:absolute;width:100;height:100;top:100;left:100;background-color:blue">Camada
1</div>
<form name="meuform" action="#">
<input type="button" onclick="mover('camada1')" value="Mover Camada">
</form>
</body>
</html>
```

Ver o exemplo

*Artigo por **Carlos Cuenca Díaz***

Escritura nas Camadas com IE 5, 6, NS 6, 7

O texto que se encontra armazenado dentro de uma camadas, se encontra dentro da propriedade innerHTML. Esta propriedade, está implementada no Internet Explorer desde a versão 4, e Netscape o incorporou nas versões 6 e 7 apesar de que não se encontra definido no padrão do W3C.

> **Referência:** Este artigo oferece ajudas para a qual é necessário conhecer um pouco Javascript e o método getElementById(). Recomendamos consultar estes links:
>
> Programação em JavaScript
> Camadas com Internet Explorer 5, 6, Netscape 6, 7 e Opera

A propriedade armazena o código HTML que se corresponde com o que se visualiza dentro da camada, ou seja, se a camada contém " ola " a propriedade conterá " ola " entretanto, se a propriedade contém um link http://www.criarweb.com então a cadeia de texto conterá "http://www.criarweb.com, por isso poderá ser escrito qualquer tipo de conteúdo dentro da camada.

O seguinte exemplo, contém um formulário com uma caixa de texto, o usuário pode escrever qualquer texto ou código HTML, e ao clicar o botão, este se mostrará dentro da camada.

A função recebe por parâmetro o nome de uma camada e o formulário que contém a caixa de texto, salva o valor do quadro de texto dentro de uma variável e a seguir atribui à propriedade innerHTML da camada:

```
function escreveCamada(camada,formulario){
texto=formulario.caixa.value;
document.getElementById(camada).innerHTML=texto;
}
```

Exemplo completo:

```
<html>
<head>
<title>Untitled</title>
<script language="JavaScript">
function escreveCamada(camada,formulario){
texto=formulario.caixa.value;
document.getElementById(camada).innerHTML=texto;
}
</script>
</head>
<body>
<div id="minhacamada" style="position:absolute;width:100;height:100;top:100;left:100;background-color:yellow">
</div>
<form name="meuformulario" action="#">
Texto: <input type="text" name="caixa" size="50"> <input type="button"
onclick="escreveCamada('minhacamada',this.form)" value="escrever">
</form>
</body>
</html>
```

*Artigo por **Carlos Cuenca Díaz***

Como iluminar tabelas, células ou filas

Passo 1: Colocar este script no head <head> xxx </head>

```
<script>
function um(src,cor_entrada) {
   src.bgColor=cor_entrada;src.style.cursor="hand";
}
function dois(src,cor_default) {
   src.bgColor=cor_default;src.style.cursor="default";
}
</script>
```

Passo 2: Se o que você quer é que se ilumine uma célula observe o seguinte código:

```
<table border="0" cellspacing="0" cellpadding="0" bgcolor="#000000" align="center">
<tr>
<td>
<table border="0" cellspacing="1"cellpadding="0" align="center" width="278">
<tr bgcolor="#FFFFFF">
<td onMouseOver="um(this,'cccccc');" onMouseOut="dois(this,'FFFFFF');" align="center" width="100"
valign="middle"><font face="Arial, Helvetica, sans-serif" size="1">
PASSE POR CIMA</font></td>
<td width="95"> </td>
<td width="83"> </td>
</tr>
<tr bgcolor="#FFFFFF">
<td width="100"> </td>
<td width="95"> </td>
<td width="83"> </td>
</tr>
<tr bgcolor="#FFFFFF">
<td width="100"> </td>
<td width="95"> </td>
<td width="83"> </td>
</tr>
</table>
</td>
</tr>
</table>
```

PASSE POR CIMA

Passo 3: Se o que você quer é que se ilumine uma fila observe o seguinte código:

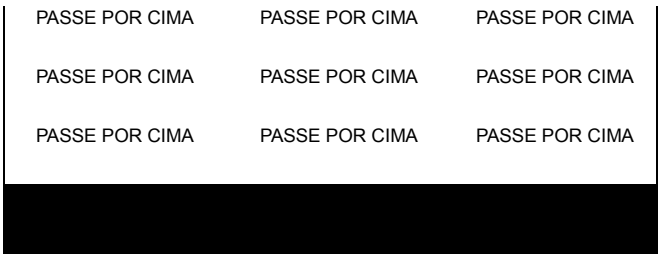
```
<table border="0" cellspacing="0" cellpadding="0" bgcolor="#000000" align="center" width="317">
<tr>
<td>
<table border="0" cellspacing="1" cellpadding="0" align="center" width="325">
<tr onMouseOver="um(this,'cccccc');" onMouseOut="dois(this,'FFFFFF');" bgcolor="#FFFFFF">
<td align="center" width="108" valign="middle" height="17">
<font face="Arial, Helvetica, sans-serif" size="1">PASSE POR CIMA</font></td>
<td width="114" height="17">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
<td width="99" height="17">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
</tr>
<tr bgcolor="#FFFFFF">
<td width="108"> </td>
<td width="114"> </td>
<td width="99"> </td>
</tr>
<tr bgcolor="#FFFFFF">
<td width="108"> </td>
<td width="114"> </td>
<td width="99"> </td>
</tr>
```

```
</table>
</td>
</tr>
</table>
```

PASSE POR CIMA PASSE POR CIMA PASSE POR CIMA

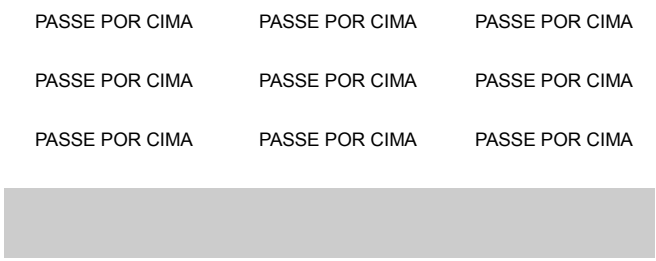
Passo 3: Se o que você quer é que se ilumine uma tabela completa observe o seguinte código:

```
<table border="0" cellspacing="1" cellpadding="0" bgcolor="#000000" align="center" width="317">
<tr>
<td>
<table border="0" cellspacing="1" cellpadding="0" align="center" width="325" height="62"
onMouseOver="um(this,'cccccc');" onMouseOut="dois(this,'FFFFFF');" bgcolor="#FFFFFF">
<tr>
<td align="center" width="108" valign="middle" height="17">
<font face="Arial, Helvetica,sans-serif" size="1">PASSE POR CIMA</font></td>
<td width="114" height="17">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
<td width="99" height="17">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
</tr>
<tr>
<td width="108">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
<td width="114">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
<td width="99">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
</tr>
<tr>
<td width="108">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
<td width="114">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
<td width="99">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
</tr>
</table>
</td>
</tr>
</table>
```



Paso 4: Se o que você quer é que se iluminem as bordas de uma tabela observe o seguinte código:

```
<table border="0" cellspacing="0" cellpadding="0" bgcolor="#CCCCCC" align="center" width="317">
<tr>
<td>
<table onMouseOver="um(this,'000000');" onMouseOut="dois(this,'CCCCCC');" border="0" cellspacing="1"
cellpadding="0" align="center" width="325" height="60">
<tr bgcolor="#FFFFFF">
<td align="center" width="108" valign="middle" height="17">
<font face="Arial, Helvetica,sans-serif" size="1">PASSE POR CIMA</font></td>
<td width="114" height="17">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
<td width="99" height="17">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
</tr>
<tr bgcolor="#FFFFFF">
<td width="108">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
<td width="114">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
<td width="99">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
</tr>
<tr bgcolor="#FFFFFF">
<td width="108">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
<td width="114">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
<td width="99">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASSE POR CIMA</font></div>
</td>
</tr>
</table>
</td>
</tr>
</table>
```



*Artigo por **Fabio Núñez Iturriaga***

Inibir radio button com Javascript

Veremos neste workshop como se pode desabilitar um elemento de formulário de tipo radio button. Com outras palavras, vamos ver a maneira de fazer com que, ao clicar um campo tipo radio de um formulário, não mude a opção escolhida por padrão no código HTML da página.

Pode-se ver o exemplo em funcionamento sob estas linhas. Observem que, ao clicar os radio button, não muda a opção escolhida inicialmente, que é a primeira.

○ Olá pessoal!
○ Aqui estamos
○ E você aí?

Detalhes prévios

Os campos de formulário tipo radio se manejam como um grupo. Na hierarquia de objetos do navegador ficam debaixo do objeto form e dentro de um array que toma o nome atribuído ao grupo de botões. Pode-se ver esta formação no artigo Controle de botões de radio em Javascript.

Solução

A maneira que nós implementamos de solucionar este assunto é definindo uma variável com o índice do array do botão de radio que tem que estar selecionado. Ademais, teremos uma função que se chamará ao clicar em qualquer botão de radio que se encarregará de selecionar o botão de radio por padrão, deste modo, embora selecionemos outro elemento do conjunto, se selecionará automaticamente o elemento marcado por padrão. Ademais, a função receberá o índice do botão de radio clicado e retirará o foco da aplicação de tal elemento.

Podemos ver o código a seguir:

```
<html>
<head>
   <title>Exemplo para desabilitar radio butons</title>
```

```
<script>
indice_marcado = 0
function desabilitar(formulario,idradio){
   formulario.meuradio[indice_marcado].checked = true
   formulario.meuradio[idradio].blur()
}
</script>
</head>

<body>
<h1>Exemplo para desabilitar radio butons</h1>

<form name="f1">
<input type="radio" name="meuradio" value="O que for" onclick="desabilitar(this.form,0)" checked> Olá pessoal!
<br>
<input type="radio" name="meuradio" value="outra coisa" onclick="desabilitar(this.form,1)"> Aqui estamos
<br>
<input type="radio" name="meuradio" value="mais coisas" onclick="desabilitar(this.form,2)"> E você aí?
</form>

</body>
</html>
```

Nos elementos de formulário de tipo radio button temos o manejador de eventos onclick que se chama quando se clica nesse botão de radio. Tal manejador de eventos chama a uma função passando-lhe o formulário onde estamos trabalhando e o índice do botão de radio atual, que começa em zero.

A função desabilitar(), definida no bloco de script do cabeçalho, contém duas sentenças. A primeira volta a colocar a seleção no botão de radio adequado, utilizando a propriedade checked do radiobutton, e a segunda retira o foco do elemento clicado, com o método blur().

Podemos ver o exemplo em funcionamento em uma página a parte.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Atualizar dois frames com um só link

Com o que já sabemos sobre o controle de frames, podemos realizar um exemplo para consolidar os conhecimentos. Trata-se de um exercício muito simples para conseguir que, ao clicar um link, se atualize a página contida em dois frames diferentes.

Como um link só serve para atualizar o conteúdo de um frame, necessitaremos executar umas sentenças javascript que sim nos permitam atualizar dois frames de uma vez.

Se não se entende o objetivo perseguido neste exemplo, podemos vê-lo em funcionamento em uma página a parte.

Comecemos vendo a declaração de frames, que não tem nenhuma complicação pois é simplesmente um código HTML que podemos aprender a programar nos artigos dedicados a frames do manual de HTML.

```
<html>
<head>
   <title>Exemplo de frames numero 1</title>
</head>
```

```
<frameset rows="50%,*">
   <frame name="frame1" src="pagina1.html" marginwidth="10" marginheight="10" scrolling="auto"
frameborder="0">
   <frame name="frame2" src="pagina2.html" marginwidth="10" marginheight="10" scrolling="auto"
frameborder="0">
</frameset>
</html>
```

Agora vejamos o código do primeiro dos frames, que é o que tem a função Javascript para controlar os frames.

```
<html>
<head>
   <title>Pagina 1</title>
<script language="JavaScript">
function atualiza_2_frames(){
   window.parent.frames[1].location="http://www.google.com"
   window.location="http://www.yahoo.com"
}
</script>
</head>
<body bgcolor="#ff9999">
<br>
<br>
<a href="javascript:atualiza_2_frames()">Atualiza dois frames com um so link</a>
</body>
</html>
```

Ao clicar o link se chama a uma função, colocada no cabeçalho da página, por comodidade e para evitar ter que escrever várias sentenças no atributo href do link.

A função, onde verdadeiramente está o miolo do exercício, é extremamente simples. A primeira sentença acessa ao frame colocado em segundo lugar (que tem o índice 1) e atualiza sua propriedade location, que é a URL da página que se está visualizando. Neste caso coloca a web de Google em tal frame, embora seja indiferente o que desejemos colocar e poderíamos ter situado um endereço com um caminho relativo ao documento atual.

Na segunda sentença acessamos diretamente à propriedade location do objeto window, porque desejamos atualizar o mesmo frame onde está colocado o script. Poderíamos ter utilizado uma sentença como a seguinte:

window.parent.frames[0].location=" http://www.yahoo.com "

Porém, neste caso não é necessário acessar à declaração de frames e logo ao frame 0 porque, como dizia, já estamos nele.

Por último vejamos o código do segundo frame, que não tem nada de especial.

```
<html>
<head>
   <title>Pagina 1</title>
</head>
<body bgcolor="#9999ff">
<br>
<br>
Este é o corpo do frame 2, que tem índice 1 no vetor de frames
</body>
</html>
```

Pode-se ver o exemplo em funcionamento em uma página a parte.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Calcular a idade em Javascript

Neste artigo vamos explicar uma função que calcula a idade de uma pessoa. Para isso recebe um string com a data de nascimento da pessoa e devolve o número de anos que tem. Estamos diante um exercício que ilustra muito bem o trabalho com datas em Javascript.

> **Referência:** Para aprender algo que nos sirva de base no cálculo de datas seria interessante ler o artigo Clase Date em Javascript.

O método de trabalho

Nós estamos pensando em receber uma data em formato português: algo como "12/10/1975", de tipo string. O primeiro será separar os diferentes valores de ano, mês, dia. Para isso, utilizamos o método split(), que pertence à classe String (tipo da data que vamos receber), que devolve um array com o valor de cada uma das partes da cadeia, utilizando como separador o caractere "/". Depois da separação, no array devolvido, deveríamos ter três campos, onde o primeiro (o de índice 0) salvará o dia, o segundo o mês e o terceiro o ano.

> **Referência:** Os métodos da classe String podem ser vistos no artigo Classe String em Javascript.

Vamos realizar a seguir algumas comprovações para certificarmos que as datas estejam corretas, ou seja, que temos um valor numérico como dia, outro como mês e outro como ano. Se não for assim, devolveremos false, que deveria se interpretar como que a função é incapaz de calcular a idade, porque a data de nascimento passada não é correta.

A seguir restaremos o número de anos da data atual, que poderiam ser 2007, com o número de ano da data de nascimento, que será algo como 1975. Neste caso, daria 32, porém nós vamos considerar 31, pois não sabemos se a suposta pessoa já fez anos no ano curso, ou não. Ou seja, hoje que é junho, se fez anos em março, essa pessoa já teria 32 anos, porém se faz anos em agosto, teria agora 31 anos.

Portanto, nosso próximo passo será saber em qual mês a pessoa fez anos e disso, poderíamos ter três casos.

1. Se o mês atual for menor que o mês de nascimento. Então, é porque ainda não cumpriu os anos ainda neste período anual. (Os anos, no exemplo anterior, seriam 31)
2. Se o mês atual for maior que o mês de nascimento, quer dizer que essa pessoa sim que já celebrou seu aniversário este ano. (Os anos, no exemplo anterior, seriam 32)
3. Se os dois meses forem iguais, temos que observar o dia de uma maneira similar a como foi realizado com os meses:
    1. Se o dia atual for menor que o dia de nascimento, é porque faltam uns dias ainda para seu aniversário (Visto o exemplo anterior, os anos seriam 31).
    2. Se o dia atual for maior ou igual que o dia de nascimento é porque já cumpriu anos (Visto o exemplo anterior, os anos seriam 32).

O script para calcular a idade

Bom, com estas explicações esperamos que qualquer pessoa com um nível médio de Javascript, poderia realizar o código desta função, porém o objetivo é mostrar-lhes nossa proposta de código, que está comentada para que possa ser entendida facilmente.

```
//calcular a idade de uma pessoa
//recebe a data como um string em formato portugues
//devolve um inteiro com a idade. Devolve false em caso de que a data seja incorreta ou maior que o dia atual
function calcular_idade(data){

    //calculo a data de hoje
    hoje=new Date()
    //alert(hoje)

    //calculo a data que recebo
    //descomponho a data em um array
    var array_data = data.split("/")
    //se o array nao tem tres partes, a data eh incorreta
    if (array_data.length!=3)
       return false

    //comprovo que o ano, mes, dia são corretos
    var ano
    ano = parseInt(array_data[2]);
    if (isNaN(ano))
       return false

    var mes
    mes = parseInt(array_data[1]);
    if (isNaN(mes))
       return false

    var dia
    dia = parseInt(array_data[0]);
    if (isNaN(dia))
       return false


    //se o ano da data que recebo so tem 2 cifras temos que muda-lo a 4
    if (ano<=99)
       ano +=1900

    //subtraio os anos das duas datas
    edad=hoje.getYear()- ano - 1; //-1 porque ainda nao fez anos durante este ano

    //se subtraio os meses e for menor que 0 entao nao cumpriu anos. Se for maior sim ja cumpriu
    if (hoy.getMonth() + 1 - mes < 0) //+ 1 porque os meses comecam em 0
       return idade
    if (hoje.getMonth() + 1 - mes > 0)
       return idade+1

    //entao eh porque sao iguais. Vejo os dias
    //se subtraio os dias e der menor que 0 entao nao cumpriu anos. Se der maior ou igual sim que já cumpriu
    if (hoje.getUTCDate() - dia >= 0)
       return idade + 1

    return idade
}
```

Nota: Para entender esta função será necessário saber que, quando se executa return dentro de uma função, se devolve o valor indicado e se sai da função, sem que se possa executar outras sentenças que

existam debaixo do return.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Iluminar formulários com CSS e Javascript

Ler o artigo sobre iluminação de tabelas, células, filas me parece muito interessante e necessário em alguns trabalhos de programação de páginas webs. Interessado em tal artigo e com ajuda do manual de Programação em Javascript II, que também está publicado neste site, mostra-se este pequeno trabalho.

Passo 1:

As cores de fundo das caixas de texto e algumas outras propriedades se manipulam muitas vezes utilizando folhas de estilo. Neste primeiro passo, será feito de forma bastante simples. Coloque este script no head <head> xxx </head>

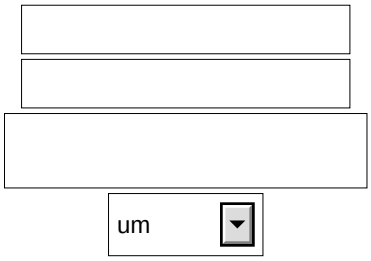
```
<script>
function form_um(num_form,num_elem_form,cor_entrada) {
document.forms[num_form].elements[num_elem_form].style.backgroundColor=cor_entrada;
document.forms[num_form].elements[num_elem_form].focus();
}
function form_dois(num_form,num_elem_form,cor_default) {
document.forms[num_form].elements[num_elem_form].style.backgroundColor=cor_default;
}
</script>
```

Passo 2:

Agregamos o código da seguinte maneira:

```
<form method="post" action="" name="meuformulario">
<p>
<input type="text" name="campo1" onMouseOver="form_um(0,0,'Lavender');" onMouseOut="form_dois(0,0,'ffffff');"
class="caixa" >
<br>
<input type="text" name="campo2" onMouseOver="form_um(0,1,'red');" onMouseOut="form_dois(0,1,'ffffff');"
class="caixa">
<br>
<textarea name="campo3" onMouseOver="form_um(0,2,'blue');" onMouseOut="form_dois(0,2,'ffffff');"
class="caixa"></textarea>
<br>
<select name="campo4" onMouseOver="form_um(0,3,'cccccc');" onMouseOut="form_dois(0,3,'ffffff');"
class="caixa">
<option value="1">um</option>
<option value="2">dois</option>
<option value="3">tres</option>
<option value="4">quatro</option>
<option value="5">cinco</option>
</select>
</form>
```

O resultado é o seguinte:



Até este ponto vemos que nosso formulário se vê bastante convencional, para melhorar um pouco nossa apresentação, deve-se incluir código de estilo, como o que segue:

Passo 3:

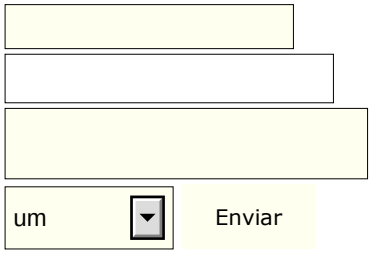
Colocar este script no head <head> xxx </head>:

```
<style type="text/css">
<!--
.caixa {
PADDING-RIGHT: 0.1em; PADDING-LEFT: 0.1em; PADDING-BOTTOM: 0.1em; FONT: 9pt "Verdana, Tahoma, Arial";
MARGIN-LEFT: 0.1em; PADDING-TOP: 0.1em; BACKGROUND-COLOR: #FFFFF0; TEXT-ALIGN: left
}
.boton {
BORDER-RIGHT: #000000 1px solid; BORDER-TOP: #000000 1px solid; FONT-SIZE: 11px; BACKGROUND: #FFFFF0;
BORDER-LEFT: #000000 1px solid; COLOR: #000000; BORDER-BOTTOM: #000000 1px solid; FONT-STYLE: normal;
FONT-FAMILY: verdana, arial, "trebuchet MS", helvetica, sans-serif
}
-->
</style>
```

Passo 4:
O que falta fazer é somente o link dos objetos do formulário com o respectivo estilo que desejarmos. Então, o código ficaria assim como segue e o resultado se vê no seguinte quadro.

```
<form method="post" action="" name="meuformulario2">
<input type="text" name="campo12" onMouseOver="form_um(1,0,'Lavender');"
onMouseOut="form_dois(1,0,'FFFFF0');" class="caixa" >
<br>
<input type="text" name="campo22" onMouseOver="form_um(1,1,'red');"
onMouseOut="form_dois(1,1,'FFFFF0');">
<br>
<textarea name="textarea" onMouseOver="form_um(1,2,'blue');"
onMouseOut="form_dois(1,2,'FFFFF0');" class="caixa"></textarea>
<br>
<select name="select" onMouseOver="form_um(1,3,'cccccc');"
onMouseOut="form_dois(1,3,'FFFFF0');" class="caixa">
<option value="1">um</option>
<option value="2">dois</option>
<option value="3">três</option>
<option value="4">quatro</option>
<option value="5">cinco</option>
</select>
<input type="submit" name="Submit" value="Enviar" class="botao">
</form>
```

O resultado 2 é o seguinte:



As funções form_um e form2 se explicam a seguir:

function form_um(num_form,num_elem_form,cor_entrada)

Os 2 primeiros parâmetros correspondem a números que representam a um formulário (num_form), você pode observar que o primeiro formulário se chama meuformulario e o segundo meuformulario2, ao invés de empregar estes nomes, se empregam valores numéricos, o seguinte parâmetro representa um objeto de tal formulário (num_elem_form) e o terceiro parâmetro representa um valor de uma cor ou seu respectivo nome.
O corpo de cada função corresponde à manipulação das propriedades dos objetos dos formulários.

> **Referência:** Para entender melhor parte do código Javascript, recomenda-se ler o manual Programação em Javascript II.

*Artigo por **Fabio Núñez Iturriaga***

Autozoom de texto com Javascript

Este script faz com que o tamanho da letra de um banner de texto vá crescendo até alcançar um valor máximo pré-definido.

O funcionamento é bem simples, como qualquer animação em Javascript, necessita um timer ou temporizador que cada certo tempo se dispara chamando a uma função responsável do efeito de animação, neste caso modificar o tamanho de um texto. Esta função se chama cresceLetra().

Ao carregar a página chama-se a função resetear() que é a que inicia todos os valores: objeto ao que se aplica o efeito (um elemento DIV), tamanho mínimo e máximo do texto e velocidade com a que crescerá. Esta função ativa o timer que chama a cresceLetra(), encarregada de aumentar em 1 px o tamanho do texto atuando sobre a propriedade style.fontSize do objeto ao que se aplica o efeito, e de reativar o timer.

Assim até que o texto alcance o tamanho máximo em cujo caso volta a chamar a resetear() (esta vez sem argumentos) que repõe o tamanho inicial e se repete o ciclo. Esta chamada se faz mediante um timer para manter em tela o texto a tamanho grande durante um certo tempo (500 milésimos de segundos no exemplo).

O código Javascript

```
<script language="JavaScript">
var banZoom = null

function cresceLetra()
{
var obj = banZoom
var tma
tma = parseInt(obj.style.fontSize)
window.status = obj.style.fontSize
if (tma<obj.maxTam)
   {
   obj.style.fontSize = tma + 1
   setTimeout("cresceLetra("+obj.maxTam+")",20)
   }
else
   setTimeout("resetear()",500)

}

function resetear(mn, mx, rapidez, idBan)
{
if (banZoom == null)
   {
   banZoom = document.getElementById(idBan)
   banZoom.maxTam = mx
   banZoom.minTam = mn
   banZoom.rapidez = rapidez
   }
banZoom.style.fontSize = banZoom.minTam
setTimeout("cresceLetra()",rapidez)
}
</script>
```

O código HTML

O evento onload se vincula a resetear() com os argumentos iniciais que são, por esta ordem, tamanho mínimo das letras, tamanho máximo que devem alcançar, rapidez com que crescerão e o identificador do elemento ao que se aplica.

Neste caso este elemento é um bloco DIV cujo identificador ID é 'letras'. Este bloque DIV você pode desenha-lo como quiser em quanto a cores, tipo de letra a usar, bordas, imagem de fundo...

```
<body onload="resetear(10, 48, 10, 'letras')">
<DIV id="letras" style="position:absolute; font-size:10px; height:56px; width: 761px; background-color:
#CCFFCC;border: 1px none #000000;">Toda a frase vai crescendo</DIV>
<p> </p>
<p> </p>
<p>E o que acontece com o resto </p>
</body>
```

O exercício pode-se ver em funcionamento em uma página a parte.

*Artigo por **Juan Carlos Gámez***

Javascript para evitar que a página se mostre em um frame

Existe uma utilidade muito simples sobre o controle de frames em Javascript que também é muito útil para qualquer web site. Trata-se de evitar que nossa página se mostre dentro de qualquer divisão de frames e pode ser muito interessante para evitar que um link de qualquer site introduza nossa página dentro de seu design ou estrutura de menus.

Talvez aparecer dentro de um frame em muitos casos não nos importe muito, porém, reduzem o espaço para mostrar nossa própria página e ainda comprimem um desenho.Portanto, não tem porque fazer nenhum bem.

O script

Com uma só linha de código é suficiente para criar este efeito. Esta linha pode ser colocada em qualquer parte do documento HTML, embora seria recomendável que ficasse pela parte superior ou dentro do cabeçalho, para que tenha que carregar a página inteira para se expandir a todo o espaço da janela.

```
<script language="JavaScript">
<!--// evito que se carregue em outro frame
if (top.location != self.location)top.location = self.location;
//-->
</script>
```

Neste script se comprova se as propriedades top.location, que faz referência à URL da declaração de frames, no caso de que tenha e self.location, que faz referência à URL do documento onde está o script.

Se as duas URLs são iguais significaria que a página não está carregada dentro de um frame e se são diferentes querá dizer que sim esta se mostrando no espaço de frame.

No caso de serem distintas, simplesmente se indica que na janela do navegador ao completo (top.location) mostre-se a URL da página onde está o script (self.location).

Não tem muitas complicações. Pode-se ver em uma página a parte.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Elementos de formulário select associados

Vamos conhecer um dos truques mais solicitados de Javascript, que tem muita relação com o tema de formulários e onde se utiliza o evento onchange de Javascript. É um exemplo sobre como realizar uma página com dois selects onde, segundo o valor escolhido em um deles, mudem as opções possíveis do outro select.

O melhor para ver o que vamos fazer é ver uma página web onde se mostra em funcionamento o script. Para ver seu funcionamento, devemos mudar a seleção do primeiro select e comprovaremos como as opções do segundo select mudam automaticamente.

O exemplo que ilustramos utiliza estados e países. Ao escolher no primeiro select um país, no segundo deve nos mostrar os estados desse país para que possamos escolher um estado, mas

somente um que esteja no país selecionado no primeiro término.

Conhecer o objeto select e os option

É importante conhecer os objetos de formulário select e os option. Os select correspondem com as caixas de seleção desdobráveis e os option com cada uma das opções da caixa desdobrável. Podemos ver um artigo que fala sobre isso.

Teoricamente nos interessa fazer várias coisas que têm a ver com extrair o valor de um select em qualquer momento, observar seu número de opções e, para cada opção, colocar seu valor e seu texto associado. Tudo isto aprenderemos a fazer neste exemplo.

> **Referência:** Para conhecer o trabalho e a hierarquia de objetos javascript (Tudo isso é a base do trabalho com os elementos das páginas em Javascript) devemos ler o manual de Javascript II.

Modo de solucionar o problema

Para começar, vamos utilizar um formulário com dois selects, um para o país e outro para o estado.

```
<form name="f1">
<select name=pais onchange="muda_estado()">
<option value="0" selected>Selecione...
<option value="1">Espanha
<option value="2">Brasil
<option value="3">Portugal
<option value="4">França
</select>

<select name=estado>
<option value="-">-
</select>
</form>
```

Observamos no select associado ao país deste formulário que, quando se muda a opção de país, deve-se chamar a função muda_estado(). Veremos mais adiante esta função, agora é importante observar que está associada ao evento onchange que se ativa quando muda a opção no select.

Todo o resto será código Javascript. Começamos definindo um montão de de arrays com os estados de cada país. Neste caso temos só 4 países, então necessitaremos 4 arrays. Em cada array tenho uma lista de estados de cada país, colocados em cada um dos elementos do array. Ademais, deixaremos o primeiro campo com um valor "-" que indica que não foi selecionado nenhum estado.

```
var estados_1=new Array("-","Andalucía","Asturias","Baleares","Canarias","Castilla y León","Castilla-La Mancha","...")
var estados_2=new Array("-","Rio de Janeiro","Bahia","São Paulo","Santa Catarina","Minas Gerais","...")
var estados_3=new Array("-","Algarve","Alentejo","Norte","Vale do Tejo","...")
var estados_4=new Array("-","Aisne","Creuse","Dordogne","Essonne","Gironde ","...")
```

Observemos que os índices do array de cada país se correspondem com os do select do país. Por exemplo, a opção Espanha, tem o valor associado 1 e o array com os estados de Espanha se chama estados_1.

O script se completa com uma função que realiza o carregamento dos estados no segundo

select. O mecanismo realiza basicamente estas ações:

- Detecto o país que foi selecionado
- Se o valor do país não for 0 (o valor 0 é quando não foi selecionado nenhum país)
    - Tomo o array de estados adequado, utilizando o índice do país.
    - Marco o número de opções que deve ter o select de estados
    - Para cada opção do select, coloco seu valor e texto associado, que faz corresponder com o indicado no array de estados.
- SE NÃO (O valor de país é 0, não foi selecionado país)
    - Coloco no select de estado um único option com o valor "-", que significava que não havia estado.
- Coloco a opção primeira do select de estado como o selecionado.

A função tem o seguinte código. Está comentado para que se entenda melhor.

```
function muda_estado(){
   //tomo o valor do select do pais escolhido
   var pais
   pais = document.f1.pais[document.f1.pais.selectedIndex].value
   /vejo se o pais está definido
   if (pais != 0) {
      //se estava definido, entao coloco as opcoes do estado correspondente.
      //seleciono o array de estado adequado
      meus_estados=eval("estados_" + pais)
      //calculo o numero de estados
      num_estados = meus_estados.length
      //marco o número de estados no select
      document.f1.estado.length = num_estados
      //para cada estado do array, o introduzo no select
      for(i=0;i<num_estados;i++){
         document.f1.estado.options[i].value=meus_estados[i]
         document.f1.provincia.options[i].text=meus_estados[i]
      }
   }else{
      //se não havia estado selecionado, elimino os estados do select
      document.f1.estado.length = 1
      //coloco um traço na única opção que deixei
      document.f1.estado.options[0].value = "-"
      document.f1.estado.options[0].text = "-"
   }
   //marco como selecionada a opção primeira de estado
   document.f1.estado.options[0].selected = true
}
```

Podemos ver uma página com o exemplo em funcionamento.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Contar os caracteres escritos em um textarea

O desenho deste script foi motivado pela necessidade de fazer a típica caixa para enviar mensagens SMS pela Internet. O tamanho máximo de uma mensagem SMS de celular é de 160 caracteres, com o qual é muito útil que a própria página lhe informe sobre o número de caracteres que levam escritos na mensagem, para que o interessado não passe do máximo permitido.

O funcionamento é muito simples de entender, mas de qualquer forma, podemos ver o script em funcionamento para saber exatamente o que propomos.

O formulário

A página apresentará um formulário com dois campos. O primeiro com o textarea onde o usuário escreverá a mensagem e o segundo, um campo de texto onde mostraremos em todo momento os caracteres escritos.

O único detalhe para ter em conta, relacionado com Javascript é o par de eventos que temos definidos dentro do campo textarea, que servem para chamar à função que realiza a conta dos caracteres no momento em que o usuário pressiona ou solta as teclas. Teoricamente, utiliza-se o evento onKeyDown para definir as ações a realizar quando se apertar a tecla e onKeyUp, para definir ações a executar quando se soltar a tecla apertada.

```
<form action="#" method="post">
<table>
<tr>
   <td>Texto:</td>
   <td><textarea cols="40" rows="5" name="texto" onKeyDown="conta()" onKeyUp="conta()"></textarea></td>
</tr>
<tr>
   <td>Caracteres:</td>
   <td><input type="text" name=caracteres size=4></td>
</tr>
</table>
</form>
```

O script que conta caracteres

Com o formulário e os dois eventos introduzidos temos tudo que é necessário para que se contem -e recontem- os caracteres cada vez que o visitante, situado sobre o textarea, clica sobre as teclas, ou seja, cada vez que se escreve texto no textarea. Agora simplesmente nos falta definir a função que se encarrega de realizar a conta propriamente dita e situa-la no outro campo de texto do formulário.

```
<script>
function conta(){
      document.forms[0].caracteres.value=document.forms[0].texto.value.length
}
</script>
```

Talvez muitos tenham se surpreendido com a simplicidade do script, mas é que realmente não faz falta mais.

A propriedade value do textarea contém o texto escrito e por sua vez, a propriedade length salva o número de caracteres de tal texto. Assim, document.forms[0].texto.value.length equivale ao número de caracteres introduzidos dentro do textarea. Este valor se atribui ao conteúdo do campo de texto do formulário onde salvamos o número de caracteres, mediante a propriedade value do campo: document.forms[0].caracteres.value.

Com tudo isso, se mostrará no campo de texto o número de caracteres do textarea. Pode-se ver por exemplo em uma página a parte.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Passo de parâmetros em HTML com client-side Javascript

O script seguinte se colocará à prova na página que deve de receber os parâmetros, ou poderá se copiar em um arquivo .js vinculado. Em ambos casos o código não deve estar em nenhuma função, para que se execute sempre que se carrega a página. Uma vez executado o script teremos os valores recebidos nas correspondentes variáveis. Aquelas que não tivermos recebido terão o valor padrão que tivermos fixado.

O Código:

```
<script language="javascript">
//Autor: Bruno Suárez Laffargue
//Versao: 1.1

//Definimos as variaveis necessarias
variable=umValorPadrao;//Havera que estabelecer

//Capturamos a URL
var callingURL = document.URL;

//Separamos os parametros
var cgiString = callingURL.substring(callingURL.indexOf('?')+1,callingURL.length);

//Observamos o separador entre parametros
var DELIMETER = '&';

//Eliminamos a almofadinha, se existir... cortamos por prevencao!
if (cgiString.indexOf('#')!=-1){
   cgiString=cgiString.slice(0,cgiString.indexOf('#'));
}

//Partimos o cgiString ja limpo, separando cada par variable=valor
//em uma das posicoes do array
var arrayParams=cgiString.split(DELIMETER);

//Percorremos o array de parametros avaliando cada um dos pares variable=valor
for (var i=0;i<arrayParams.length;i++){
   eval(arrayParams[i].substring(0,arrayParams[i].indexOf('=')+1)+"\""+
   arrayParams[i].substring(arrayParams[i].indexOf('=')+1,arrayParams
    [i].length)+"\"");
}
</script>
```

Uma das restrições funcionais para que isto funcione é ter tantas variáveis javascript definidas como parâmetros forem recebidos, iniciados a um valor padrão. Ademais, estas variáveis hão de se chamar exatamente igual que os parâmetros, já que senão, não funcionará. Em nenhum caso.

*Artigo por **Bruno Suárez Laffargue***

Moldura dinâmica em Javascript com texto que muda

Realizamos um simples script em Javascript para realizar uma moldura com informação dinâmica, que muda com cada impressão da página. O exercício consiste em uma tabela HTML que mostra uma informação que muda, utilizando Javascript para que nos proporcione um

texto aleatório, que logo imprimiremos dentro da tabela HTML.

Para começar veremos como obter um texto aleatório. A idéia a desenvolver é bem simples. Criamos um array com os diferentes textos, entre os que se escolherá um aleatoriamente. Obtemos um valor numérico aleatório entre 0 e máximo índice do array e se imprime o texto que há no array nessa posição aleatória.

```
function texto_aleatorio(){
  var textos = new Array()
  textos[0] = "Temos os melhores produtos do mercado, com controles de    qualidade intensivos."
  textos[1] = "Distribuímos em todo o mundo com os melhores tempos de    entrega e confiança nos envios."
  textos[2] = "Não temos concorrentes que nos superem. Contrate    conosco e comprove. É muito fácil."
  textos[3] = "Disponha do melhor serviço de atenção ao cliente e uma    resposta rápida aos seus problemas."
  textos[4] = "Os melhores serviços, produtos e, claro, os menores    preços. Tudo são vantagens."

  aleat = Math.random() * (textos.length)
  aleat = Math.floor(aleat)

document.write(textos[aleat])
}
```

Na primeira linha se cria o array e nas seguintes se iniciam seus diferentes campos com textos variados. Tal como fizemos o exercício, o número de campos que tiver o array é indiferente, portanto poderíamos aumentar seus campos, introduzindo novas frases, e así as possibilidades de textos serão mais variadas.

Mais adiante na função se obtém um número aleatório. Para obtê-lo utilizamos a classe Math, mais concretamente o método random(). Random devolve um valor decimal aleatório entre 0 e 1. Algo como 0.453. Se multiplicarmos esse valor pelo número de campos do array obteremos um número entre 0 e o número de campos, porém ainda têm valores decimais e nós desejamos que seja inteiro para poder utiliza-lo como índice no array. Para converter esse valor a inteiro, o arredondamos para baixo com floor(), que devolve o número mais próximo, arredondamos por baixo.

Nas últimas linhas de função se imprime o valor aleatório.

O código HTML da moldura

```
<table width="180" border="0" cellspacing="1" cellpadding="2" bgcolor="000000">
  <tr>
  <td class="barra" bgcolor="993333"><font color="#FFFFFF"><b>Nossas vantagens</b></font></td>
  </tr>
  <tr>
   <td class="fuente8" bgcolor="#FFFFFF"><script language=javascript>texto_aleatorio()</script></td>
  </tr>
</table>
```

É uma tabela HTML muito simples. Simplesmente mostra uma célula com o cabeçalho ou titular da caixa e uma segunda célula na que simplesmente há uma chamada à função que escreve o texto aleatório.

O resultado pode ser visto em uma página a parte. Tenha em conta que há que atualizar a página para ver como vão se mostrando diferentes mensagens, entre todas as que configuramos.

Podemos complicar este exemplo em tudo que desejarmos para criar páginas dinâmicas que

mudem os conteúdos entre diferentes acessos do visitante. Poderíamos fazer que se incluam diferentes elementos aleatórios, como imagens, animações, links, etc.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Criação de gráficos de barras com Javascript

Vamos apresentar um sistema para realizar gráficos de barras em páginas web, com a ajuda de Javascript. Sem dúvida, qualquer pessoa poderá encontrar várias soluções para este problema, embora nós vamos recomendar uma que de verdade pode simplificar muito nossa vida e que não requer nenhum requisito especial. Simplesmente que o navegador do visitante suporte Javscript.

Os gráficos de barras servem para fazer muitas coisas e é uma das tarefas típicas se que se pode necessitar em um projeto um pouco avançado. Geralmente, se utilizam para mostrar resultados de informes, estatísticas ou temas similares de uma maneira muito visual.

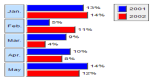


Imagem de um dos gráficos que se podem gerar com este mecanismo.

Com o seguinte exemplo, vamos aprender a fazer gráficos dinamicamente, ou seja, que se constroem com uns dados que podem ser variáveis. Se o gráfico fosse construído sempre com os mesmo dados, poderíamos cria-lo com algum programa como o Excel e logo converte-lo em imagem e apresenta-lo dentro da web. Entretanto, é muito habitual que os dados sejam extraídos de uma fonte como um banco de dados ou algo similar, onde podem variar com o tempo. Neste caso, cada vez que se imprima a página, se deveria acessar ao banco de dados, extrair a informação e gerar o gráfico dinamicamente.

BAR_GRAPH

O sistema que apresentamos aqui se chama BAR_GRAPH e serve para gerar gráficos de barras de diferentes tipos com a ajuda de Javascript. Os gráficos têm um aspecto visual muito adequado e são totalmente personalizáveis, podendo configurar o sistema para que mostre gráficos horizontais, verticais e de progresso. Também inclui a possibilidade de criar etiquetas, legendas, agrupar certos valores, utilizar cores diferentes, etc.

O produto é gratuito para uso pessoal, embora se se utiliza de maneira comercial, o autor solicita que se faça uma doação de 6 euros, através de Paypal.

Para a criação dos gráficos não se necessita de nenhum material, exceto umas bibliotecas Javascript que são as que oferecem todas as funcionalidades de criação de gráficos.

Para incluir a biblioteca em nossas páginas web, basta baixa-las do web site do autor do produto: http://www.gerd-tentler.de/tools/graphs/ e chamá-la através de nossos scripts.

Para incluir uma biblioteca Javascript, o habitual é colocar no cabeçalho da página uma chamada ao arquivo onde está o código. Algo como isto:

```
<head>
<title>Exemplo de gráficos</title>
```

```
<script src="graphs.js" type="text/javascript"></script>
</head>
```

Observamos que se inclui um arquivo chamado graphs.js que se supõe que estará no mesmo diretório que a página web. Dentro desse arquivo encontraremos a declaração de um objeto que será o encarregado de receber os dados de origem com os quais se deseja construir o gráfico e também de mostrá-lo quando tiver introduzido todos os dados iniciais.

Para criar e mostrar um gráfico, o único que temos que fazer é instanciar o objeto gráfico, inserir os valores iniciais e chamar ao método que faz com que se mostre o gráfico.

Pode-se ver um exemplo a seguir das instruções iniciais de criações do gráfico:

```
//instanciamos o gráfico
graph = new BAR_GRAPH("hBar");
//inserimos valores
graph.values = "380,150,260,310,430";
//mostramos o gráfico na página
document.write(graph.create());
```

Pode-se complicar um pouco mais este sistema para incorporar, por exemplo, dois turnos de valores, agrupados em dois grupos.

```
//instanciamos o gráfico
graph = new BAR_GRAPH("hBar");
//definimos as etiquetas que acompanharão a cada casal de valores
graph.labels = "Jan.,Feb.,Mar.,Apr.,May";
//iniciamos os dados. 1 elemento tem um grupo de dois valores.
graph.values = "380;420, 150;340, 260;120, 310;250, 430;370";
//definimos uma legenda
graph.legend = "2001,2002";
//mostramos o gráfico na página
document.write(graph.create());
```

Se desejarmos ver muitos mais exemplos de gráficos possíveis seria muito interessante acessarmos à página do produto, onde veremos os tipos de gráficos mais habituais que podem se realizar. http://www.gerd-tentler.de/tools/graphs/

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Jogos em Javascript

Javascript se encaixa dentro das linguagens para a web do lado do cliente. É compatível com os navegadores mais habituais, embora não com todos. Outra das características desta linguagem é que pode variar de um navegador a outro, inclusive pode ser diferente em versões distintas do mesmo navegador.

Por tudo isso e porque Javascript é muito simples de utilizar para pequenos programas e muito complicado para aplicações mais elaboradas, não é uma linguagem muito apropriada para fazer jogos para a web. Pode acontecer que um jogo não se veja corretamente em todos os navegadores ou que não funcione em certas plataformas. Em qualquer caso, certamente é muito difícil de manter e de melhorar uma vez realizado.

Não obstante, no mundo há pessoas para tudo e muitos loucos do Javascript nos deixaram autênticas obras de arte e de engenho, que nos demonstram que não é tão difícil fazer aplicações avançadas para a web com esta linguagem. Neste artigo, vamos mostrar vários jogos de Javascript, realizados com muita dedicação e que em quase todos os casos, podemos incluir livremente em nossas páginas web.

JsTetris

O conhecido jogo de peças que caem e temos que colocar da maneira mais ordenada. Funciona tanto em Internet Explorer como em Nestcape. O único que não funciona ainda muito rápido como deveria são os eventos do teclado, que às vezes não respondem tão rápido como se desejaria.
http://gosu.pl/dhtml/JsTetris.html

Buscaminas

O típico jogo do Windows, muito útil aliás não só para exercitar a lógica, como também para adquirir uma maior soltura com o manejo do mouse, para aqueles que possuem dificuldade em move-lo.
http://www.ecirce.com/dhtml/minesweeper/

Memory Game

Um jogo que temos várias cartas que devem ser agrupadas por duplas. Neste caso, proporciona-se o jogo em várias versões para Javascript, Flash, Java e PHP...
http://mypage.bluewin.ch/katzenseite/docs/en/games/games.html

Serpente

Um clássico onde estiver. Você maneja uma serpente que come bolinhas e cada vez se torna maior.
http://www.dynamicdrive.com/dynamicindex12/snake/index.htm

Puzzle de bolhas

Trata-se de lançar bolhas coloridas. Quando consegue-se 3 bolhas juntas da mesma cor, se descartam da tela. Perde-se quando não cabem mais bolhas na tela. É algo parecido ao Tetris.
http://www.javafile.com/games/bubbles/bubbles.php

Existem muito mais jogos em Javascript. Aqui só assinalamos alguns que nos parecem divertidos ou muito avançados tecnicamente. Para encontrar outros jogos em Javascript recomendamos acessar a algumas páginas com coleções de scripts como Hotscripts http://www.hotscripts.com ou javafile http://www.javafile.com.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Cross-Browser. DHTML compatível com todos os navegadores.

É bastante conhecido, para todos os que tiveram experimentado com camadas e Javascript, o problema da compatibilidade de navegadores e a necessidade de realizar diferentes códigos

DHTML dependendo do explorador que tenha o cliente.

Se você ainda não tiver encontrado problemas de compatibilidade de seus scripts, possivelmente nunca deve ter enfrentado a programação do lado do cliente em Javascript, mais concretamente com o trabalho com as camadas e portanto, não lhe interessará muito este artigo.

Agora também, se tiver problemas com as camadas ao programar uma web e não conseguir que se veja bem nos distintos navegadores, este artigo poderá tornar a sua vida mais simples. Isto graças a cross-browser.com, um website que oferece uma interessante biblioteca de funções, que vamos apresentar a seguir, para o trabalho com HTML Dinâmico compatível com todos os navegadores mais habituais.

X Library VS CBE API

Em cross-browser.com nos proporcionam duas bibliotecas diferentes para o trabalho com DHTML. Na verdade, trata-se de um mesmo conjunto de funções, sendo uma biblioteca a evolução de outra. CBE API é a biblioteca original e X Library é a sua evolução.

CBE API é a biblioteca mais antiga e ainda tem maiores funcionalidades, porém em um espaço de tempo X Library irá dispor das mesmas funcionalidades que a outra, embora com diversas melhorias como uma considerável redução do peso dos arquivos Javascript.

À princípio, se entrarmos em cross-browser.com, nos apresentam X Library, embora um pouco mais escondido encontraremos seu antecessor CBE API. O conselho é que se utilize X Library para projetos novos. De qualquer forma, CBE API continua perfeitamente vigente (por enquanto).

Como utilizar X Library

Temos que começar baixando a versão atual de X Library. Para isso, entramos em cross-browser.com e acessamos à sessão de Toys (para o criador deste website, seus brinquedinhos são bibliotecas DHTML). Aqui podemos fazer o download da última versão, onde encontraremos as bibliotecas e exemplos de uso.

Uma vez descompactado o pacote, podemos encontrar uma pasta chamada "x", onde estão as bibliotecas. Observamos que há uma boa quantidade de arquivos .JS, cada um com diversas funções. Dependendo da ação que se deseja realizar temos que chamar a uma função ou outra, e teremos que saber em que arquivo .JS se encontra. As funções mais habituais se encontram no arquivo x_core.js. Em nossas páginas não temos porquê incluir todos os arquivos do diretório, só os que formos utilizar. Podemos ver em que arquivo está cada função na documentação do próprio programa, mais concretamente na referência de funções (o diretório onde se hospeda a documentação está dentro da pasta "docs" que está dentro do diretório "x").

Dentro do diretório "x" também há uma pasta de exemplos, chamada "examples", aos quais podemos dar uma olhada para começar a nos familiarizarmos com as possibilidades da biblioteca. Nós criamos dois exemplos para comprovar as bondades deste software. Para isso, seguimos o exemplo do código da própria página cross-browser.com e o dos diferentes exemplos que se oferecem. Algumas funcionalidades dos exemplos, como a função xInclude() estão ainda em fase de experimentação, portanto nós não a utilizamos e em seu lugar incluímos os arquivos que necessitamos tal como se faz habitualmente em Javascript.

Exemplo mostrar e ocultar camada

X Library dispõe de uma simples função para mostrar e ocultar uma camada. Simplesmente recebe o identificador da camada que se deseja mostrar ou ocultar. À princípio, podemos lhe enviar à função o próprio objeto camada ou então, um string com seu identificador (que se indica com o atributo id na etiqueta).

> **Referência:** No manual de CSS temos explicações de o que são as camadas e a forma de cria-las e definir seu posicionamento e aparência.

```
<html>
<head>
   <title>Exemplo de utilização de Cross-Browser</title>
<script type='text/javascript' src='x_core.js'></script>
<script type='text/javascript'>
function mostra(){
   xShow('c1');
}
function oculta(){
   xHide('c1');
}
</script>
</head>

<body>
<div id="c1" style="position:absolute; left: 200px; top: 100px; background-color:#9999aa; width:100px;
height:80px;">
Hola!
</div>

<form>
<input type=button onclick="muestra()" value="Mostra Capa">
<input type=button onclick="oculta()" value="Oculta Capa">
</form>

</body>
</html>
```

Neste exemplo, criamos uma camada (com identificador "c1") e dois botões. Quando se clica um deles se mostra a camada e quando se clica o outro se oculta. Para isso, criamos também duas funções, que se chamam quando se clica nos botões, cuja obrigação é mostrar e ocultar a camada utilizando a biblioteca X Library.

Faremos o uso das fuunções xShow() e xHide(), que recebem o identificador da camada que há que mostrar ou ocultar respectivamente. Estas duas funções se encontram no arquivo "x_core.js", que incluímos na página como bloco de script externo.

Pode-se visualizar este simples exemplo em uma página web a parte.

Exemplo para fazer um movimento de camada

O segundo exemplo que criamos é uma camada que se move pela página da esquerda para direita. Esta página tem também dois botões, para deter o movimento ou coloca-lo em funcionamento.

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

```
    <title>fazemos um scroll</title>
<script type='text/javascript' src='x_core.js'></script>
<script type='text/javascript'>
function inicia(){
    velocidade=2
}
function detem(){
    velocidade=0
}
function move(){
    posicao+=velocidade
    posicao %= 500
    xMoveTo('c1',posicao,100)
    setTimeout("move()",100)
}

window.onload = function()
{
    velocidade = 0
    posicao = 200
    move()
}

</script>
</head>

<body>
<div id="c1" style="position:absolute; left: 200px; top: 100px; background-color:red; width:100px; height:20px;">
Hola!
</div>

<form>
<input type=button onclick="inicia()" value="Move a Camada">
<input type=button onclick="detem()" value="Para a Camada">
</form>

</body>
</html>
```

Neste caso, em relação à biblioteca, utilizamos a função xMoveTo(), que recebe o identificador da camada a mover e as novas coordenadas onde coloca-la. Esta função também se encontra dentro do arquivo "x_core.js", que incluímos com o primeiro bloco de script.

Para entender o movimento da camada temos que ter visto alguma vez a função setTimeout(), que recebe uma instrução Javascript para executar e uma quantidade de milésimos de segundos que devem de passar antes da execução. No exemplo temos uma função que se chama move(), que se encarrega de variar a posição atual da camada. Esta função se chama a si mesma por meio de setTimeout(), com um atraso de 100 milésimos de segundos, portanto, a função move() nunca para de se executar, teoricamente 10 vezes por segundo.

> **Nota:** setTimeout é um método do objeto window. Temos vários artigos que utilizam esta função, por exemplo: Relógio em Javascript ou Texto em movimento na barra de estado.

Logo, definimos uma variável velocidade, que é o número de pixels que se desloca de camada em cada chamada move(). A única coisa que fazem os dois botões é modificar o valor dessa variável velocidade. O botão que para o movimento simplesmente atribui o valor zero à velocidade.

Outra coisa que temos que ver é que foi definida uma série de instruções a executar quando se

carrega a página, no bloco window.onload = function(). Entre essas ações a executar encontra-se a configuração da posição inicial da camada e a velocidade do movimento. Ademais, faz-se uma chamada inicial a move(), que desencadeia o fluxo do movimento, pois a função move se encarrega de chamar-se a si mesma até que o usuário abandona a página.

Se se deseja, pode-se ver o exemplo em funcionamento.

Conclusão

Não era nossa intenção explicar todas as funcionalidades desta ferramenta, e sim torna-la conhecida. Tampouco é nossa intenção explicar como realizar um movimento de uma camada ajudados por setTimeout(), mesmo assim pedimos desculpas para aquele leitor que não pôde entender por que realizamos um script de movimento dessa forma.

Nosso objetivo era apresentar as bibliotecas e mostrar como, com muito pouco código e sem ter que conhecer os segredos de cada navegador, pode-se realizar um exemplo já bastante avançado do manejo de camadas.

Recomendamos que não re-inventem a roda! Já que se dispõe de ferramentas de trabalho com camadas tão versáteis como X Library, é muito mais interessante basear nossos scripts nela do que quebrarmos a cabeça para inventar mecanismos compatíveis com cada navegador.

Criamos um manual onde vamos ir comentando vários exemplos de efeitos interessantes que se podem realizar utilizando estas bibliotecas. O manual chama-se Workshop de Cross-Browser DHTML.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

HTML Area. Editor WYSIWYG

Vamos apresentar uma ferramenta gratuita que serve para criar elementos de formulário, parecidos aos <textarea>, porém com a particularidade de que permitem introduzir texto com estilos, como em negrito, sublinhado, diferentes tipos de fontes e inclusive, tabelas ou imagens. Em suma, provê das funções típicas dos editores HTML WYSIWYG (what you see is what you get), mas dentro de um campo de formulário de uma página web.

htmlArea se inclui na página com umas poucas linhas Javascript fáceis de escrever. Com isso, obtemos um editor que permite funcionalidades como:

- Formatar texto em negrito, cursivas e sublinhados
- Mudar a tipografia e a cor
- Alinhar os diferentes parágrafos
- Incluir listas, linhas horizontais, links, images...

Como o programa está realizado em Javascript e trabalha unicamente do lado do cliente, poderemos utilizá-lo em qualquer tipo de servidor (não requer programação ASP ou PHP). A desvantagem é que funciona unicamente em versões de Internet Explorer 5.5 ou superiores. Pelo menos, em outros navegadores, não dará erros, e sim, que simplesmente veremos um campo <textarea> normal.

Como se insere o htmlArea

Vamos ver como se inclui este tipo de editor em uma página web. Como primeiro passo, baixamos o software que poderemos encontrar na página de início de htmlArea. O arquivo é muito pequeno. Vem em .zip, por isso, devemos descomprimí-lo em nosso computador.

Uma vez descomprimido, encontramos exemplos e instruções para seu funcionamento. Executando o arquivo chamado example.html podemos fazer uma pequena prova, para ver se nosso navegador suporta htmlArea. Se tudo estiver correto, podemos fazer nossa primeira prova.

Vamos criar um arquivo HTML, para fazer nosso primeiro exemplo, no mesmo diretório onde descomprimimos o software. Editamos e colocamos no cabeçalho o seguinte código:

```
<script language="Javascript1.2">
<!--
// Carregamento do htmlarea
_editor_url = "" // URL do arquivo htmlarea var win_ie_ver = parseFloat(navigator.appVersion.split("MSIE")[1]);
if (navigator.userAgent.indexOf('Mac') >= 0) { win_ie_ver = 0; }
if (navigator.userAgent.indexOf('Windows CE') >= 0) { win_ie_ver = 0; }
if (navigator.userAgent.indexOf('Opera') >= 0) { win_ie_ver = 0; }
if (win_ie_ver >= 5.5) {
document.write('<scr' + 'ipt src="' +_editor_url+ 'editor.js"');
document.write(' language="Javascript1.2"></scr' + 'ipt>');
} else { document.write('<scr'+'ipt>function editor_generate() { return false; }</scr'+'ipt>'); }
// -->
</script>
```

O único que há que editar neste código é a variável "_editor_url", à que temos que atribuir a rota onde se encontram os arquivos de htmlArea. Como neste caso o arquivo de exemplo está no mesmo diretório que htmlArea, atribuímos um string vazio à variável:

```
_editor_url = "" // URL do arquivo htmlarea
```

Continuamos colocando um formulário com um campo textarea dentro.

```
<form>
<textarea name="campo1" style="
```

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Ocultar um e-mail de um link para evitar o spam

Uno dos mecanismos que utilizam as pessoas que enviam spam (spammers) para obter endereços de correio para sua lista de distribuição é rastrear a web em busca de endereços de e-mail. Todas os endereços que aparecem nas páginas web, à vista ou escritas no código, são suscetíveis de ser capturadas e utilizadas para o envio de spam. Por isso, não é má idéia proteger nossos e-mails para tornar a tarefa difícil aos spammers, e evitar que em pouco tempo comecemos a receber mensagens não desejadas.

Um link a um correio eletrônico é assim:

```
<a href="mailto:correio@meudominio.com">correio@meudominio.com</a>
```

Tanto em href como no texto do link aparece nosso correio eletrônico. Neste artigo veremos umas idéias para evitar que apareçam nossos dados, de modo que não possam captar os endereços.

Utilizar uma imagem no texto do link

Uma boa solução consiste em utilizar uma simples imagem onde aparece o correio. Esta imagem terá o texto do correio eletrônico, para que o visitante possa visualizar o endereço na página, porém escrita sobre uma imagem. Isso é indetectável por um robô que escaneie a página e nossos clientes poderão ver claramente qual é o correio onde devem nos escrever.

Se não colocarmos o link e colocarmos somente a imagem, acabariam nossos problemas. O visitante não poderia clicar no endereço na própria página para nos enviar um mail, mas muito provavelmente seja suficientemente esperto para copia-la no programa de correio que for utilizar.

Utilizar Javascript para ocultar o endereço

Podemos por mediação de Javascript fazer um pequeno programa para que nosso endereço não apareça no código, pelo menos não tão claramente. Podemos, por exemplo, parti-lo em diferentes pedaços e logo concatena-lo, de maneira que não possa se ver por completo em nenhum lugar do código da página.

Vejamos este script:

```
<script language="JavaScript">
usuario="pedro"
dominio="qualquerum.com"
conector="@"


function dar_correio(){
  return usuario + conector + dominio
}

function escreve_link_correio(){
  document.write("<a href='mailto:" + dar_correio() + "'>" + dar_correio() + "</a>")
}
</script>
```

Primeiro, definem-se três variáveis que formam o correio eletrônico que desejamos ocultar. Logo, temos duas funções úteis:

A função dar_correio() devolve o correio eletrônico que se deseja ocultar. Simplesmente concatena as partes do correio eletrônico, que tinham sido definido nas variáveis mais acima.

Por sua parte, escreve_link_correio(), escreve na página web um link ao correio eletrônico completo. Um link a um correio eletrônico é assim:

```
<a href="mailto:correio@meudominio.com">correio@meudominio.com</a>
```

Esta função se apoia no dar_correio() para obter o correio que se desejava ocultar.

Para que apareça na página o link ao correio eletrônico devemos fazer uma chamada à função escreve_link_correio(), no lugar do corpo que desejarmos que se mostre.

> **Nota:** Se o computador do usuário não tiver Javascript ou tiver desabilitado, não poderá ver esses endereços de correio escritos na página desde Javascript. Por isso, pode ser uma boa idéia mudar este truque com o de mostrar uma imagem com o correio, para que pelo menos se veja a imagem. Embora ainda existam navegadores só em texto, com o qual nem sequer se veria a imagem. Enfim, existe um mundo de possibilidades.

Este seria o código para mostrar em qualquer parte da página.

```
<body>
<!-- em qualquer parte do corpo da página -->
<script>escreve_link_correio()</script>
</body>
```

Se o robô do spammer for muito esperto, provavelmente possa colocar em execução o Javascript para interpreta-lo e

saber onde está escondido o endereço de correio. Isso parece no momento pouco provável. Existem tanto endereços nas páginas web, que é possível que eles não se entretenham tanto tempo para obter endereços ocultos no código da página.

Não obstante, certamente existem outras maneiras de ocultar um pouco melhor com Javascript esse endereço. Pode ser que o das variáveis definidas acima de tudo seja um pouco óbvio. Deixo para você investigarem esta tarefa se desejarem. Talvez falaremos dela em um artigo posterior. Enviem seus comentários se tiverem alguma ajuda para melhorar o script.

> **Nota:** Temos outro artigo relacionado:Esconder com CSS o e-mail aos spambots.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Função em Javascript para a inserção de datas

Esta é uma função que realizei para tornar mais fácil a escritura de campos data com formatos de dd/mm/aaaa e uma pequena validação, também apresento uma pequena função para saber se um valor é numérico:

1 devemos ter um formulário html com um campo com a seguinte forma:

<input name="data" type="text" size="10" maxlength="10" onKeyUp = "this.value=formateadata(this.value);">

Aqui se pode observar que fazemos um chamado à função: formateadata

2- no mesmo html colocamos os javascript com as funções ou o colocamos à parte em um arquivo.js para onde deveríamos fazer um chamado.

```
function IsNumeric(valor)
{
var log=valor.length; var sw="S";
for (x=0; x<log; x++)
{ v1=valor.substr(x,1);
v2 = parseInt(v1);
//Comprovo se é um valor numérico
if (isNaN(v2)) { sw= "N";}
}
if (sw=="S") {return true;} else {return false; }
}

var primeiroslap=false;
var segundoslap=false;
function formateadata(data)
{
var long = data.length;
var dia;
var mes;
var ano;

if ((long>=2) && (primeiroslap==false)) { dia=data.substr(0,2);
if ((IsNumeric(dia)==true) && (dia<=31) && (dia!="00")) { data=data.substr(0,2)+"/"+data.substr(3,7);
primeiroslap=true; }
else { data=""; primeiroslap=false;}
}
else
{ dia=data.substr(0,1);
if (IsNumeric(dia)==false)
{data="";}
if ((long<=2) && (primeiroslap=true)) {data=data.substr(0,1); primeiroslap=false; }
}
if ((long>=5) && (segundoslap==false))
{ mes=data.substr(3,2);
```

```
if ((IsNumeric(mes)==true) &&(mes<=12) && (mes!="00")) { data=data.substr(0,5)+"/"+data.substr(6,4);
segundoslap=true; }
else { data=data.substr(0,3);; segundoslap=false;}
}
else { if ((long<=5) && (segundoslap=true)) { data=data.substr(0,4); segundoslap=false; } }
if (long>=7)
{ ano=data.substr(6,4);
if (IsNumeric(ano)==false) { data=data.substr(0,6); }
else { if (long==10){ if ((ano==0) || (ano<1900) || (ano>2100)) { data=data.substr(0,6); } } }
}

if (long>=10)
{
data=data.substr(0,10);
dia=data.substr(0,2);
mes=data.substr(3,2);
ano=data.substr(6,4);
// Ano nao bisexto e é fevereiro e o dia é maior a 28
if ( (ano%4 != 0) && (mes ==02) && (dia > 28) ) { data=data.substr(0,2)+"/"; }
}
return (data);
}
```

*Artigo por **Juan Carlos Montejo***

DHTML Calendar

Quando realizamos uma interface de usuário, é típico ter campos onde o visitante deva introduzir uma data. Estas têm formatos bastante estritos e são complicadas de escrever, sendo então muito cômodo para o usuário contar com a possibilidade de utilizar um calendário para selecionar a data.

Pelo fato de nem sempre o visitante ter a capacidade de entender a programação em PHP, ou aceso a um servidor que permita a publicação de conteúdos programados com PHP, é muito interessante conhecer outras maneiras de implementar um calendário em uma página web. Neste caso vamos apresentar DHTML Calendar, um calendário realizado em Javascript, compatível para todos os navegadores. Este script é gratuito para incorporar um calendário, portanto podemos utilizá-lo sem nenhum tipo de limite.

Como é DHTML Calendar

É um sistema muito potente e facilmente configurável com uma interessante interface, totalmente dinâmica. Pode-se incluir de diversas maneiras dentro de uma página, como um pop-up, ou diretamente no corpo da página, o que o torna muito útil em diversas situações.

O script para configurar o calendário variará de um modo de apresentação a outro. No download do calendário se oferecem alguns exemplos rápidos para mostrar o calendário. Exemplos para os impacientes, que podem ser muito bom para começar rapidamente. Uma das maneiras mais típicas de apresentar o calendário pode ser utilizando um campo de texto e um botão. Ao clicar o botão se mostra o calendário e, uma vez selecionada uma data, se escreve no campo de texto.

O código do exemplo seria o seguinte, muito parecido a um dos exemplos para os impacientes proporcionados no pacote de download.

```
<html>
<head>

  <title>Calendário de provas</title>

  <-Folha de estilos do calendário -->
  <link rel="stylesheet" type="text/css" media="all" href="calendar-green.css" title="win2k-cold-1" />

  <!-- biblioteca principal do calendario -->
 <script type="text/javascript" src="calendar.js"></script>
```

```
 <!-- biblioteca para carregar a linguagem desejada -->
  <script type="text/javascript" src="lang/calendar-es.js"></script>

  <!-biblioteca que declara a função Calendar.setup, que ajuda a gerar um calendário em poucas linhas de código -->
  <script type="text/javascript" src="calendar-setup.js"></script>

</head>

<body>

<!-- formulario com o campo de texto e o botão para lançar o calendário-->
<form action="#" method="get">
<input type="text" name="date" id="campo_data" />
<input type="button" id="lancador" value="..." />
</form>

<!-- script que define e configura o calendario-->
<script type="text/javascript">
   Calendar.setup({
    inputField     :    "campo_fecha",     // id do campo de texto
     ifFormat     :     "%d/%m/%Y",     // formato da data que se escreva no campo de texto
     button     :    "lancador"     // o id do botão que lançará o calendário
});
</script>

</body>
</html>
```

O código anterior está comentado para que se entenda mais facilmente. Tem várias partes.

- No cabeçalho se incluem vários arquivos com as funções e estilos do calendário. Estes arquivos se encontram nos arquivos de download do calendário. Existem vários estilos que se podem utilizar para ajustar o aspecto do calendário ao desenho da página. Também se deve incluir a linguagem na qual apresentar o calendário, neste caso espanhol, que está também incluído no pacote de download.
- Já no corpo da página, se mostra o formulário. É muito simples, pois só tem um campo de texto e um botão para mostrar o calendário.
- No script de javascript, também dentro do corpo da página, se utiliza a função Calendar.setup, que serve para carregar o calendário e configurá-lo com os valores que desejarmos. Todas as opções de configuração têm valores padrão, embora sempre vamos ter que definir, como mínimo, os dados que colocarmos neste exemplo. O dado "inputField" serve para indicar o identificador (atributo id) do campo input onde se tem que escrever a data. O valor "ifFormat" serve para ajustar o formato da data que se deseja escrever no campo de texto. Por último, o valor "button" deve conter o identificador do botão que lançará o calendário ao clicá-lo.

Foi um exemplo o mais simplificado possível. Podemos ver o funcionamento do calendário que gera este código em uma página a parte.

Na documentação do produto poderemos encontrar mais exemplos do calendário e uma explicação detalhada de sua utilização.

Para maiores informações na página do produto: http://www.dynarch.com/projects/calendar/

DHTML Calentar é um projeto hospedado em SourceForge.net, com a seguinte página de projeto: http://sourceforge.net/projects/jscalendar

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Gerar uma cor aleatória com Javascript

Este exercício é muito simples, porém pode ser muito útil para algumas pessoas. Trata-se de uma pequena função que serve para gerar uma cor aleatória, em formato hexadecimal, que é o utilizado na criação de webs.

Extraímos de outro exemplo realizado em Javascript, no qual necessitávamos gerar uma cor de maneira aleatória. Acreditamos que possa ser interessante para comentá-lo em um artigo à parte, se por acaso alguém necessitar criar uma cor totalmente aleatória em suas páginas.

Para criar uma cor aleatório precisamos simplesmente de 6 números em hexadecimal (números do 0 à F). Se obtivermos aleatoriamente estes 6 números hexadecimais, teremos criado o código de uma cor aleatória.

Função para gerar aleatórios

Portanto, para gerar um número aleatório, vamos nos apoiar em uma função relatada em outro artigo de CriarWeb.com: Geração de números aleatórios Javascript .

A função para gerar aleatórios que vamos utilizar então é a seguinte:

```
function aleatorio(inferior,superior){
  numPossibilidades = superior - inferior
  aleat = Math.random() * numPossibilidades
  aleat = Math.floor(aleat)
  return parseInt(inferior) + aleat
}
```

Como nós necessitamos de um aleatório hexadecinal, para nos apoiarmos nesta função, vamos gerar um número aleatório em decimal, que logo converteremos a hexadecimal. Para fazer esta conversão utilizaremos um array de valores hexadecimais como este:

```
hexadecimal = new Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F")
```

Agora, para obter esse valor aleatório hexadecimal, o único que temos que fazer é obter um índice entre 0 e o número de campos deste array. Então, o valor aleatório hexadecimal será o que tiver no array no campo cujo índice se obteve aleatoriamente. Isto se faz desta maneira:

```
posarray = aleatorio(0,hexadecimal.length)
valor_hexadecimal_aleatorio = hexadecimal[posarray]
```

Na primeira linha se obtém o índice do array aleatório, que está entre 0 e o número de posições do array. Na segunda valor_hexadecimal_aleatorio se obterá acessando ao array, na posição gerada aleatoriamente.

Obter a cor aleatória

Agora que já vimos uma maneira de obter um valor hexadecimal aleatório, vamos ver como obter uma cor aleatória, que não é mais do que obter 6 valores hexadecimais aleatórios e concatená-los.

```
hexadecimal = new Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F")
cor_aleatoria = "#";
for (i=0;i<6;i++){
  posarray = aleatorio(0,hexadecimal.length)
  cor_aleatoria += hexadecimal[posarray]
}
```

Depois da execução deste código, a variável cor_aleatoria conterá a cor gerada aleatoriamente.

Colocá-la em uma função

Para acabar esta pequena prática, vamos ver como se pode por tudo isto em uma função, que poderemos utilizar em qualquer contexto.

```
function dar_cor_aleatoria(){
  hexadecimal = new Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F")
  cor_aleatoria = "#";
  for (i=0;i<6;i++){
    posarray = aleatorio(0,hexadecimal.length)
    cor_aleatoria += hexadecimal[posarray]
  }
  return cor_aleatoria
}
```

Para ver este exemplo em funcionamento criamos uma página que mostra uma série de 50 cores gerados aleatoriamente.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

A aprendizagem na Internet

Apesar do computador não ser o remédio do futuro ensino, ninguém põe em dúvida que é um meio didático muito potente que pode mudar a forma de ensinar dos professores e o modo de aprender dos alunos.

Quando o aluno se depara com o computador e com um software educativo ou com a própria Internet, o protagonista já não é o professor, e sim o próprio aluno, quem lê, compreende e aprende por si mesmo.

O professor Jesús A. Beltrán Lera distingue a pedagogia da reprodução passiva e a pedagogia da imaginação ativa e inovadora.

As páginas web também podem ser divididas em duas classes: as que apresentam textos e desenhos que se podem copiar e imprimir (porém são passivas e estáticas) e as páginas interativas (com um diálogo entre o computador e o aluno). No mundo do ensino é muito aconselhável que os exercícios sejam do segundo tipo.

Um modelo interativo pode ser este: A tela lhe apresenta uma "Ajuda" com o conteúdo que o aluno necessita para fazer bem os exercícios; depois o computador lhe apresenta uma pergunta com várias alternativas de resposta e o aluno responde a uma delas; se a resposta for certa o computador lhe pontua mudando a cor da resposta de preto a vermelho e se for falsa lhe informa de seu erro com um alerta dando a informação correta. A interatividade permite a correção instantânea da resposta do aluno, o que supõe um fator motivador.

Ao terminar todas as perguntas, o aluno acessa a um botão de "Pontuação" que ao clicá-lo nos dá três informações: o número de acertos, o número de erros e uma qualificação a partir destes três comentários de avaliação: "Muito bem. Magnífico", se não cometeu erros; "Bem, porém pode melhorar", se a pontuação (de 0 a 10) for de 7 ou superior; e "Deve repetir o exercício", se a pontuação for inferior a 7 pontos. Esta interatividade produz uma realimentação positiva que incrementa a motivação do aluno e estimula a atividade escolar.

Com HTML e Javascript pode-se preparar exercícios como os comentados anteriormente e sobre distintos temas como ortografia, leitura compreensiva, poesias, adivinhação e cálculo.

Vejamos dois exemplos de ortografia em espanhol, um com a ajuda em "alert" e o outro utilizando o "popup".

No primeiro exemplo, colocaremos dentro do cabeçalho da página (entre o e) um script com as funções que controlam os acertos, erros e pontuações:

```
<script LANGUAGE="JavaScript">

var sumafa=0

var sumaaci=0

var res=0

var nota=0

function resbien(sumafa2)
{
  sumaaci=sumaaci+1;
  return true
}

function resmal(sumafa2)
{
  sumafa=sumafa+1;
  return true
}
```

```
function averiguarNota(nota2){
  res=sumaaci+sumafa
  res=res/10
  nota=sumaaci/res
  if (res <=0)
  {
    alert("Puedes empezar este ejercicio. ¡Suerte!")
  }
  else
  if (nota >=10)
  {
    alert("Has tenido "+sumaaci+" aciertos y "+sumafa+" errores. Muy Bien. Magnífico")
  }
  else
  if (nota >7)
  {
    alert("Has tenido "+sumaaci+" aciertos y "+sumafa+" errores. Bien, pero puedes mejorar.")
  }
  else
  {
    alert("Has tenido "+sumaaci+" aciertos y "+sumafa+" errores. Debes repetir el ejercicio.")
  }
sumaaci=0
sumafa=0 }

</script>
```

Depois colocaremos o formulário no corpo da página. Contém o exercício propriamente dito e os botões de "Ajuda", "Pontuação" e "Apagar contestações":

```
<h1 align="center"><input Type="Button" Value="Ayuda" onclick="alert('abrir, advertir, afirmativo, amable, cueva, n
nsubir, obstruir, atrever, avión, bandido.')"></h1>
<form NAME="bv">
<div align="center"><center><table border="8" width="346" cellspacing="1" cellpadding="1">
<tr>
<td width="182" bgcolor="#FFFFFF"><div align="center"><center><p><small><font face="Verdana"
color="#FF0000"><b>a_rir</b></font></small></td>
<td width="152" align="center" bgcolor="#FFFFFF"><b><div align="center"><center><p>
<input Type="Button" Value=" b " onclick="resbien(); this.style.color='red'">
<input Type="Button" Value=" v " onclick="resmal(); alert('abrir, advertir, afirmativo, amable, cueva, n nsubir,
obstruir, atrever, avión, bandido.')">
<input Type="Button" Value=" w " onclick="resmal(); alert('abrir, advertir, afirmativo, amable, cueva, n nsubir,
obstruir, atrever, avión, bandido.')">
</b></td>
</tr>
<tr align="center">
<td width="182" bgcolor="#FFFFFF"><div align="center"><center><p><small><font
face="Verdana" color="#FF0000"><b>ad_ertir</b></font></small></td>
<b><td width="152" align="center" bgcolor="#FFFFFF"><b><input Type="Button" Value=" b "
onclick="resmal(); alert('abrir, advertir, afirmativo, amable, cueva, n nsubir, obstruir, atrever, avión, bandido. ')">
<input Type="Button" Value=" v " onClick="resbien(); this.style.color='red'"> <input
Type="Button" Value=" w "
onClick="resmal(); alert('abrir, advertir, afirmativo, amable, cueva, n nsubir, obstruir, atrever, avión, bandido.')">
</b></b></td>
</tr>
<tr align="center">
<td width="182" bgcolor="#FFFFFF"><div align="center"><center><p><small><font
face="Verdana" color="#FF0000"><b>afirmati_o</b></font></small></td>
<b><td width="152" align="center" bgcolor="#FFFFFF"><b><input Type="Button" Value=" b "
onclick="resmal(); alert('abrir, advertir, afirmativo, amable, cueva, n nsubir, obstruir, atrever, avión, bandido. ')">
<input Type="Button" Value=" v " onClick="resbien(); this.style.color='red'"> <input
Type="Button" Value=" w "
onClick="resmal(); alert('abrir, advertir, afirmativo, amable, cueva, n nsubir, obstruir, atrever, avión, bandido.')">
</b></b></td>
```

```
</tr>
</table>
</center></div><div align="center"><center><p><strong>
<input Type="Button" value="Puntuación" onclick="averiguarNota()">
<input TYPE="SUBMIT" VALUE="Borrar contestaciones" onblur="this.style.color='black'"> </p>
</center></div></form></strong>
```

Podemos ver o exemplo primeiro em funcionamento em uma página à parte.

O segundo exemplo que se propõe tem o mesmo script de pontuações, um script com a função do popup e o código de ortografia de G e J. O código do script do popup é:

```
<script LANGUAGE="JavaScript">
cnt = 0;
var wnd;
function popup(pagina)
{
  cnt++;
  if( wnd != null )
  {
    wnd.close();
    wnd = null;
  }
  wnd = open(pagina, "gl"+cnt, "width=650, left=70, top=128, height=270");
}

</script>
```

E o código do corpo da página:

```
<p align="center"><input TYPE="BUTTON" NAME="Ayuda" VALUE="AYUDA" onClick="popup('frayu05.htm');"
<p> </p>
<form NAME="bv" onsubmit>
<div align="center"><center>
<table border="8" width="100%" cellspacing="1" cellpadding="1">
<tr>
<td width="75%" bgcolor="#FFFFFF"><div align="center"><center><p><small><b><font
face="Verdana">Avísame cuando  <font color="#FF0000">despe_e  </font>el avión</font></b></small></td>
<td width="25%" align="center" bgcolor="#FFFFFF"><input Type="Button" Value=" g " onClick="resmal(); alert('El
sonido G suave con A, O, U, se escribe GA, GO, GU y con E, I, se escribe GUE, GUI. n nEjemplos: goma, galleta,
guapa, Miguel, guitarra, gorro, guerra.')">
<input Type="Button" Value=" gu " onClick="resbien(); this.style.color='red'"> </td></tr>
<tr align="center">
<td width="75%" bgcolor="#FFFFFF"><div align="center"><center><p><small><b><font
face="Verdana">Tu mamá es muy  <font color="#FF0000">ele_ante</font></font></b></small></td>
<td width="25%" align="center" bgcolor="#FFFFFF"><input Type="Button" Value=" g "
onClick="resbien(); this.style.color='red'"> <input Type="Button" Value=" gu "
onClick="resmal(); alert('El sonido G suave con A, O, U, se escribe GA, GO, GU y con E, I, se escribe GUE, GUI. n
nEjemplos: goma, galleta, guapa, Miguel, guitarra, gorro, guerra.')">
</td></tr>
<tr align="center">
<td width="75%" bgcolor="#FFFFFF"><div align="center"><center><p><small><b><font
face="Verdana">Los Reyes Magos me  <font color="#FF0000">tra_eron  </font>una
bicicleta</font></b></small></td>
<td width="25%" align="center" bgcolor="#FFFFFF"><input Type="Button" Value=" g "
onClick="resmal(); alert('Llevan J las formas de los verbos que no tienen G ni J en el infinitivo. n nEjemplos: de decir,
dijeron; de traer, trajimos, trajeron.')">
<input Type="Button" Value=" j " onClick="resbien(); this.style.color='red'"> </td></tr>
</table></center></div>
<div align="center"><center><p><strong>
<input Type="Button" value="Puntuación" onClick="averiguarNota()">
<input TYPE="SUBMIT" VALUE="Borrar contestaciones" onblur="this.style.color='black'"> </p>
</center></div></form></strong>
```

O segundo exemplo pode ser visto em uma página à parte.

*Artigo por **Agustin Jareño***

Menu Dinâmico com Javascript

O leitor já deve ter sentido necessidade de dar dinamismo e agilidade ao seu website. Pois fique desde já a saber que se deve apostar num menu bastante dinâmico e flexível, pois ele é o veículo que vai conduzir cada cibernauta pela rede do seu sítio. Se colocar links atafulhados e desorganizados no seu menu, então é provável que os visitantes desistam após consultarem somente uma página.

Por outro lado, se apresentar um índice sóbrio e bem estruturado, terá muito maiores probabilidades de este ser usado pelos utilizadores e, consequentemente, aumentar os seus registos de tráfego. Agora a escolha é sua! Continuar com um menu estático e menos apelativo? Ou trocá-lo por um que lhe proporcione um aumento dos seus rendimentos?

Se a sua escolha for a acertada (a segunda), então este artigo interessa-lhe, pois descreve esquematicamente, e de forma o mais sucinta e compreensível possível, um processo de criação de menus dinâmicos para colocar no seu Website. O código é em Javascript, pois neste caso uma linguagem de programação do lado do cliente só traz vantagens, pois aumenta a rapidez da aplicação.

Veja um exemplo do seguinte código em acção!

Antes de converter o seu menu, dedique algum tempo para pensar numa possível organização dos links. Uma proposta é separá-los por temas diferenciados, como por exemplo: ?Desporto?, ?Cinema?, ?Teatro?, etc.. De seguida, introduza em cada grupo aqueles que achar adequados ao contexto. Organize os grupos por ordem decrescente de interesse, estando os mais relevantes no topo da página e os menos interessantes em último lugar. Assim, as hiperligações de topo terão mais visibilidade e chamarão mais cibernautas. Findo este processo, pode começar a converter o seu menu.

Antes de mais, deve copiar e colar o seguinte código em Javascript no início do site, pois é ele que vai gerir o seu índice de hiperligações.

```
<script>
var arrayX = new Array ();
var flagX = new Array ();

function treeWatch(id)
{
switch (flagX[id])
{
case "0":
document.getElementById ("subfolder"+id).innerHTML = "";
flagX[id]="1";
break;

case "1":
document.getElementById ("subfolder"+id).innerHTML = arrayX[id];
flagX[id]="0";
break;
}
}
</script>
```

Este código está dividido em duas secções que são delimitadas pelo ?switch?: ou o grupo está visível, ou não está visível. Se o valor da lista for ?0? então o código vai ocultar a parte do menu em foco. Se o valor, por outro lado, for ?1?, nesse caso o script irá mostrar a secção pretendida.

Agora só falta definir o que deseja tornar dinâmico no seu menu e se essa parte deve aparecer oculta, logo de início, ou não.

Nesse sentido, deve colocar a informação da seguinte forma:

```
<script>arrayX[y] = ?códigoHTML?; flagX[y] = "1";</script>
```

Onde se lê y, deve colocar a ordem em que deseja que apareça a secção. Se quiser introduzir um certo tema em

primeiro lugar, tem de colocar no campo y o valor 0. De forma geral, para colocar em ordem n, deve-se indicar o valor n-1.

<script>treeWatch ('visibilidade');</script>

Por outro lado, se desejar que aquele campo deve ser visível à partida, coloque o valor de ordem relativo ao bloco desejado (por exemplo, para activar o 3º bloco, deve introduzir ?2? onde se lê ?visibilidade?). Se desejar que o bloco apareça omisso de início, não coloque a linha de código anterior.
Para finalizar, só falta colocar o seguinte código HTML onde desejar que a secção apareça.

Uma vez mais, o valor y refere-se à ordem em que a secção aparece no menu. Os valores de y devem ser sempre iguais para o mesmo bloco. Por outro lado, o objecto ?img?, ou seja, a imagem, tem um evento ?onclick? que vai mostrar ou esconder (assim que seja clicado) o bloco do menu correspondente. Uma nota breve para referir que não é estritamente necessário que o objecto seja uma imagem. O essencial é implementar algures num objecto o evento **onclick=?treeWatch('y')?**.
Só um aparte para referir que este script já está desenhado no sentido de ser integrado com outras linguagens de programação do lado do servidor. Só para dar uma ideia... Se desejar que certa secção fique activa após um clique específico num certo link, então basta colocar o seguinte código (exemplo em PHP):

<script>treeWatch ('<?=$varVisibilidade?>');</script>

Agora que já possui um menu dinâmico, regozije-se com o seu tráfego a subir!

Veja um exemplo do código em acção!

*Artigo por **João Coelho***

Página que muda aleatoriamente a cor de fundo

Vamos criar uma página que tem uma cor de fundo aleatória, de modo que, cada vez que se visite se mostre com um fundo diferente. Entretanto, como a cor da página vai ser diferente a cada vez, para certificarmos que o texto poderá ser lido corretamente, faremos com que o texto da página seja ou branco ou negro, dependendo da gama da cor de fundo: se for escuro, o texto da página será branco e se o fundo for claro, o texto se verá em negro.

Há que se dar conta que, se a cor é aleatória, às vezes sairá mais escuro e às vezes mais claro. Para que se leia bem o texto, sua cor tem que contrastar o suficiente com a cor de fundo, por isso, calcularemos a obscuridade ou a claridade do fundo para observar a cor do texto.

Pode-se ver em funcionamento o exemplo que será desenvolvido nesta página.

Em um artigo anterior do workshop de Javascript já explicamos uma maneira de conseguir uma cor aleatória em Javascript.

Apesar de no mencionado artigo já estar a função Javascript para obter uma cor aleatória transcreveremos novamente aqui, pois fizemos um par de mudanças minúsculas ao código:

```
function dame_numero_aleatorio(superior, inferior){
  var numPossibilidades = (superior + 1) - inferior;
  var aleat = Math.random() * numPossibilidades;
  aleat = Math.floor(aleat);
  aleat = (inferior + aleat);
  return aleat
}

function dame_cor_aleatoria(){
  color_aleat="#"
  hexadecimal = new Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F")
  var inferior = 0;
  var superior = hexadecimal.length-1;
```

```
  for (i=0;i<6;i++){
    color_aleat += hexadecimal[dame_numero_aleatorio(superior, inferior)];
  }
  return color_aleat
}
```

Agora veremos a maneira de conhecer a obscuridade ou claridade de uma cor aleatória gerada por Javascript. Para calcular a obscuridade (ou claridade) de uma cor em formato RGB hexadecimal, vamos realizar vários passos:

Temos que saber que, a maiores valores de RGB, a cor resultante será mais clara. Se os valores de RGB são mais baixos, a cor será mais escura. Os valores de R, G e B, separadamente podem ir, em decimal, desde 0 a 255. Diremos que será claro quando for maior que 255 / 2 e que é escuro quando for menor de 255 / 2. Vamos supor um umbral a partir do qual consideramos a cor mais escura ou mais clara. Digamos que se somamos separadamente os valores vermelho, verde e azul e dá mais da metade de ((255 + 255 +255) / 2), é que a cor é clara. Se estiver abaixo desse umbral, a cor é escura.

1. Separaremos os valores hexadecimais dos três componentes da cor (vermelho, verde e azul)
2. Converteremos esses valores a inteiros em base 10
3. Somamos os valores de cada cor, obtendo um número inteiro, do que vamos deduzir a claridade ou obcuridade.
4. Se o número resultado da soma for menor que ((255 + 255 + 255) / 2), então é que a cor de fundo é escura, logo a cor do texto deve ser clara. E ao contrário, se deduzimos que a cor de fundo é clara, então a cor de texto terá que ser escura.

Isto se faz da seguinte maneira, em código Javascript, tendo uma cor em um string com o formato #RRGGBB:

```
//obtenho uma aleatoria
colorin = dame_cor_aleatoria()

//vou extrair as tres partes da cor
rojo = colorin.substring(1,3)
verde = colorin.substring(3,5)
azul = colorin.substring(5,7)

//vou converter a inteiros os string, que tenho em hexadecimal
intvermelho = parseInt(vermelho,16)
intverde = parseInt(verde,16)
intazul = parseInt(azul,16)

//agora somo os valores
obscuridade = intvermelho + intverde + intazul

//se o valor obscuridade for menor que ((255 + 255 + 255) / 2) é que é uma cor mais escura
//se for escura, a cor do texto será branca
if (obscuridade<(255+255+255)/2)
  colortexto="#ffffff"
else
  colortexto="#000000"
```

Para atualizar a cor de fundo e de texto de uma página web se poderia fazer com estas linhas de código:

```
document.fgColor=colortexto
document.bgColor=colorin
```

Porém, isto dá um problema em alguns navegadores, ao mudar a cor do texto, que não se pode fazer se previamente se escreveu algo na página.

Então, vamos marcar a cor do fundo e do texto utilizando os conhecidos atributos bgcolor e text da etiqueta <body>.

Escreveremos o <body> mediante javascript, colocando os valores de cor aleatória e cor do texto que extraímos das variáveis que os contém.

```
document.write('<body bgcolor="' + colorin + '" text="' + colortexto + '">')
```

Isso é todo. Já temos a página com a cor de fundo aleatória e a cor do texto com suficiente contraste.

A página onde implementamos este exercício pode-se ver em funcionamento aqui. Podemos ver seu código fonte para obter script do exemplo completo.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Script de recarregamento da página com Javascript

Em algumas ocasiões necessitamos que uma página recarregue seus conteúdos a cada certo tempo, para mostrar informação atualizada às pessoas que a visitam. Isto é Às vezes típico dos serviços que mostram informação em tempo real, segundo se vai produzindo.

Este artigo surge como resposta à dúvida de um visitante que, ademais, necessitava que o tempo que passasse entre cada recarregamento da página fosse sempre diferente. Para isso, simplesmente fazemos que se recarregue a página depois de um número de segundos aleatório.

Um tema que ademais é necessário para que tudo funcione corretamente é que a página não se mostre desde o cachê do navegador. Sabemos que quando uma página já foi solicitada previamente, fica muitas vezes no cache de nossos navegadores de modo que, se se volta a solicitar, se mostra a cópia que temos armazenado localmente, em vez de se solicitar e baixar através do servidor de Internet. Nestes sistemas de recarregamento há que se assegurar que os conteúdos não se obtenham desde o cache, para que as atualizações possam ser vistas corretamente com cada recarregamento da página.

O recarregamento com Javascript

Vejamos como resolvemos todas estas necessidades, neste caso mediante Javascript.

Para começar, temos que obter um número aleatório de segundos, para que o recarregamento se realize a intervalos irregulares, tal como nos solicitavam.

Para isso, vamos utilizar a seguinte função de cálculo de números aleatórios, que comentamos e provamos em outros exemplos relatados em CriarWeb.com.

```
function aleatorio(inferior,superior){
    numPossibilidades = superior - inferior
   aleat = Math.random() * numPossibilidades
   aleat = Math.floor(aleat)
   return parseInt(inferior) + aleat
}
```

Chamaremos à função para obter um número aleatório, neste caso entre 5 e 10:

```
num_aleatorio = aleatorio(5, 10)
```

Para solucionar o tema de que a página não se mostre desde o cache do navegador vamos enviar-lhe um parâmetro pela URL, assim a URL que solicitemos será sempre diferentes e nosso navegador se verá obrigado a solicitar a página ao servidor cada vez que se recarregue. Poderíamos ter utilizado outras técnicas como colocar no cabeçalho do http a ordem para que não se salve no cache, porém por experiência própria, esta é a única maneira que nos assegura que todos os navegadores vão recarregar a página solicitando sempre ao servidor.

Vou gerar um string para enviá-lo por parâmetro a esta mesma página. Como dizíamos, o parâmetro o passaremos pela URL. Não faremos nada com este dado, mas como cada vez será diferente, nos assegura que o navegador sempre solicitará ao servidor a página, em vez de mostrar outra vez a que tem em cache. Utilizaremos a data e tempo para gerar o dato que mude sempre.

```
minhaData = new Date()
dato_url = minhaData.getYear().toString() + minhaData.getMonth().toString() + minhaData.getDate().toString() +
minhaData.getHours().toString() + minhaData.getMinutes().toString() + minhaData.getSeconds().toString()
```

Na variável dado_url salvamos o ano, seguido do mês, dia, horas, minutos e segundos. Como todos os dados de data, se têm que extrair desde um objeto date, que criamos com a sentença new Date(). Logo, a este objeto lhe invocamos diversos métodos para obter os dados que necessitamos. Os dados são devolvidos em tipo inteiro e para concatená-los como se fossem string, necessitamos aplicar o método toString(), que têm todos os objetos de Javascript para

convertê-los em cadeias.

Já só falta realizar o recarregamento propriamente dito. Para isso, temos que aplicar um atraso, que conseguiremos com a função setTimeout(), que recebe como primeiro parâmetro a instrução que se quer executar e como segundo parâmetro, o tempo em milésimos de segundos que se quer esperar.

setTimeout("window.location='pagina.html?parametro=" + dado_url + "'", num_aleatorio * 1000)

Se observarmos, utilizamos window.location para atribuir uma nova URL ao navegador. Logo, utilizamos a variável dado_url para passá-la como parâmetro. Ademais, para marcar o atraso entre recarregamentos utilizamos a variável num_aleatorio, multiplicada por 1000 para passar a milésimos de segundos.

Isto é tudo. Pode-se ver o exemplo em funcionamento neste link.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Mudar a cor às células de uma tabela com Javascript

Veremos neste artigo um par de soluções a uma pergunta típica de trabalho com Javascript: mudar a cor à célula de uma tabela dinamicamente e como resposta a ações do usuário. Este exemplo pode ser realizado muito facilmente e ofereceremos um par de soluciones, que podemos aplicar dependendo do caso concreto em que nos encontremos.

A primeira solução que vamos ver é como mudar a cor de uma célula ao passar o mouse por cima. Quando entrarmos com o mouse na célula deve mudar para uma cor e quando sairmos a mudaremos por outra.

O segundo caso que vamos realizar é mudar a cor da célula como resposta a eventos de elementos que estão fora da própria célula. Em concreto, mudaremos a cor de uma célula quando se acione um formulário que está fora das células a mudar.

Mudar a cor de uma célula ao passar o mouse por cima

É um caso muito típico que se pode ver em muitas webs. Este exemplo é bastante simples de fazer com CSS, sem necessidade de escrever nem uma linha de código Javascript, porém, apesar disso o mostraremos aqui.

No exemplo teremos uma tabela com várias células e para cada uma teremos definido o evento onmouseover e onmouseout, para chamar a uma função que se encarregue de mudar a cor quando se entre na célula e quando se saia, respectivamente.

A tabela terá esta forma:

```
<table width=100>
<tr>
  <td bgcolor="#dddddd" id="celula1" onmouseover="mudar_cor_over(this)"
onmouseout="mudar_cor_out(this)">Campo numero 1</td>
</tr>
<tr>
  <td bgcolor="#dddddd" id="celula2" onmouseover="mudar_cor_over(this)"
onmouseout="mudar_cor_out(this)">Campo numero 2</td>
</tr>
...
<tr>
  <td bgcolor="#dddddd" id="celula10" onmouseover="mudar_cor_over(this)"
onmouseout="mudar_cor_out(this)">Campo numero 10</td>
</tr>
</table>
```

As funções que se encarregam de alterar a cor recebem por parâmetro a palavra this, que é uma referência à célula onde está o evento. Se estamos no evento da célula 1, this faz referência a essa mesma célula 1. As funções são as seguintes:

```
function mudar_cor_over(celula){
  celula.style.backgroundColor="#66ff33"
```

```
}
function mudar_cor_out(celula){
  celula.style.backgroundColor="#dddddd"
}
```

Como se pode ver, recebem a célula cuja cor se deseja mudar (que se enviou com a palavra this no manejador do evento). Logo, executam a sentença necessária para mudar a cor.

Mudar a cor de uma célula sem passar a referência this

O segundo caso que havíamos adiantado que íamos fazer, era mudar a cor a uma célula desde outra parte da página. Vimos que, se estamos na mesma célula, podemos enviar uma referência do próprio campo com a palavra this, porém se não estamos codificando um evento da célula, e sim estamos em outro lugar do código da página, não teremos possibilidade de enviar essa simples referência.

Então, nos torna necessário obter a referência da célula por outro mecanismo. Entra em jogo a função de Javascript (que em realidade é um método do objeto document) chamada getElementById(). Esta função recebe o nome de um identificador e devolve uma referência ao elemento que tem esse identificador.

Atribui-se identificadores aos elementos de HTML com o atributo id. Desta maneira:

```
<td id="celula1">
```

Podemos ver no código da tabela, escrito linhas acima, que cada célula tem um identificador definido. Utilizaremos esse identificador para obter a referência à célula que desejamos alterar sua cor.

Por exemplo, se quisermos obter uma referência à célula com identificador "celula1", utilizaríamos a chamada a essa função assim:

```
celula = document.getElementById("celula1")
```

Logo, com a referência da célula, podemos mudar a cor como vimos antes:

```
celula.style.backgroundColor="#dddddd"
```

Em nosso exemplo para ilustrar este método criamos um formulário com dois campos de texto e um botão. No primeiro campo de texto escreveremos o número da célula cuja cor queremos mudar. No segundo, escreveremos o nome da cor que queremos colocar ao campo, ou seu código RGB. Quando apertarmos ao botão, chamaremos a uma função que se encarregará de mudar a cor da célula.

O formulário terá um código como este:

```
<form name=fcolor>
Numero de celula: <input type=text name=celula size=3>
<br>
Cor: <input type=text name=minhacor size=8>
<br>
<input type=button value="Mudar cor" onclick="muda_cor()">
</form>
```

E a função javascript que se encarregará de mudar a cor terá este código:

```
function muda_cor(){
  celula = document.getElementById("celula" + document.fcolor.celula.value)
  celula.style.backgroundColor=document.fcolor.minhacor.value
}
```

Como se pode ver, para obter a referência utilizamos a função document.getElementById() e passamos o id da célula que queremos mudar sua cor. O identificador da célula se compõe pela palavra "celula" e o número da célula, que tiramos do formulário.

Logo, se coloca na célula a cor que se tira do outro campo do formulário.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Pop-ups DHTML – OpenPopups

Todos nós já sabemos que a maioria dos navegadores dispõe de sistemas para bloquear os incômodos pop-ups, e quando estes não os bloqueiam, existem barras de navegação, como a de Google, que também bloqueia a apresentação de pop-ups. A maioria das vezes, estes pop-ups são muito incômodos e temos que comemorar porque agora a maioria se pode detectar e não permitir sua abertura, porém muitos de nossos sites utilizam este sistema para mostrar informação legítima que nossos visitantes deveriam conhecer.

De qualquer forma, existem métodos para mostrar pop-ups que possam ser mais complicados de bloquear, como os pop-ups DHTML, que são uma emulação das janelas secundárias, porém que funciona por camadas e HTML dinâmico para mostrar ou ocultar seu conteúdo. Este tipo de pop-ups não se considera como janelas secundárias, por isso não se bloqueiam.

Somente os navegadores que tenham Javascript desabilitado deixarão de mostrar estes pop-ups. Lembremos que depende de como esteja configurado Internet Explorer, às vezes te mostra uma mensagem de alerta quando se intenta executar um script em Javascript. O usuário é o responsável de permitir, ou não, executar scripts na página. Por isso, não é tão raro que inclusive os pop-ups DHTML se possam bloquear, mas pelo menos significam um avance com respeito às janelas secundárias habituais.

OpenPopups

Todo o anterior serve para apresentar um script Javascript Open Source (gratuito e de código livre) para criar Pop-ups DHTML. Vale a pena conhecer este script, porque certamente pode ser muito interessante para nossas páginas web.

A web onde se pode baixar o sistema de pop-ups DHTML é: http://www.openwebware.com/products/openpopups/

Desde tal web se podem baixar os arquivos necessários para a instalação do sistema de pop-ups e algum código de exemplo. De qualquer forma, explicaremos aqui em português, para que todos possam entender.

Referência: Em CriarWeb.com publicamos alguns outros artigos sobre como fazer um pop-up DHTML, porém utilizando a biblioteca Cross-Browser. Pode ser de interessante leitura para quem quiser se aprofundar no tema ou encontrar outras possibilidades para realizar pop-ups DHTML. Está em nosso manual de Workshop de Cross-Browser DHTML.

Tem-se que descompactar os arquivos que se baixam da web, mantendo a mesma estrutura de diretórios.

Uma vez estando descompactados, em um diretório dentro de nosso web site, que chamaremos, por exemplo, "d_openpopups", já podemos acessá-los através de qualquer página para mostrar pop-ups DHTML. Para isso, o primeiro é incluir o Javascript com a biblioteca.

```
<script language="JavaScript" type="text/javascript" src="/d_openpopups/openpopups/openpopups.js"></script>
```

Onde "d_openpopups" deve ser o diretório onde descompactamos os arquivos. Tal como está escrita a rota para o script, se supõe que colocamos este diretório na raiz do diretório de publicação da web.

Logo, temos que adicionar um evento onload na etiqueta <body>, para ocultar os pop-ups ao carregar a página.

```
<body onload="hideDiv()">
```

A função hideDiv() recebe o número de popups que vamos utilizar na página. Se tivermos um só pop-up DHTML chamaremos passando um 1 como parâmetro: hideDiv(1). Se tivermos 5 pop-ups DHTML, lhe passaremos um 5 como parâmetro: hideDiv(5).

A seguir, temos que criar as camadas com o código fonte dos pop-ups a mostrar. Algo como:

```
<div id="Div1">
   Código do Popup
</div>
```

Há que observar que a camada tem como identificador (atributo id) "Div1". Isso é para o pop-up 1. Se tivéssemos outros pop-ups, deveríamos dar-lhes nomes com números consecutivos: Div2, Div3…

Para acabar, temos que fazer a chamada à função Javascript que deve mostrar o pop-up. Essa função se chama createWindow() e recebe vários parâmetros:

1. Título da janela
2. Largura da janela (a altura será a necessária para que caiba todo o conteúdo)
3. Cor de fundo da janela

4. O identificador da camada (só o número, 1, 2, 3…)
5. Se quisermos que se mostre o ícone para minimizar (1 para mostrá-lo e 0 se não quisermos que se mostre)
6. A posição "left" da janela (o número de pixels à esquerda da janela)
7. A posição "top" da janela (o número de pixels que deve ter acima da janela.

Por exemplo, uma chamada possível a esta função seria:

createWindow('Título', 300, '#ffff88', 1, 0, 100, 25);

Um detalhe que para nós fez falta mudar para que tudo funcionasse corretamente, embora não tenha visto explicado nada disto na documentação do produto, são os diretórios das imagens e as declarações de estilos que utilizam os pop-ups DHTML. Esses diretórios vêem especificados no arquivo de scripts javascript chamado openpopups.js.

Nas seguintes linhas do código se especificam os diretórios das imagens e os CSS:

```
// CSS Diretory
cssDir = "/d_styles/";

// Images Directory
imageDir = "images/";
```

À princípio, segundo entendo, não teria porquê tocar nessas linhas, porque não modifiquei a estrutura de diretórios do arquivo de download, porém se não as toca os exemplos não funcionam corretamente. Para que as rotas se encontrem, tive que colocar a estrutura de diretórios desde a raiz do domínio até as pastas onde estão os arquivos CSS e as imagens. Seria algo como isto:

```
// CSS Diretory
cssDir = "/d_openpopups/openpopups/styles/";

// Images Directory
imageDir = "/d_openpopups/openpopups/images/";
```

Código completo

Vamos mostrar o código de uma página que tem dois pop-ups DHTML e com um par de métodos de carregamento dos popups, um por meio do botão e outro por meio de um link.

```
<html>
<head>
   <title>Exemplo OpenPopups</title>
   <script language="JavaScript" type="text/javascript" src="/d_openpopups/openpopups/openpopups.js">
   </script>
</head>
<body onLoad="hideDiv(2);">
Esta página mostra um par de pop-ups DHTML.
<br>
<br>
Esperamos que sejam interessantes.
<form>
   <input type="button" value="Abrir Popup DHTML 1" onClick="createWindow('Ejemplo 1', 150, '#ffff88', 1, 1, 20,
40);">
</form>
<p>
Agora vejamos o exemplo 2, abertura com um link:
<a href="#" onClick="createWindow('Anuncio MercadoProfesional.com', 468, '#EEEEEE', 2, 0, 240, 165);">Abre o
segundo popup</a>

<div id="Div1">
   <div style="border: 1px solid #ff8800; background-color: #FFFF88; padding: 5px;">
   <b>Aqui poderíamos colocar tanto texto quanto quisermos! E todo tipo de conteúdo HTML!
   </div>
   <ul>
   <li>Com listas</li>
   <li>Links</li>
   <li>Tabelas</li>
   <li>...</li>
   </ul>
</div>
```

```
<div id="Div2">
    <div align="center"><a href="http://www.mercadoprofesional.com" target="_blank"><img src="mp468.gif"
width="468" height="60" border="0" alt="MercadoProfesional.com"></a></div>
</div>

</body>
</html>
```

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Validar a extensão de um arquivo a subir com Javascript

Neste artigo vamos mostrar como validar a extensão de um arquivo mediante Javascript. Temos um formulário com um campo file e quando se envia o arquivo, se realiza uma comprovação para ver se a extensão está entre as permitidas. Se estiver, se realiza o envio do formulário para fazer o upload do arquivo.

Neste script só se realiza a comprovação da extensão, em nenhum caso a recepção do arquivo e seu armazenamento no servidor, pois com Javascript não podemos realizar essas ações, já que é uma linguagem que se executa no cliente e não tem acesso ao servidor para fazer um upload.

Neste exemplo vamos definir um formulário com um campo file e um botão de enviar que chama a uma função que será encarregada de comprovar se a extensão está permitida e submeter o formulário se estiver tudo correto. O formulário seria o seguinte:

```
<form method=post action="#" enctype="multipart/form-data">
<input type=file name="arquivoupload">
<input type=button name="Submit" value="Enviar" onclick="comprova_extensao(this.form,
this.form.arquivoupload.value)">
</form>
```

Agora veremos a função comprova_extensao() que recebe uma referência ao formulário e a rota do arquivo que desejamos subir desde dentro de nosso computador. A função realizará uma série de comprovações que veremos a seguir. O código será o seguinte:

```
function comprova_extensao(formulario, arquivo) {
  extensoes_permitidas = new Array(".gif", ".jpg", ".doc", ".pdf");
  meuerro = "";
  if (!arquivo) {
    //Se não tenho arquivo, é porque não se selecionou um arquivo no formulário.
     meuerro = "Não foi selecionado nenhum arquivo";
  }else{
    //recupero a extensão deste nome de arquivo
    extensao = (arquivo.substring(arquivo.lastIndexOf("."))).toLowerCase();
    //alert (extensao);
    //comprovo se a extensão está entre as permitidas
    permitida = false;
    for (var i = 0; i < extensoes_permitidas.length; i++) {
      if (extensoes_permitidas[i] == extensao) {
      permitida = true;
      break;
      }
    }
    if (!permitida) {
      meuerro = "Comprova a extensão dos arquivos a subir. \nSó se podem subir arquivos com extensões: " +
extensoes_permitidas.join();
     }else{
       //submeto!
      alert ("Tudo correto. Vou submeter o formulário.");
      formulario.submit();
      return 1;
     }
  }
```

```
  //se estou aqui é porque não se pode submeter
  alert (meuerro);
  return 0;
}
```

O primeiro que fazemos é definir um array com as extensões permitidas para fazer o upload. Também definimos uma variável chamada meuerro, onde vamos salvar o texto explicativo do erro, se é que se produz.

Logo, comprovamos se recebemos uma rota do arquivo. Se não houver tal rota, se define o erro correspondente "Não foi selecionado nenhum arquivo". No caso contrário é que temos uma rota, com o qual vamos buscar o nome do arquivo.

A rota que podemos receber pode ter uma forma como esta:
C:\diretorio\outro diretorio\arquivo.doc

Da rota nos interessa obter só a extensão do arquivo. Por isso vamos obter a parte que há depois do último ponto. Isto se faz utilizando vários métodos dos objetos string de Javascript:

```
extensao = (arquivo.substring(arquivo.lastIndexOf("."))).toLowerCase();
```

Simplesmente estamos selecionando a parte do string que há depois do último ponto. E estamos passando a extensão a minúsculas, no caso em que esteja escrita com maiúsculas.

A seguir, para comprovar se esta extensão está entre as permitidas fazemos um loop for, que percorre todo o array de extensões permitidas e vai comparando-as à extensão que recortamos do nome do arquivo. No momento que encontra uma coincidência se sai do loop e põe a variável boleana permitida a true. Se não encontrasse coincidências essa variável boleana ficaria como false.

Logo, se comprova a variável boleana permitida. Se estiver em false é que não se permite a extensão, então defino o correspondente erro. Se estiver a true é que a extensão figurava entre as permitidas, então se envia o formulário e se sai da função.

Ao final da função se mostra o possível erro que tiver detectado. Só se mostrará o erro se não se chegou a mandar o formulário, porque se tivesse sido enviado, se sairia anteriormente da função.

Esperamos que esta validação tenha sido de utilidade. Pode-se ver em funcionamento em uma página a parte.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Detectar a resolução da tela do usuário com Javascript

Com Javascript se pode calcular a resolução de tela do usuário que nos visita. Atenção, não nos referimos ao tamanho da janela do navegador, e sim ao tamanho total em pixels que tivermos configurado em nosso sistema.

As resoluções de tela podem ser valores como 800x600, 1024x760, 1280x800... e se configuram pelo usuário no painel de controle, em propriedades de tela.

Nós com Javascript podemos acessar a esses valores através do objeto screen:

Com screen.width obtemos a largura em pixels da definição de tela.
Com screen.height obtemos a altura em pixels.

Então, se quisermos escrever na página os valores de largura e altura da resolução de tela, poderíamos utilizar um javascript como este:

A resolução atual de sua tela é:

```
<script language="JavaScript">
document.writeln(screen.width + " x " + screen.height)
</script>
```

Se desejarmos, podemos fazer diferentes coisas dependendo da definição de tela do usuário, por exemplo, com uma

estrutura if:

Consideramos sua tela:

```
<script language="JavaScript">
if (screen.width<1024)
  document.write ("Pequena")
else
  if (screen.width<1280)
    document.write ("Media")
  else
    document.write ("Grande")
</script>
```

Este código mostrará o tamanho da janela como pequena (menos de 1024 pixels de largura), media (Maior ou igual a 1024 pixels de largura e menor de 1280) ou grande (1280 pixels de largura ou mais). Porém, poderíamos ter feito outras cosas diferentes dependendo da resolução detectada.

Os exemplos deste artigo pode-se ver em uma página à parte.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Esconder a URL de um link na barra de estado

Quando colocamos o mouse em cima de um link se mostra na barra de estado do navegador a URL a qual vai dirigido. Isto é algo que resulta muito útil quando navegamos, porque podemos ver, antes de clicar o link, o endereço ao que nos dirigirá o navegador se clicamos. Porém, por muito útil que resulte aos navegantes, às vezes os webmaster por umas razões ou por outras preferem que não se veja a URL que nos enviará o link ao colocar o mouse em cima.

Com Javascript podemos alterar o texto que aparece na barra de estado do navegador em qualquer momento, portanto será ideal para esconder o texto que aparece na barra de estado.

> **Nota:**A decisão de eliminar a URL que aparece na barra de estado ao se colocar em cima de um link é uma má idéia desde o ponto de vista da usabilidade. Todos nós utilizamos essa informação para ter uma referência e eliminá-la pode ser incômodo para o visitante.

Temos duas maneiras de esconder o texto da barra de estado. A primeira seria modificando a etiqueta do link, porém teríamos que fazer link a link para todos os que se deseje. Também mostraremos um modo de fazer isso para todos os links da página de uma só vez.

Esconder o texto da barra de estado de link a link

Simplesmente vamos atribuir um comportamento quando se pose o mouse em cima de um link e outro comportamento para quando se saia do link.

```
<a href="http://www.guiarte.com" onmouseover="window.status='Guiarte, sitio de turismo y arte';return true"
onmouseout="window.status='';return true"> Guiarte.com
```

Se vemos esta etiqueta do link se comprovará que tem dois eventos definidos:

- onmouseover, para definir ações quando se pose o mouse sobre o link.
  Neste evento indica com window.status um novo texto para a barra de estado. Logo, fazemos o return true para que não se realize nenhuma ação adicional por este evento.
- onmouseout, para definir ações quando o mouse sai do link.
  Neste evento apagamos o texto da barra de estado que aparecia ao posar sobre o link.

A vantagem deste modo é que podemos colocar um texto diferente na barra de estado para cada link da página. Como dizíamos, a desvantagem é que temos que fazê-lo em cada link que quisermos evitar que se veja a URL.

Pode-se ver um exemplo em uma página a parte.

Nota: Na configuração pré-determinada de Firefox não se permite alterar o texto da barra de estado, por isso este script não mudará esse texto. Porém, como temos o "return true" no manejador do evento, pelo menos evitará que se veja a URL do link.

Ocultar o texto da barra de estado para todos os links

Agora vejamos outro método de fazer isto, de uma só vez para todos os links que tiver na página. Simplesmente vamos fazer um código para apagar o texto da barra de estado, que vai se executar indefinidamente cada intervalo de tempo. Assim, apesar de aparecer a URL do link na barra de estado durante uns instantes, nosso código se executará a cada pouco para apagá-lo.

Vejamos a seguinte instrução Javascript:

```
setInterval ("window.status = ''",10);
```

Isto é uma chamada ao método de window setInterval(), que serve para executar um código Javascript indefinidamente em intervalos definidos. O primeiro parâmetro é a instrução que vai executar window.status = '', que serve para apagar o texto da barra de estado. O seguinte parâmetro são os milésimos de segundos que têm que transcorrer entre execuções da instrução, neste caso 10 milésimos de segundos.

Se colocamos esta instrução em um script em qualquer parte da página, preferivelmente no cabeçalho, faremos que desapareça o escrito na barra de estado em questão de instantes.

```
<script language="JavaScript">
setInterval ("window.status = ''",10);
</script>
```

Pode-se ver em funcionamento o script aqui.

Nota: Na configuração padrão de Firefox não se permite mudar o texto da barra de estado, por isso este script parecerá ter nenhum efeito.

Para encontrar mais informação sobre como mudar esta configuração de Firefox consultando a FAQ: Por que não se muda o texto da barra de estado em Firefox com Javascript?

Conclusão

Embora o texto da barra de estado seja útil, talvez prefiramos que não apareça, ou que se mostre uma mensagem personalizada. Espero que estas duas soluções sejam úteis para esses casos.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Como integrar conteúdo RSS em minha página?

Um dos benefícios que o uso de RSS nos pode oferecer é o poder integrar o conteúdo de Feeds dentro de nossas páginas o qual nos traz informação atualizada para manter nossas páginas ao dia com o qual sempre manteremos aos nossos visitantes com informação renovada e sem a necessidade de nos dedicar a manter o site, pois para a elaboração deste processo há várias opções no mercado as quais podemos desenvolver nós mesmos, comprá-las ou inclusive conseguir código reutilizável que se consegue pela rede e GRÁTIS. Comentarei esta oportunidade pois consegui em uma página um leitor de RSS feeds gratuito e tão simples como copiar 2 linhas de um Javascript e integrá-lo em seu código e logo coloca a página de onde vai tomar a informação e pronto, já tem conteúdo atualizado em sua página, ótimo não é?

Anexo aqui as linhas que devem ser incluídas em seu site, o leitor está desenvolvido em php.

Coloquem estas linhas de código:

```
<script language="JavaScript" type="text/JavaScript" src="http://arupbhanja.com/rssfeed.php?
```

file=http://construcanarias.bitacoras.com/rss1.xml&max=10"> </script>

Explicando

Indica que vem código javascript…
<script language="JavaScript" type="text/JavaScript"

É o endereço de onde tira o leitor de feeds…
src="http://arupbhanja.com/rssfeed.php?

O endereços do feeds…
file=http://construcanarias.bitacoras.com/rss1.xml

Quantidade máxima de tópicos a mostrar, neste caso são 10, mas se pode modificar…
&max=10"></script>

Recomendo que também visitem Feeddigest (em inglês).

*Artigo por **César Pietri***

Fazer com que um iframe se ajuste à altura de uma janela com Javascript

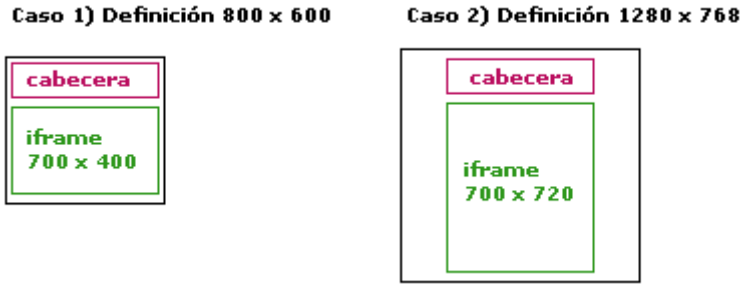
Tenho uma página que tem um iframe e quero que ocupe o espaço máximo disponível, porém não disponho de toda a página, porque há outros conteúdos na página. Ademais, como às vezes a janela do navegador é maior ou menor, o espaço que posso atribuir ao iframe é diferente.

Neste workshop de Javascript vamos realizar um cálculo do espaço disponível na página para que um iframe que temos dentro ocupe a maior área possível. Tudo tendo em conta que cada usuário pode entrar com uma definição de tela distinta e com um navegador distinto.

> **Nota:** Lembramos que um iframe é um frame que se pode inserir no corpo de uma página, atribuindo uma altura e uma largura.

Primeiramente, gostaria que se entendesse bem o problema com o qual me encontro, ao não saber que área há disponível na página para cada usuário que nos visita.

Vejamos esta imagem, que nos pode esclarecer rapidamente o caso no qual nos encontramos.



Imaginemos uma definição de 800 x 600. Então, o espaço para o iframe será o tamanho útil onde se visualiza a página, menos o espaço reservado para o cabeçalho. Agora, por exemplo, em uma definição de 1280 x 768, como o espaço útil para a página é maior, o espaço no qual quero que se veja meu iframe também será maior. Continua sendo o tamanho útil onde se visualiza a página, menos o espaço reservado para o cabeçalho, porém como agora o espaço útil é maior, o iframe também tem que se apresentar com maior tamanho.

A solução passa por utilizar um Javascript para calcular o espaço útil da página e diminuir o espaço do cabeçalho. Então, teremos a dimensão altura que tem que ter o iframe.

Para calcular este dado temos que ter em conta que Internet Explorer e Firefox têm modos distintos. Ou seja, a propriedade espaço útil da página é distinta nestes dois browsers, por isso o script pode complicar um pouco.

Em Internet Explorer: o espaço útil se calcula com a propriedade document.body.clientHeight.

Em Mozilla Firefox: o espaço útil nos devolve a propriedade window.innerHeight

Com este script podemos calcular o tamanho que devemos reservar ao iframe:

```
if (window.innerHeight){
  //navegadores baseados em mozilla
  espaco_iframe = window.innerHeight - 110
}else{
  if (document.body.clientHeight){
    //Navegadores baseados em IExplorer, pois nao tenho innerheight
     espaco_iframe = document.body.clientHeight - 110
  }else{
     //outros navegadores
     espaco_iframe = 478
  }
}
```

O primeiro if serve para os navegadores Firefox, Netscape e similares, que têm a propriedade window.innerHeight

O segundo if é para IExplorer que conhece document.body.clientHeight.

Nos dois casos temos que diminuir 110, que é o espaço que ocupa o cabeçalho. O último if é no caso de que o javascript não entenda nenhuma das duas propriedades, para lhe dar um valor padrão.

Logo, escreveríamos mediante javascript a etiqueta iframe com os dados obtidos previamente:

```
document.write ('<iframe frameborder="0" src="minhapagina.html" width="770" height="' + espaco_iframe + '">')
document.write ('</iframe>')
```

E o que aconteceria se os navegadores não entenderem Javascript, ou estiverem desabilitado?

Nesse caso nos convém utilizar a etiqueta noscript, para mostrar um iframe com os valores por padrão (noscript só se tem em conte se não houver suporte para javascript):

```
<noscript>
<iframe frameborder="0" src="minhapagina.html" width="770" height=478>
</iframe>
</noscript>
```

O código completo seria o seguinte:

```
<script>
if (window.innerHeight){
  //navegadores baseados em mozilla
  espaco_iframe = window.innerHeight - 110
}else{
  if (document.body.clientHeight){
     //Navegadores baseados em IExplorer, pois nao tenho innerheight
     espaco_iframe = document.body.clientHeight - 110
  }else{
     //outros navegadores
     espaco_iframe = 478
  }
}
```

```
document.write ('<iframe frameborder="0" src="minhapagina.html" width="770" height="' + espaco_iframe + '">')
document.write ('</iframe>')
</script>
<noscript>
<iframe frameborder="0" src="minhapagina.html" width="770" height=478>
</iframe>
</noscript>
```

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

É vantajoso o uso de ParseInt para validar números?

A utilização do parseInt para validar números em muitos casos não costuma ser a solução mais efetiva, devido a que permite a presença de letras e/ou espaços, e o resultado poderia não ser o esperado.

Por que parseInt pode causar problemas?

Esta pergunta se responde a si mesma vendo vários exemplos sobre o funcionamento de parseInt:

- "123456": este String retorna como resultado o número 123456 o qual é o resultado esperado.
- "123456asd": este String retorna como resultado o número 123456 apesar de que o String continha letras (vantagem ou desvantagem?).
- "asd": este String retorna como resultado NaN o qual é o resultado esperado.
- "": este String vazio retorna como resultado NaN o qual é o resultado esperado.
- " 123456asd": este String (que contém vários espaços no início do número e letras no final) retorna como resultado o número 123456 (vantagem ou desvantagem?).
- " 123 123 asd" este String (que contém espaços e letras) retorna como resultado o número 123 (vantagem ou desvantagem?).

Como se pode observar, parseInt apresenta o seguinte comportamento:

1. Retornará um número válido si: O String começa por um número.
2. O String começa por espaço(s) seguido de um número.
    - Exemplos de números válidos: "123456"
    - " 123456"
    - "12345asdasd"
    - " 12345 asdd"
3. Todo String que cumpra com as 2 regas anteriores (ser um número válido), será truncado: quando se encontre uma letra, espaço ou caracteres especiais (vírgulas, acentos,...) dentro do String. Como resultado, retornará os dígitos que estejam mais à esquerda da primeira letra (espaço ou caractere) encontrada.
    - Exemplos de números válidos truncados: "123456" retorna como resultado 123456
    - " 123456" retorna como resultado 123456
    - "12345asdasd" retorna como resultado 12345
    - " 123.. asdd" retorna como resultado 123

Uma alternativa ao parseInt, que valida que os String contenham só números está a seguir:

```
function validarNumero(c_numero)
{
  //checar a longitude de c_numero:
  // Si (c_numero.length é igual a Zero) quer dizer que c_numero é uma cadeia Vazia.
  // Se (c_numero.length é diferente(maior) de Zero) podemos assegurar que c_numero contém pelo menos uma
letra
  //a qual se pode fazer a validação
  if (c_numero.length == 0)
  {
     return "NaN";
  }
  else
  {
     //Percorre-se c_numero por todos seus caracteres checando que todos sejam dígitos
     //a condição >="0" e <="9" é baseada no valor ascii que têm os números na tabela ascii.
     //Se algum dos caracteres não for um número a função retornará um NaN
     //Senão retornará o Número
```

```
    for (i = 0; i < c_numero.length; i++)
    {
      if (!((c_numero.charAt(i) >= "0") && (c_numero.charAt(i) <= "9")))
      return "NaN";
    }
    return c_numero;
  }
}
```

Exemplos de validação de números

utilizando a função ParseInt:

Resultado de aplicar a função:

utilizando a função validarNumero (chamando à função validar):

Resultado de aplicar a função:

utilizando a função validarNumero (chamando à função validarComplexo):

Resultado de aplicar a função:

*Artigo por **José Antonio Jiménez Garelli***

Efeito para desabilitar/habilitar o fundo da Página

Bom, não é uma janela emergente como tal, é mais como um simulacro, porém que faz às vezes de janela emergente e sem perigo de que o navegador bloqueie tal janela. É necessário ter conhecimentos (pelo menos básicos) de:

- HTML - podem ver a sessão de HTML de CriarWeb clicando Aqui.
- CSS - podem ver a sessão de CSS de CriarWeb clicando Aqui.
- JavaScript - podem ver a sessão de JavaScript de CriarWeb clicando Aqui.
- Para o exemplo deste artigo iremos utilizar imagens transparentes com diferentes níveis de opacidade, pelo qual é necessário que saiba utilizar um editor de imagens (photoshop, firework, ...) para criar as imagens transparentes ao seu gosto. No caso de que não tenha um editor de imagens, você pode utilizar as do exemplo sem nenhum problema.

Pode-se ver o exemplo em funcionamento Aqui, assim terá uma idéia mais clara do que vamos fazer. Todo o artigo se baseia na explicação do exemplo. O exemplo foi comprovado no Internet Explorer versão 6 e 7, e no Mozilla FireFox 2, e todos com resultados positivos.

Explicação do Exemplo:

Finalmente a parte boa deste artigo!!

A grosso modo, o corpo principal (body) do arquivo Html do exemplo, está composto por 3 parágrafos os quais têm a finalidade de encher a página, conseguindo assim que se veja carregada de informação. Ao final de cada parágrafo há um link o qual é o encarregado de mostrar a janela emergente através de código feito em JavaScript . Esta é a única diferença significativa entre os 3 parágrafos (a chamada à função JavaScript). O código do primeiro parágrafo é o seguinte:

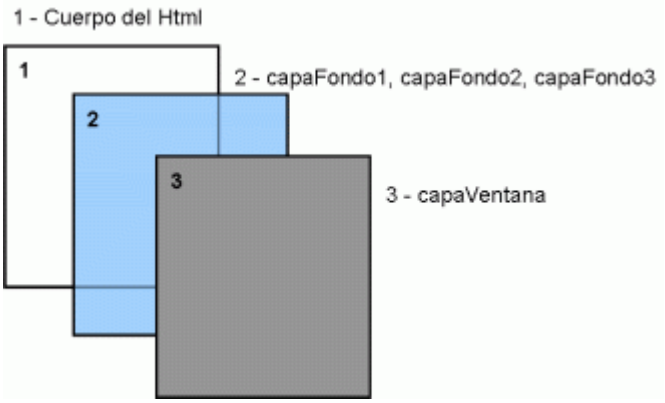
```
<p>
Este é o conteúdo do primeiro parágrafo, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla.<br>
Este é o conteúdo do primeiro parágrafo, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla.<br>
Este é o conteúdo do primeiro parágrafo, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla.<br>
Este é o conteúdo do primeiro parágrafo, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla.<br>
<a href="javascript:abrirJanela('1');">Simulacro de janela Emergente 1</a>
<p>
```

Uma vez que tiver visto os 3 parágrafos no código do exemplo, o que verá a seguir são 4 camadas (div) chamadas camadaJanela, camadaFundo1, camadaFundo2 e camadaFundo3. Apliquei a estas camadas código CSS e à princípio todas estão ocultas. A camada camadaJanela é a mais complexa de todas e é porque nela está o código do que chamei "janela emergente". O código é mais simples do que parece:

Usei uma tabela principal para projetar uma "janelinha de informação" exata às do sistema operacional Windows XP, onde atribui a cada célula uma imagem, e na célula central, fiz outra tabela onde coloquei a mensagem da janelinha e o botão de Aceitar.

As outras camadas não têm absolutamente nada. Mais adiante veremos porque.

Com isto finalizo o que seria a explicação do corpo do Html. Antes de explicar os códigos que fiz em JavaScript, é importante conjecer a arquitetura da página, ou seja, a forma em que diagramei a página para conseguir o efeito desejado:



Usando CSS se pode dar um nível de profundidade às camadas. Isto é o que fiz!!! Apliquei o atributo z-index para colocar camadas em cima de outras. A camada que tiver um maior valor numérico para o atributo z-index, é a que se verá mais por cima de todas e as demais ficarão por baixo de acordo com o atributo z-index. Para o exemplo deste artigo, o corpo principal da página tem um z-index = 1, enquanto que as camadas chamadas camadaFundo1, camadaFundo2 e camadaFundo3 lhes atribui um valor de 2. À camada camadaJanela lhe atribui 3 porque é a que quero que fique por cima de todas as demais. Como disse anteriormente, todas as camadas inicialmente estão oculta (utiliza-se o atributo visibility com o valor hidden) e a idéia é aplicar JavaScript para mostrar tais camadas.

Nas 3 camadas centrais (camadaFundo1, ...fundo3), é onde radica o truque de que o corpo principal da janela fique desativado quando se mostre a janela emergente. Utilizando CSS faço com que estas camadas tenham o máximo de largura (width) que possam ter e uma altura (heigth) que coloquei a minha conveniência, de modo que cobrem todo o corpo principal. As camadas centrais se diferenciam pelas imagens que nelas se mostram. Todas têm imagens transparente com opacidades de 40%, 50% e 60% e diferentes filtros de transparência. Ao usar imagens com tendência (opacidade) ao transparente sobre o fundo da camada, a camada permite que se veja o que há por debaixo dela, neste caso, o corpo principal da página. É importante destacar, que se colocam uma imagem 100% transparente será equivalente a não colocar imagem, e o fundo se verá normal. A idéia é colocar uma imagem que não seja totalmente transparente, de modo que se veja a cor da imagem e conseguir assim que o fundo pareça inativo.

Há 2 funções JavaScript, as quais mostram e ocultam as camadas. O código da função que mostra as camadas é o seguinte:

```
function abrirJanela(janela)
{
   if (janela=="1")
  {
```

```
    document.getElementById("camadaFundo1").style.visibility="visible";
    document.getElementById("camadaFundo2").style.visibility="hidden";
    document.getElementById("camadaFundo3").style.visibility="hidden";
  }
  else if (janela=="2")
  {
    document.getElementById("camadaFundo1").style.visibility="hidden";
    document.getElementById("camadaFundo2").style.visibility="visible";
    document.getElementById("camadaFundo3").style.visibility="hidden";
  }
  else
  {
    document.getElementById("camadaFundo1").style.visibility="hidden";
    document.getElementById("camadaFundo2").style.visibility="hidden";
    document.getElementById("camadaFundo3").style.visibility="visible";
  }
  document.getElementById("camadaJanela").style.visibility="visible";
  document.formulario.bAceitar.focus();
}
```

Esta é a função que se executa cada vez que se clica em qualquer dos 3 links. Ao clicar no link localizado no primeiro parágrafo, este chama à função e lhe passa como parâmetro o número um (1), o qual indica o número do parágrafo. Ao clicar nos outros dois links, se passará como parâmetro 2 e 3 de acordo ao parágrafo. Dentro da função se obtém os estilos de cada camada e se utiliza a propriedade visibility para mostrar ou ocultar segundo seja o caso. Dentro desta função se dá o foco ao botão Aceitar.

Uma vez que se mostre a janelinha emergente, esta se pode tirar (ocultar) pressionando sobre o botão Aceitar ou sobre a X. Isto fará um chamado à função JavaScript respectiva:

```
function fecharJanela()
{
  document.getElementById("camadaFundo1").style.visibility="hidden";
   document.getElementById("camadaFundo2").style.visibility="hidden";
   document.getElementById("camadaFundo3").style.visibility="hidden";
   document.getElementById("camadaJanela").style.visibility="hidden";
   document.formulario.bAceitar.blur();
}
```

Esta função se explica por si mesma. Oculta todas as camadas e tira o foco ao botão Aceitar da janelinha. Com isto finalizo a explicação do artigo. Espero que lhe sirva e possa aplicá-lo em suas criações.

Pode-se ver o resultado final deste script em uma página à parte.

*Artigo por **José Antonio Jiménez Garelli***

Validar número de checkbox marcados com Javascript

Deixarei aqui umas linhas de minha própria criação de um script Javascript que tive que fazer para comprovar o estado de elementos checkbox ou campos de verificação de formulários.

Trata-se de utilizar os típicos campos de verificação, porém com um limitador de grupo. Pode-se utilizar em loterias de vários resultados, nos futuros testes das auto-escolas com a possibilidade de marcar várias respostas, etc.

Temos uma série de grupos de checkbox e o que queremos fazer é nos assegurar que em cada grupo, de maneira independente, não se tenha marcado mais de um número definido de campos. Por exemplo, temos x grupos de 3 campos de verificação cada um. Se o usuário marca um campo de checkbox de um dos grupos não há problema algum. Se marca 2 campos tampouco há problemas, mas se tenta marcar os três checkbox do grupo Javascript não o permite e mostra uma mensagem de erro.

Podemos ver o exemplo em funcionamento para termos uma idéia mais concreta.

Formulário HTML

Vamos ter um formulário com, neste caso, dois grupos de campos de verificação.

```
<form action="" method="post" enctype="multipart/form-data" name="formulario" id="formulario">
<table width="76">
<td width='20' valign='top'><input type='checkbox' onclick='validar(formulario.checkbox1,0)' name='checkbox1'
value='checkbox1'></td>
<td width='20' valign='top'><input type='checkbox' onclick='validar(formulario.checkbox2,0)' name='checkbox2'
value='checkbox2'></td>
<td width='20' valign='top'><input type='checkbox' onclick='validar(formulario.checkbox3,0)' name='checkbox3'
value='checkbox3'></td>
<tr>
<td width='20' valign='top'><input type='checkbox' onclick='validar(formulario.checkbox4,1)' name='checkbox4'
value='checkbox4'></td>
<td width='20' valign='top'><input type='checkbox' onclick='validar(formulario.checkbox5,1)' name='checkbox5'
value='checkbox5'></td>
<td width='20' valign='top'><input type='checkbox' onclick='validar(formulario.checkbox6,1)' name='checkbox6'
value='checkbox6'></td>
</tr>
</table>
</form>
```

Como podemos ver, o nome de cada campo é distinto. E ademais temos uma função que se executa quando se clica sobre o checkbox (evento onclick), que será a encarregada de realizar a verificação.

Função para verificar checkbox por grupos

Vejamos o código Javascript que utilizamos para realizar a comprovação de que vários checkbox não possam estar clicados ao mesmo tempo no mesmo grupo.

Primeiro, definimos um par de variáveis globais, que utilizaremos para definir os campos máximos que podem estar marcados ao mesmo tempo, e outra para levar a conta dos campos que há marcados em cada grupo.

```
//Número máximo de campos marcados por cada fila
var maxi=2;

//O contador é um array de forma que cada posição do array é uma linha do formulário
var contador=new Array(0,0);
```

Agora a função que realizará a conta de campos e informará de uma possível falha na comprovação, se se clicam mais que os que se deve.

```
function validar(check,grupo) {
  //Comprovo se o campo está marcado
  if (check.checked==true){
     //está marcado, então aumento em um o contador do grupo
     contador[grupo]++;
     //comprovo se o contador chegou ao máximo permitido
     if (contador[grupo]>maxi) {
        //se tiver chegado ao máximo, mostro mensagem de erro
        alert('Não se pode escolher mais de '+maxi+' campos ao mesmo tempo.');
        //desmarco o campo, porque não se pode permitir marcar
        check.checked=false;
        //diminuo uma unidade ao contador de grupo, porque desmarquei um campo
        contador[grupo]--;
     }
  }else {
     //se o campo não estiver marcado, diminuo um ao contador de grupo
     contador[grupo]--;
  }
}
```

A função recebe dois parâmetros. Primeiro, o campo de formulário checkbox que se clicou. Logo, o número de grupo ao que pertence esse checkbox.

Necessita-se o checkbox para conhecer seu estado e para mudá-lo caso seja necessário. O grupo o utiliza para saber a que contador deve se referir, para saber o número de campos que há clicados nesse grupo.

A função está comentada para facilitar sua leitura e compreensão.

O exemplo em funcionamento pode-se executar em uma janela à parte.

*Artigo por **Javier Bernal Lérida***

Evitar que um textarea supere um número de caracteres permitidos

Este script de Javascript é bastante utilizado em muitos web sites. Trata-se de controlar o tamanho do texto que se escreve em um textarea para evitar que os caracteres escritos superem os permitidos. O controle dos caracteres escritos se faz com Javascript, dinamicamente no lado do cliente, de modo que quando o usuário chega à longitude permitida, não se permite escrever mais conteúdo no campo textarea.

Vejamos o exemplo em funcionamento para termos uma idéia exata do objetivo deste artigo.

O exemplo é simples. Simplesmente vamos definir um número de caracteres permitidos. Com cada letra que escreva o usuário vamos comprovar se a quantidade de caracteres que há no textarea é permitida.

- Se for permitida, não fazemos nada.
- Se não for permitida, caso estivermos passando o número de caracteres que pode conter o textarea, não se deixa escrever mais no campo do formulário. Isso o conseguiremos colocando o texto que havia antes que se escrevesse esse caractere não permitido.

Adicionalmente, vamos levar a conta dos caracteres escritos em um campo de texto, para que o usuário possa visualizar os caracteres que levam escritos. Ademais, quando se chegue ao limite de caracteres permitidos se colocará em vermelho o campo de texto que conta os caracteres do textarea.

Este exercício está realizado a partir de outro exercício que publicamos anteriormente em CriarWeb.com, que seria bom ler: Contar caracteres escritos em um textarea

O exercício tem duas partes, o script Javascript e o formulário HTML. Comecemos vendo o formulário:

```
<form action="#" method="post">
<table>
<tr>
  <td>Texto:</td>
  <td><textarea cols="40" rows="5" name="texto" onKeyDown="valida_longitude()"
onKeyUp="valida_longitude()"></textarea></td>
</tr>
<tr>
  <td>Caracteres:</td>
  <td><input type="text" name=caracteres size=4></td>
</tr>
</table>
</form>
```

Não tem nenhuma complicação. Porém, há que prestar atenção aos eventos do textarea, que são onKeyDown e onKeyUp, que se desatam quando o usuário aperta ou solta teclas do teclado. Em ambos eventos se chama à função javascript valida_longitude(), que se encarregará de fazer todo o trabalho.

Vejamos agora o Javascript:

```
<script>
conteudo_textarea = ""
num_caracteres_permitidos = 10

function valida_longitude(){
  num_caracteres = document.forms[0].texto.value.length
```

```
  if (num_caracteres > num_caracteres_permitidos){
    document.forms[0].texto.value = conteudo_textarea
  }else{
    conteudo_textarea = document.forms[0].texto.value
  }

  if (num_caracteres >= num_caracteres_permitidos){
    document.forms[0].caracteres.style.color="#ff0000";
  }else{
    document.forms[0].caracteres.style.color="#000000";
  }

  conta()
}
function conta(){
  document.forms[0].caracteres.value=document.forms[0].texto.value.length
}
</script>
```

Primeiro se definem duas variáveis:

```
conteudo_textarea = ""
num_caracteres_permitidos = 10
```

A variável conteúdo_textarea armazena o conteúdo do campo textarea. À princípio está iniciada à cadeia vazia, porque o textarea supomos que está vazio.

Temos também uma variável num_caracteres_permitidos, que armazena o número de caracteres que se permite escrever no textarea. Neste caso o definimos como 10.

Agora colocamos a função valida_longitud(). O primeiro que fazemos é averiguar a quantidade de caracteres escritos, e o armazenamos na variável num_caracteres.

```
num_caracteres = document.forms[0].texto.value.length
```

Logo fazemos a parte mais importante deste script: Vemos se os caracteres escritos são menores ou iguais que os permitidos, para agir em conseqüência.

```
if (num_caracteres <= num_caracteres_permitidos){
  contenido_textarea = document.forms[0].texto.value
}else{
  document.forms[0].texto.value = conteudo_textarea
}
```

Se os caracteres escritos são menores ou iguais que os caracteres permitidos, então está tudo bem. O que fazemos é atualizar a variável que mantém o conteúdo do textarea, conteudo_textarea, introduzindo o que há no textarea atualmente, que é de um tamanho permitido.

Se o escrito no textarea for maior que o permitido, trata-se de uma situação que não se pode aprovar. Então, simplesmente escrevemos no textarea o que há na variável conteudo_textarea, que era o que havia antes e que estava validado em longitude corretamente.

Isso é tudo, é simples! Porém, agora vamos fazer uma pequena melhora para que quando o textarea chegue à longitude máxima permitida o campo de texto que leva a conta dos caracteres se coloque de cor vermelha.

```
if (num_caracteres >= num_caracteres_permitidos){
  document.forms[0].caracteres.style.color="#ff0000";
}else{
  document.forms[0].caracteres.style.color="#000000";
}
```

Como se pode ver, simplesmente se comprova de novo se o número de caracteres é maior ou igual que os permitidos. Então, sendo assim, se atualiza a propriedade style.color do campo de texto "caracteres", que mostra o número de caracteres escritos. Com style.color se pode modificar a propriedade de estilo CSS que define a cor do texto do campo. Se houvesse chegado aos caracteres permitidos, se colocaria cor vermelha, caso contrário, se colocaria cor preta.

Por último, fazemos uma chamada à função conta(), que já havíamos criado no artigo anterior:

```
function conta(){
  document.forms[0].caracteres.value=document.forms[0].texto.value.length
}
```

Esta função simplesmente atualiza o campo de texto, colocando o número de caracteres escritos no textarea.

Podemos ver de novo o exemplo em funcionamento.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Javascript não intrusivo

Uma das premissas mais importantes pensadas ao desenhar mediante o uso de padrões é a separação de camadas lógicas, ou seja, por um lado temos o projetado, que se representa mediante linguagem (x)html, por outro lado está o desenho visual, que normalmente se anexa mediantes folhas de estilo (css) até aqui tudo está muito claro.

Porém, o que ocorre com o comportamento que se quer atribuir a alguns objetos do documento, aqui é onde entra em jogo a linguagem JavaScript.

Imaginemos por exemplo que temos um link ao que queremos dar uma funcionalidade um pouco diferente ao resto, abri-lo em uma janela nova por exemplo, na maioria das vezes o primeiro que nos vem em mente é fazer algo mais ou menos similar ao seguinte:

```
<a href="popup.html" onclick="window.open('popup.html', 'width=400,height=450,resizable=yes')">Abrir popup</a>
```

Lamentavelmente, esta linha acaba com toda nossa teoria de separação de camadas, mas felizmente, existem maneiras alternativas para atribuir eventos usando JavaScript. No caso concreto deste exemplo uma maneira mais limpa de realizar o mesmo necessitava algo mais de código para poder se realizar.

Primeiro, teremos que atribuir uma identidade única ao link, e logo mediante o DOMDocument Object Model atribuiremos o evento a tal id, algo mais ou menos assim.

```
<a href="popup.html" id="mypopup">Abrir popup</a>

<script type="text/javascript">

  var x = getElementByID('mypopup');

  x.onclick = function() {

   window.open('popup.html', 'width=400,height=450,resizable=yes')

  }

</script>
```

Simples, não é? Bom, talvez não seja tão seja tão simples, porém graças a mentes inquietas como a de Ben Nolan, dispomos de algumas ferramentas que sim que fazem com que seja uma tarefa simples.

Refiro-me a behaviour, uma biblioteca JavaScript baseada na função document.getElementsBySelector escrita por Simon Willison. Este fantástico "pedaço de código" nos permite esquecermos de programar complexas funções que atribuem eventos segundo classe, id ou selector.

Continuando com o exemplo anterior, se incluirmos esta biblioteca podemos conseguir o mesmo efeito atribuindo a função diretamente ao id selecionado.

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

```
  var myrules = {

    '#mypopup' : function(element){

      element.onclick = function(){

        window.open('popup.html', 'width=400,height=450,resizable=yes')

      }

    }

};

Behaviour.register(myrules);

</script>
```

Pessoalmente, acho que é uma biblioteca de imensa utilidade, agora só falta ver como poder tirar proveito dela da melhor forma possível.

*Artigo por **Alex Sancho***

Controle de introdução de caracteres de um campo de texto com Javascript

Isto pode ser útil para campos que só admite números ou letras.

Por exemplo, vamos fazer que em um campo de texto de um formulário só se permitam colocar números decimais do tipo 9999.99:

Precisamos de uma função em JavaScript (por exemplo):

```
function fieldNumber (objeto)
{
var valorCampo;
var evento_key = window.event.keyCode;
var numPosPonto = 0;
var strParteInteira = "";
var strParteDecimal = "";
var NUM_DECIMAIS = 2;

switch (evento_key)
{
case 48:
case 49:
case 50:
case 51:
case 52:
case 53:
case 54:
case 55:
case 56:
case 57:
case 46:
break;
default:
window.event.keyCode = 0;
return false;
}

valorCampo = objeto.value;
if (evento_key == 46)
```

```
if (valorCampo.indexOf(".") != -1)
{
window.event.keyCode = 0;
return false;
}
/* Só pode teclar o número de decimais indicado em NUM_DECIMAIS */
if ((numPosPonto = valorCampo.indexOf(".")) != -1)
{
strParteInteira = valorCampo.substr(0,(numPosPonto - 1));
strParteDecimal = valorCampo.substr((numPosPonto + 1), valorCampo.length)
if (strParteDecimal.length > (NUM_DECIMAIS - 1))
{
window.event.keyCode = 0;
return false;
}
}
return true;
}
```

Teremos uma página com o formulário e a caixa de texto. Teremos que chamar à função "fieldNumber" no evento onkeypress:

```
<input type="text" name="txtImporte" onkeypress="fieldNumber(this)">
```

Se tiverem algum problema, não hesitem em entrar em contato comigo enviando um e-mail a iszori@hotmail.com

*Artigo por **Ismael Zori***

Listagem de diferentes Framework Javascript

Estou fazendo uma pesquisa sobre Frameworks Javascript e Ajax para escolher um deles e utilizá-lo em nossos projetos. À princípio vi que na web há uma infinidade de opções, algumas com muito boa pinta.

Parece que o mundo dos framework para Javascript está se popularizando muito, a julgar pelas numerosas opções. Nós até agora para fazer Javascript Cross-browser (compatível com todos os navegadores) vimos utilizando umas bibliotecas que explicamos no manual Cross Browser Javascript DHTML. Por outro lado, para trabalhar com Ajax e PHP vimos utilizando as bibliotecas Xajax, que detalharemos em breve em outro manual. Porém claro, com um Framework de Javascript talvez "matamos dois coelhos com uma cajadada só" e nos facilita muito a criação de interfaces de usuário avançadas em Javascript, necessárias para fazer projetos da web 2.0.

Para começar estou fazendo uma listagem das diferentes opções que encontrei. Logo, pesquisarei mais a fundo os framework que vi que estão tendo mais aceitação pela comunidade de desenvolvedores e os provarei. Então, escreverei artigos mais técnicos e didáticos.

Então, para não demorar mais, aqui vai a listagem de Frameworks Javascript:

Mootools: "O framework javascript compacto"
Este produto tem boa pinta. Segundo parece é simples e bem planificado. Entre as virtudes que vi mais destacadas é que é ligeiro, podendo inclusive definir que partes do framework incluir e quais não, para que se carreguem os scripts mais rápido no cliente. Muitas pessoas me falaram muito positivamente deste framework, portanto, talvez seja pelo qual comece a pesquisa detalhada.
http://mootools.net/

JQuery: "Biblioteca Javascript para escrever menos e fazer mais"
Parece ser que este é um dos frameworks com mais aceitação, por estar muito bem documentado e por ser muito simples e permitir desenvolver com um código limpo e elegante. O peso das bibliotecas é razoável e alem disso tem muitos fãs incondicionais, com o qual não cabe dúvida que será um bom projeto.
http://jquery.com/

Prototype: "O framework javascript cujo propósito é facilitar o desenvolvimento de aplicações dinâmicas"
Este framework também é muito interessante, pois há muitos usuários que o utilizam habitualmente e com êxito. Parece uma opção altamente profissional e ademais tem a garantia que o utilizam para a criação de suas webs

empresas muito conhecidas a nível mundial. Para mim oferece muitas garantias, porém há certos detratores que acusam a este framework de ser muito pesado e de tornar os web sites lentos onde se utiliza.
http://www.prototypejs.org/

YUI: "The Yahoo! User Interface Library"
É um framework que utilizam os desenvolvedores de Yahoo! para fazer seu portal, que há tempo se distribuiu para uso livre. Que provenha de Yahoo! para mim já é uma importante garantia e parece que tem desenvolvidos uma importante gama de controles e componentes. Teria que prová-lo pessoalmente para dar uma opinião, porém parece que há muitas pessoas que também o acusam de ser um pouco pesado.
http://developer.yahoo.com/yui/

Dojo: "Experiências grandes... para todos"
Parece um produto também bastante atraente e uma opção séria. Não obstante, li opiniões discordantes sobre ele. Alguns não duvidam em qualificá-lo entre os melhores frameworks Javascript e outros acusam que é pesado e pouco depurado, que produz erros facilmente.
http://www.dojotoolkit.org/

Qooxdoo: "A nova era do desenvolvimento web"
É um framework Javascript ajax multi-propósito, opensource com dois tipos de licença. Li poucas opiniões sobre este software, porém parece digno de considerar.
http://qooxdoo.org/

GWT Google Web Toolkit: "Constrói aplicações Ajax em linguagem Java"
É um conjunto framework opensource desenvolvido em Java, com o qual se criaram aplicações populares de Google, como Google Maps ou Gmail. Sem dúvida, ao se tratar de um produto de Google, não cabe dúvida que é uma opção a considerar seriamente. Tem um compilador que converte as classes Java em código Javascript e HTML compatível com todos os navegadores.
http://code.google.com/webtoolkit/

Rico: "Javascript para aplicações de Internet de conteúdo enriquecido"
Outra das opções mais conhecidas para desenvolver aplicações para a web 2.0. É open source e já se encontra na versão 2.0, com o qual se supõe que o tempo de vida lhe ajudou a ser mais depurado. Li por aí que está pouco documentado.
http://openrico.org/rico/home.page

Ext JS: "Documentação, desenho e código limpo"
Este framework Javascript parece ser outra das opções sérias. Distribui-se sob licença Open Source (grátis) e licença comercial (paga, mas com suporte e alguma funcionalidade adicional). Empresas bastante importantes o utilizam, como Adobe. Chama a atenção por ter suporte para Adobe Air.
http://extjs.com/

Ainda faltam mais opções, porém vou deixá-las listadas sem muitos comentários, porque tampouco investiguei muito e não as vi em nenhum lugar comentadas como opções de primeira linha.

- The Foo Framework (um framework baseado em Prototype): http://foo.riiv.net/
- script.aculo.us (também baseado em Prototype): http://script.aculo.us/
- AJS (Framework Javascript ultraligeiro): http://orangoo.com/labs/AJS/
- ZK (Ajax web framework): http://www.zkoss.org/

Isto é tudo para o momento. Agora resta começar a trabalhar para aprender a manejar os Frameworks e tirar conclusões mais sérias. Espero que em breve possamos publicar mais sobre o tema.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Script para detecção de suporte a Ajax, Cookies e ActiveX

O site de Xajax Project publicou uns scripts interessantes para poder detectar se um navegador é compatível com a tecnologia Ajax, para estar seguros que a web que estamos desenvolvendo poderá mostrar corretamente em qualquer cliente web que tenha o usuário. Ademais, estes scripts servem para mostrar mensagens de erro se o navegador não tem suporte a Ajax, de modo que o usuário seja consciente que não vai poder ver essa web convenientemente.

Estes scripts detectam as capacidades do navegador e se podem executar para mostrar mensagens de alerta se não estão disponíveis certas funcionalidades, já seja porque o navegador do usuário não as suporte ou porque estejam desabilitadas.

O script contém três funções:

browserSupportsCookies()
Detecta se o navegador suporta cookies e devolve true no caso de que estejam suportadas e false se não for assim.

browserSupportsAjax()
Comprova se o navegador tem compatibilidade com a tecnologia Ajax, devolve true se for assim e false se não suporta Ajax por qualquer questão.

ActiveXEnabledOrUnnecessary()
Esta função detecta se o navegador suporta ActiveX ou então se ActiveX é desnecessário para a execução de Ajax. No navegador Internet Explorer 6 Ajax se executa através de ActiveX, por isso necessita dispor ActiveX para que tudo funcione. Portanto, esta função devolverá false só se o navegador for Internet Explorer 6 e tiver desabilitado ActiveX.

Não escreverei as funções no texto deste artigo, simplesmente vou colocar um link ao lugar onde se mostram as funções na página de Xajax Project:

http://xajaxproject.org/wiki/Xajax_%28any%29:_Tips_and_Tricks:_Detecting_Support

Porém, também deixarei um link a uma página em CriarWeb.com onde implementamos estes scripts, para que se possa ver em funcionamento em seus navegadores. Ainda assim, pode-se ver o código fonte da página para ver a implementação dos scripts que fizemos em CriarWeb.com e obter o código das funções no caso que mudem a URL na página de Xajax.

http://www.criarweb.com/artigos/exemplos/comprobar-compatibilidad-ajax.html

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Leitor RSS com Javascript

Estive pesquisando em diferentes sites a maneira de criar sistema em Javascript que leia RSS de outras webs, para publicar os títulos em uma página. Finalmente, encontrei um script leitor de RSS que irei comentar neste artigo.

O sistema permite ler uma folha XML que contém um feed RSS e escreve as entradas do RSS na página. No conteúdo da página não figura o RSS, e sim que está em um arquivo externo e com o script se escreve o texto das distintas entradas, com seus links e outras informações.

O script se encontra publicado na página http://www.cstruter.com/downloads.php

Eu o baixei e coloquei no servidor de CriarWeb.com, no caso de que o tirem da web onde obtive (como já nos passou com outros scripts que comentamos neste site). Pode-se baixá-lo com este link. Não obstante, recomendo entrar na página onde foi obtido, no caso de que publiquem versões novas.

Condicionantes para o uso do leitor RSS com Javascript

Antes de continuar explicando o funcionamento há que dizer que existe uma restrição de uso deste script, que é importante porque em Firefox não funcionará. Trata-se de que Firefox, como medida de segurança, não permite ler o conteúdo de outras webs. Como o RSS com os títulos se pegam de outras, pois em Firefox teremos problemas, porque não irá permitir sua leitura e a apresentação dos títulos na página. Este problema o Internet Explorer não tem, porém mesmo assim teremos que buscar outras soluções.

A solução mais simples seria a de publicar o RSS em nosso servidor. Ou seja, baixá-lo da web desejada e subi-lo por FTP ao nosso site. Claro que isto nos obrigaria a realizar uma operação manual cada vez que quiséssemos que os títulos se atualizassem e isso pode significar perder uma das vantagens de apresentar titulares RSS de outras webs, que é dispor sempre de conteúdo atualizado. Ademais, seria uma pouco cansativa a tarefa de baixar o feed RSS e subir todos os dias o arquivo XML a nossa web.

A melhor solução seria criar um script em programação do lado do servidor, como por exemplo PHP, ASP ou .NET que

realize a tarefa de baixar o RSS com os títulos e o copie em nosso servidor. Este script poderia ser executado a cada certo tempo ou cada vez que um usuário acessasse à página onde se lê o RSS remoto para apresentar os títulos. A desvantagem desta opção é que necessitamos que nosso servidor suporte programação de scripts em PHP, ASP ou similares. À parte que se fizermos programação do lado do servidor para extrair o feed RSS poderíamos diretamente tratá-lo para apresentar os dados na página, sem a necessidade deste script Javascript. Não obstante, cabe assinalar que em CriarWeb.com, nas seções monotemáticas de PHP ou ASP temos materiais para aprender a ler um arquivo remoto, que esteja em outro servidor.

Uso do leitor RSS Javascript

O script é extremadamente simples de utilizar, já que toda a parte complicada se faz por si mesmo. Simplesmente temos que especificar em uma linha de código o arquivo RSS do que tem que extrair os títulos.

Do arquivo de download, simplesmente temos que mudar a seguinte linha:

ReadRSS('cnn_tech_rss.xml','rssBodyTemplate','rssTitleTemplate');

A função ReadRSS(), que é o leitor RSS, no primeiro parâmetro tem o nome do arquivo RSS que deve ler. Nós podemos mudá-lo pelo nome do arquivo que pretendemos mostrar seus títulos.

Poderíamos mudar este arquivo pela URL completa do feed RSS do servidor onde o têm publicado. Por exemplo, para ler o RSS com as novidades de FAQ que publicamos em CriarWeb.com a função se chamaria assim:

ReadRSS('http://www.criarweb.com/rss/faq_rss.php','rssBodyTemplate','rssTitleTemplate');

Em Internet Explorer não há nenhum problema com este uso da função, simplesmente veremos que os títulos demoram um pouco mais em se gerar, devido a que tem que conectar com a página remota para baixar o RSS. Porém, poderemos comprovar que a função, quando se executa em Firefox, mostra uma mensagem advertindo o problema e sugerindo que o nosso próprio servidor copie o arquivo remoto para poder funcionar.

Baixem o script o qual nos referimos e realizem suas próprias provas.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Funções para validação alfanumérica de strings em Javascript

Criei uma série de funções de string para realizar umas comprovações sobre cadeias de caracteres, que vou utilizar mais adiante em um script mais complexo. Começo explicando estas funções soltas, que talvez tenham utilidade aos leitores. Também com a intenção de apresentar pouco a pouco a complexidade de meu objetivo final. As funções de validação de strings servem para saber se as cadeias têm ou não números, letras, letras maiúsculas e minúsculas. São uma série de funções bem simples e parecidas entre si, que fazem uso dos métodos da classe string de Javascript.

Estas funções simplesmente fazem o percorrido pelo string em busca de caracteres de um determinado tipo. Fazem o percorrido completo por todo o string até que encontram um caractere do tipo buscado, de modo que se possa saber com certeza se há ou não caracteres desse tipo. Veremos a seguir as diferentes funções:

Saber se o string contém caracteres numéricos

Esta função recebe um string e devolve 1 se se encontram caracteres numéricos e 0 se não se encontram.

```
var numeros="0123456789";

function tem_numeros(texto){
  for(i=0; i<texto.length; i++){
    if (numeros.indexOf(texto.charAt(i),0)!=-1){
      return 1;
    }
  }
  return 0;
}
```

Criamos primeiro uma variável global com os números possíveis, do 0 ao 9, que queremos buscar. Utilizaremos essa

variável dentro da função.

A função faz um percorrido a todos os caracteres do string. Para cada um se comprova se está ou não dentro da variável números, criada anteriormente.

Se estiver, se devolve 1 (como se executa o return, se sai da função devolvendo o valor 1).

Se não estiver, então se fará o percorrido do string, caractere a caractere, até o final. Então, se termina o loop e se devolve 0.

Esta função pode ser provada com estas sentenças:

```
alert(tem_numeros("ASAS1"));
alert(tem_numeros("2asasasas"));
alert(tem_numeros("asas2sG"));
alert(tem_numeros("a...."));
alert(tem_numeros("A22323G2.12"));
```

Podemos ver o exemplo em funcionamento em uma página a parte.

Saber se um string contém letras

Vejamos uma função muito similar, para saber se um string contém um caractere que seja letra, valendo tanto as maiúsculas como minúsculas.

```
var letras="abcdefghyjklmnopqrstuvwxyz";

function tem_letras(texto){
  texto = texto.toLowerCase();
  for(i=0; i<texto.length; i++){
    if (letras.indexOf(texto.charAt(i),0)!=-1){
      return 1;
    }
  }
  return 0;
}
```

O algoritmo é praticamente o mesmo que a função anterior. Primeiro, criamos um string com todas as letras do alfabeto. Logo, fazemos um percorrido buscando em cada um dos caracteres do string recebido por parâmetro uma das letras do alfabeto.

A única particularidade é a chamada ao método toLowerCase(), para converter o texto que se recebe por parâmetro a minúsculas e que a busca de letras não tenha em conta se são minúsculas ou maiúsculas (já que só buscamos se o string contém letras).

Podemos provar o string com estas linhas de código:

```
alert(tem_letras("1"));
alert(tem_letras("22323232s"));
alert(tem_letras("22323232sf"));
alert(tem_letras("a2232323"));
alert(tem_letras("A22323G2.12"));
```

Podemos ver o exemplo em funcionamento aqui.

Saber se um string tem letras minúsculas

Aqui vemos a função para comprovar se um string tem caracteres em minúsculas. É muito parecida as que vimos anteriormente.

```
var letras="abcdefghyjklmnopqrstuvwxyz";

function tem_minusculas(texto){
  for(i=0; i<texto.length; i++){
    if (letras.indexOf(texto.charAt(i),0)!=-1){
```

```
      return 1;
    }
  }
  return 0;
}
```

Com sua bateria de provas:

```
alert(tem_minusculas("1"));
alert(tem_minusculas("22323232s"));
alert(tem_minusculas("22323232sf"));
alert(tem_minusculas("a2232323"));
alert(tem_minusculas("A22323G2.12"));
```

Pode-se executar aqui.

Comprovar se um string tem letras maiúsculas

Agora veremos a função para ver se um string tem letras maiúsculas. É igual que a anterior, simplesmente temos uma cadeia com todas as letras em maiúscula para buscar no texto recebido por parâmetro.

```
var letras_mayusculas="ABCDEFGHYJKLMNOPQRSTUVWXYZ";

function tem_maiusculas(texto){
  for(i=0; i<texto.length; i++){
    if (letras_maiusculas.indexOf(texto.charAt(i),0)!=-1){
      return 1;
    }
  }
  return 0;
}
```

Estas poderiam ser as provas para comprovar o funcionamento.

```
alert(tem_maiusculas("1"));
alert(tem_maiusculas("22323232s"));
alert(tem_maiusculas("22323232sG"));
alert(tem_maiusculas("a2232323"));
alert(tem_maiusculas("A22323G2.12"));
```

Pode ver em funcionamento o script no seguinte link.

Utilizaremos estas funções no seguinte artigo, no qual vou explicar a criação de um sistema para mostrar a segurança de uma chave, para dizer o quão segura é em função do número de caracteres, se tem números e letras, se tem maiúsculas e minúsculas, etc.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Script para informar da segurança de uma senha, com Javascript

Veremos um simples script em Javascript para comprovar o grau de segurança de uma senha escrita pelo usuário. Como é um script javascript do lado do cliente, permitirá mostrar o nível de segurança da senha ao mesmo tempo que o usuário a escreve em um campo de formulário.

Poderemos utilizar este script livremente em nossas páginas, de modo que ofereçamos aos visitantes uma informação sobre o segura ou insegura que é a senha que estão escolhendo, o que lhes motivará a escrever senhas mais seguras que as que habitualmente se escrevem.

Podemos ver um exemplo do objetivo buscado antes de continuar.

Em um artigo anterior do Workshop de Javascript estivemos mostrando a maneira de fazer várias funções para comprovar um string e saber se tem letras, números, maiúsculas e minúsculas. Utilizaremos estas funções agora para

este script de informação de segurança da senha.

Para valorizar o grau de segurança de uma senha vamos ter em conta estas pontuações sobre diferentes conceitos:

Tem letras e números: +30%
Tem maiúsculas e minúsculas: +30%
Tem entre 4 e 5 caracteres: +10%
Tem entre 6 e 8 caracteres: +30%
Tem mais de 8 caracteres: +40%

Poderíamos ter escolhido qualquer outra pontuação para a segurança da senha, porém esta valerá. Também poderíamos ter criado outros critérios para decidir o grau de segurança. De qualquer forma para que fique claro este cálculo, ponho um par de exemplos:

A) Uma senha com números e letras, com 7 caracteres teria: 30% por letras e números + 30% por ter entre 6 e 8 caracteres = 60% de segurança.
B) Outra senha com letras maiúsculas e minúsculas, sem números, e com 8 caracteres: 30% por maiúsculas e minúsculas + 40% por mais de 8 caracteres = 70% de segurança.

Para controlar a segurança, apoiando-nos nas funções de validação alfanumérica de strings vistas anteriormente, faremos uma função como esta:

```
function seguranca_senha(senha){
  var seguranca = 0;
  if (senha.length!=0){
    if (tem_numeros(senha) && tem_letras(senha)){
      seguranca += 30;
    }
    if (tem_minusculas(senha) && tem_maiusculas(senha)){
      seguranca += 30;
    }
    if (senha.length >= 4 && senha.length <= 5){
      seguranca += 10;
    }else{
      if (senha.length >= 6 && senha.length <= 8){
        seguranca += 30;
      }else{
        if (senha.length > 8){
          seguranca += 40;
        }
      }
    }
  }
  return seguranca
}
```

Vamos comprovando se o string tem diversas coisas, como letras, números, maiúsculas, minúsculas, assim como sua longitude, para ir atribuindo um maior valor à segurança.

Exemplo de uso em um formulário que pede uma senha

Agora, vejamos um simples exemplo de uso da função em um formulário, que mostra a segurança de uma senha escrita pelo usuário:

O formulário poderia ser como este:

```
<form>
Senha: <input type="password" size=15 name="senha" onkeyup="mostra_seguranca_senha(this.value, this.form)">
<i>seguranca:</i> <input name="seguranca" type="text" style="border: 0px; background-color:ffffff; text-decoration:italic;" onfocus="blur()">
</form>
```

Como vemos, temos 1 campo INPUT de tipo PASSWORD onde escreveremos a senha. A este campo se introduziu um evento ONKEYUP que se executa quando o usuário clica uma tecla, porém no momento de soltá-la. Essa função será a encarregada de fazer com que se visualize a segurança da senha.

Ademais, colocamos outro campo de texto, para colocar o valor de segurança da senha. Este campo o forçamos a que não se possa escrever nele com o evento onfocus="blur()">. Só se poderá modificar mediante Javascript.

Vejamos a função mostra_seguranca_senha(), que é a que se encarrega de receber tanto a senha escrita como o formulário onde se encontra, para atualizar o valor de segurança.

```
function mostra_seguranca_senha(senha,formulario){
   seguridad=seguranca_senha(senha);
   formulario.seguranca.value=seguranca + "%";
}
```

Como vemos, se faz uso da função que devolve a segurança de um string que se utilizará como senha. Logo, se coloca esse valor no campo de texto adicional que há no formulário.

Podemos ver o exemplo em uma página a parte.

*Artigo por **Miguel Angel Alvarez - Tradução de JML***

Editor de texto WYSIWYG Javascript: TinyMCE

TinyMCE é um editor HTML que é capaz de converter os textarea de um formulário em campos WYSIWYG para poder incluir etiquetas HTML dentro dos campos de texto.

Características

- É fácil de integrar nas páginas web, já que só tem duas linhas de código.
- Pode-se personalizar através de temas e plugins.
- Também se podem instalar pacotes de idiomas.
- É compatível com a maioria dos navegadores como Firefox, Internet Explorer, Opera e Safari, embora este último está em fase experimental.
- Com o compressor GZip para PHP/.NET/JSP/Coldfusion, faz com que TinyMCE seja um 75% menor e muito mais rápido de carregar.
- Pode-se utilizar AJAX para salvar e carregar o conteúdo.

Integração de TinyMCE

Para poder utilizar TinyMCE nas páginas web, o navegador tem que ser compatível e ter Javascript habilitado.

Primeiro há que baixar TinyMCE desde a seguinte página de downloads: http://tinymce.moxiecode.com/download.php. Depois há que descompactá-lo e salvá-lo no servidor da página web para poder utilizá-lo nos textarea dos formulários.

Na página que for utilizar, primeiro há que incluir a biblioteca tiny_mce.js incluindo o arquivo externo de código Javascript.

```
<script language="javascript" type="text/javascript" src="/tinymce/jscripts/tiny_mce/tiny_mce.js"></script>
```

A seguir há que iniciar TinyMCE para converter os textarea em campos de texto WYSIWYG editáveis.

```
<script language="javascript" type="text/javascript">
tinyMCE.init({
     mode : "textareas",
     theme : "simples"
  });

</script>
```

Exemplo de integração de TinyMCE

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

```
<html>
<head>
  <title>Exemplo TinyMCE</title>
  <script language="javascript" type="text/javascript" src="/tinymce/jscripts/tiny_mce/tiny_mce.js"> </script>
  <script language="javascript" type="text/javascript">
    tinyMCE.init({
      mode : "textareas",
      theme : "advanced"
    });
  </script>
</head>

<body>
  <form method="post" name="tinymce">
    <textarea name="texto" cols="50" rows="15"></textarea>
  </form>
</body>
</html>
```

Neste exemplo, primeiro incluímos a biblioteca tiny_mce.js dentro das etiquetas <head> . Dentro destas etiquetas também iniciamos TinyMCE para que no textarea do formulário se converta em um campo de texto WYSIWYG.

Pode-se ver o exemplo em funcionamento no seguinte link.

*Artigo por **Gema Maria Molina Prados***