Programação em Javascript

Autores do manual

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

Miguel Angel Alvarez -
Tradução de JML
(36 capítulos)

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Introdução à Javascript

Javascript é uma linguagem de programação utilizada para criar pequenos programinhas encarregados de realizar ações dentro do âmbito de uma página web. Com Javascript podemos criar efeitos especiais nas páginas e definir interatividades com o usuário. O navegador do cliente é o encarregado de interpretar as instruções Javascript e executá-las para realizar estes efeitos e interatividades, de modo que o maior recurso, e talvez o único, com que conta esta linguagem é o próprio navegador.

Javascript é o seguinte passo, depois de HTML, que pode dar um programador da web que decide melhorar suas páginas e a potência de seus projetos. É uma linguagem de programação bastante simples e pensada para fazer as coisas com rapidez, às vezes com leveza. Inclusive as pessoas que não tenham uma experiência prévia na programação poderão aprender esta linguagem com facilidade e utilizá-la em toda sua potência com somente um pouco de prática.

Entre as ações típicas que se podem realizar em Javascript temos duas vertentes. Por um lado, os efeitos especiais sobre páginas web, para criar conteúdos dinâmicos e elementos da página que tenham movimento, mudam de cor ou qualquer outro dinamismo. Por outro, javascript nos permite executar instruções como resposta às ações do usuário, com o qual podemos criar páginas interativas com programas como calculadoras, agendas, ou tabelas de cálculo.

Javascript é uma linguagem com muitas possibilidades, permite a programação de pequenos scripts, e também de programas maiores, orientados a objetos, com funções, estruturas de dados complexas, etc. Toda esta potência de Javascript se coloca à disposição do programador, que se converte no verdadeiro dono e controlador de cada coisa que ocorre na página.

Neste livro vamos tratar de aproximarmos desta linguagem mais profundidade e conhecer todos seus segredos e métodos de trabalho. Ao final do livro seremos capazes de controlar a página web e discernir o melhor método para atacar os problemas ou objetivos que tivermos planejado.

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

Algo de história

Foi criado na Internet uma infinidade de serviços para realizar muitos tipos de comunicações, como correio, chats, buscas de informação, etc. Mas nenhum desses serviços se desenvolveram tanto como o Web. Se estamos lendo estas linhas não vamos precisar de nenhuma explicação do que é o web, mas sim que podemos falar um pouco sobre como foi se desenvolvendo com o passar dos anos.

O web é um sistema Hipertexto, uma quantidade descomunal de textos que contém links que relacionam cada uma das unidades básicas onde podemos encontrar informação, as páginas web. À princípio, para desenhar este sistema de páginas com links se pensou em uma linguagem que permitisse apresentar cada uma dessas informações juntos cm uns pequenos estilos, esta linguagem foi o HTML, que logo se veria que não cumpriu todos os objetivos para os que foi desenhado, mas isso é outro assunto.

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

O caso é que HTML não é suficiente para realizar todas as ações que se podem chegar a necessitar em uma página web. Isto é devido a que conforme foi crescendo o web e seus distintos usos foram se complicando as páginas e as ações que queriam se realizar através delas. O HTLM havia se tornado curto para definir todas estas novas funcionalidades, já que somente serve para apresentar o texto em uma página, definir seu estilo e pouco mais.

O primeiro ajudante para cobrir as necessidades que estavam surgindo foi Java, através da tecnologia dos Applets, que são pequenos programas que se incrustam nas páginas web e que podem realizar as ações associadas aos programas de propósito geral. A programação de Applets foi um grande avance e Netscape, até então, o navegador mais popular, havia rompido a primeira barreira do HTML ao fazer possível a programação dentro das páginas web. Não cabe dúvida que o aparecimento dos Applets supôs um grande avance na história do web, mas foi uma tecnologia definitiva e muitas outras seguiram implementando o caminho que começou com eles.

Chega Javascript:

Netscape, depois de fazer seus navegadores compatíveis com os applets, começou a desenvolver uma linhguagem de programação ao que chamou LiveScript que permitisse criar pequenos programas nas páginas e que fosse muito mais simples de utilizar que Java. De modo que o primeiro Javascript se chamou LiveScript, mas não durou muito esse nome, pois antes de lançar a primeira versão do produto se forjou uma aliança com Sun Microsystems, criador de Java, para desenvolver em conjunto essa nova linguagem.

A aliança fez com que Javascript se desenhara como um irmão pequeno de Java, somente útil dentro das páginas web e muito mais fácil de utilizar, de modo que qualquer pessoa, sem conhecimentos de programação pudesse aprofundar-se na linguagem e utilizá-la. Ademais, para programar Javascript não é necessário um kit de desenvolvimento, nem compilar os scripts, nem realizá-los em ficheiros externos ao código HTML, como ocorreria com os applets.

Netscape 2.0 foi o primeiro navegador que entendia Javascript e seu rastro foi seguido pelos navegadores da companhia Microsoft a partir da versão 3.0.

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

Diferenças entre Java e Javascript

Queremos que fique claro que **Javascript não tem nada a ver com Java**, salvo em suas origens. Atualmente são produtos totalmente distintos e não guardam entre si mais relação que a sintaxe idêntica e pouco mais. Algumas diferenças entre estas duas linguagens são as seguintes:

- **Compilador**. Para programar em Java necessitamos um Kit de desenvolvimento e um compilador. Entretanto, Javascript não é uma linguagem que necessite que seus programas se compilem, senão que estes se interpretem por parte do navegador quando este lê a página.
- **Orientado a objetos**. Java é uma linguagem de programação orientada a objetos. (Mais tarde veremos que quer dizer orientado a objetos, para quem ainda não sabe) Javascript não é orientado a objetos, isto quer dizer que poderemos programar sem

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

necessidade de criar classes, tal como se realiza nas linguagens de programação estruturada como C ou Pascal.

- **Propósito**. Java é muito mais potente que Javascript, isto é devido a que Java é uma linguagem de propósito geral, com o que se podem fazer aplicações do mais variado, entretanto, com Javascript somente podemos escrever programas para que se executem em páginas web.
- **Estruturas fortes**. Java é uma linguagem de programação fortemente tipada, isto quer dizer que ao declarar uma variável teremos que indicar seu tipo e não poderá mudar de um tipo a outro automaticamente. Por sua parte, Javascript não tem esta característica, e podemos colocar em uma variável a informação que desejarmos, independentemente do tipo desta. Ademais, poderemos mudar o tipo de informação de uma variável quando quisermos.
- **Outras características**. Como vemos Java é muito mais complexo, mas também, mais potente, robusto e seguro. Tem mais funcionalidades que Javascript e as diferenças que os separam são o suficientemente importantes como para distinguí-los facilmente.

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

Antes de começar

Previamente para começar a utilizar Javascript podemos ter uma idéia mais concreta das possíveis aplicações desta linguagem assim como as ferramentas que necessitamos para colocarmos mãos a obra.

Usos de Javascript

Vejamos brevemente alguns usos desta linguagem que podemos encontrar na web para termos uma idéia das possibilidades que tem.

Para começar, podemos ver páginas cheia de efeitos super interessantes sobre Javascript, que chegam a assemelhar-se à tecnologia Flash.

Por outro lado, podemos encontrar dentro de Internet muitas aplicações de Javascript muito mais sérias, que fazem com que uma página web se converta em um verdadeiro programa interativo de gestão de qualquer recurso. Também podemos ver exemplos dentro de qualquer página um pouco complexa, quando passamos por sites que tenham uma calculadora ou um conversor de divisas, veremos que em muitos casos foi realizado com Javascript.

Na verdade é muito mais habitual encontrar Javascript para realizar efeitos simples sobre páginas web, ou no tão simples, como podem ser rollovers (que muda uma imagem ao passar o mouse por cima), navegadores desdobráveis, abertura de janelas secundárias, etc. Podemos nos atrever a dizer que esta linguagem é realmente útil para estes casos, pois estes típicos efeitos têm a complexidade justa para ser implementados em questão de minutos sem possibilidade de erros.

O que é necessário

Para programar em Javascript necessitamos basicamente o mesmo que para programar

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

páginas web com HTML. Um editor de textos e um navegador compatível com Javascript. Um usuário de Windows possui de saída todo o necessário para poder programar em Javascript, visto que dispõe dentro de sua instalação típica de sistema operativo, de um editor de textos, o Bloco de notas, e de um navegador: Internet Explorer.

Usuários de outros sistemas podem encontrar em Internet facilmente as ferramentas necessárias para começar em páginas de download de software como Tucows.

Uma recomendação em relação ao editor de textos. Trata-se de que, apesar do Bloco de Notas ser suficiente para começar, talvez seja muito útil contar com outros programas que nos oferecem melhores prestações na hora de escrever as linhas de código. Estes editores avançados têm algumas vantagens como que colorem os códigos de nossos scripts, nos permitem trabalhar com vários documentos simultaneamente, têm ajudas, etc. Entre outros queremos destacar o Home Site ou o UltraEdit.

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

Versões de navegadores e de Javascript

Também é apropriado introduzir as distintas versões de Javascript que existem e que evolucionaram em conjunto com as versões de navegadores. A linguagem foi avançando durante seus anos de vida e incrementando suas capacidades. À princípio podia realizar muitas coisas na página web, mas tinha poucas instruções para criar efeitos especiais. Com o tempo também o HTML foi avançando e foram criadas novas características como as camadas, que permitem tratar e planificar os documentos de maneira distinta. Javascript também avançou e para manejar todas estas novas características foram criados novas instruções e recursos. Para resumir vamos comentar as distintas versões de Javascript:

- **Javascript 1**: nasceu com o Netscape 2.0 e suportava grande quantidade de instruções e funções, quase todas as que existem agora já se introduziram no primeiro padrão.
- **Javascript 1.1**: É a versão de Javascript que foi desenhado com a chegada dos navegadores 3.0. Implementava pouco mais que sua versão anterior, como por exemplo, o tratamento de imagens dinamicamente e a criação de arrays.
- **Javascript 1.2**: A versão dos navegadores 4.0. Esta tem como desvantagem que é um pouco distinta em plataformas Microsoft e Netscape, já que ambos navegadores cresceram de distinto modo e estavam em plena luta no mercado.
- **Javascript 1.3**: Versão que implementam os navegadores 5.0. Nesta versão foram limitadas algumas diferenças e asperezas entre os dois navegadores.
- **Javascript 1.5**: Versão atual, no momento de escrever estas linhas, que implementa Netscape 6.
- Por este lado, **Microsoft** também foi evoluindo até apresentar sua **versão 5.5 de JScript** (assim chamam ao javascript utilizado pelos navegadores de Microsoft).

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

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

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

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

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***

A linguagem Javascript

Nesta parte do livro vamos conhecer a maneira de trabalhar com Javascript, como incluir scripts e ser compatível com todos os navegadores. Muitas idéias do funcionamento de Javascript já foram descritas nos capítulos anteriores, mas com o objetivo de não deixarmos nada solto no ar, vamos tratar de reforçar a partir daqui todos os dados mais importantes desta linguagem.

Javascript se escreve no documento HTML

O mais importante e básico que podemos destacar neste momento é que a programação de Javascript se realiza dentro do próprio documento HTML. Isto quer dizer que na página se misturam as duas linguagens, e para que estas duas linguagens possam ser misturadas sem problemas temos que incluir alguns delimitadores que separam as etiquetas HTML das instruções Javascript. Estes delimitadores são as etiquetas <SCRIPT> e </SCRIPT>. Todo o código Javascript que colocarmos na página há de ser introduzido entre estas duas etiquetas.

Em uma mesma página podemos introduzir vários scripts, cada um que poderia se introduzir dentro das etiquetas <SCRIPT> distintas. A colocação destes scripts é indiferente, à princípio dá no mesmo aonde coloca-los, mas em determinados casos esta colocação sim que será muito importante. Em cada caso, e chegado o momento isto será informado convenientemente.

Também se pode escrever Javascript dentro de determinados atributos da página, como o atributo *onclick*. Estes atributos estão relacionados com as ações do usuário e são chamados de manejadores de eventos.

Vamos ver no próximo capítulo mais detalhadamente estas duas maneiras de escrever scripts, que tem como diferença principal o momento em que se executam as sentenças.

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

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Maneiras de executar script

Existem **duas maneiras de executar scripts** na página. A primeira destas maneiras se trata de execução direta de scripts, a segunda é uma execução como resposta à ação de um usuário. Veremos agora cada uma delas.

Execução direta

É o método mais básico de executar scripts. Neste caso se incluem as instruções dentro da etiqueta <SCRIPT>, tal como comentamos anteriormente. Quando o navegador lê a página e encontra um script vai interpretando as linhas de código e vai executando uma depois da outra. Chamamos a esta maneira execução direta, pois quando se lê a página se executam diretamente os scripts.

Este método será o que utilizemos preferentemente na maioria dos exemplos deste livro.

Resposta a um evento

É a outra maneira de executar scripts, mas antes de vê-la devemos falar sobre os eventos. Os eventos são ações que realiza o usuário. Os programas como Javascript estão preparados para apanhar determinadas ações realizadas, neste caso sobre a página, e realizar ações como resposta. Deste modo se podem realizar programas interativos, já que controlamos os movimentos do usuário e respondemos a eles. Existem muitos tipos de eventos distintos, por exemplo, o click do mouse, a seleção de texto da página, entre outros.

As ações que queremos realizar como resposta a um evento devem ser indicadas dentro do mesmo código HTML, mas neste caso se indicam em atributos HTML que se colocam dentro da etiqueta que queremos que responda às ações do usuário. No capítulo em que vimos algum exemplo rápido já comprovamos que se quiséssemos que um botão realizasse ações quando se clicasse sobre ele, devíamos indicá-las dentro do atributo onclick do botão.

Comprovamos então, que se pode introduzir código Javascript dentro de determinados atributos das etiquetas HTML. Veremos mais adiante este tipo de execução em profundidade e os tipos de eventos que existem.

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

Ocultar scripts em navegadores antigos

Já vimos que Javascript se implementou a partir de Netscape 2.0 e Internet Explorer 3.0, inclusive existem navegadores que funcionam em sistemas onde somente se pode visualizar texto e portanto determinadas tecnologias, como esta linguagem, estão fora do seu alcance. Sendo assim, nem todos os navegadores do web compreendem Javascript. Nos casos em não se interpretam os scripts, os navegadores assumem que o código destes é o texto da própria página web e como consequência, apresentam os scripts na página web como se tratasse de um texto normal. Para evitar que o texto dos scripts se escreva na página quando os navegadores não os entendem, temos que ocultá-los com comentários HTML (<!-comentario HTML -->). Vejamos com um exemplo como se deve ocultar os scripts.

<SCRIPT>

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

```
<!--
Código Javascript
//-->
</SCRIPT>
```

Vemos que o inicio do comentário HTML é idêntico a como o conhecemos no HTML, porém o fechamento do comentário apresenta uma particularidade, que começa por duas barras inclinadas. Isto é devido a que o final do comentário contém vários caracteres que Javascript reconhece como operadores e ao tratar de analisá-los lança uma mensagem de erro de sintaxe. Para que Javascript não lance uma mensagem de erro se coloca antes do comentário HTML essa barra dupla, que não é mais que um comentário Javascript, que conheceremos mais adiante quando falarmos de sintaxe.

O início do comentário HTML não é necessário comentá-lo com a barra dupla, dado que Javascript entende bem que simplesmente se pretende ocultar o código. Um esclarecimento a este ponto: se colocássemos as duas nesta linha, se veriam em navegadores antigos por estar fora dos comentários HTML. Os navegadores antigos não entendem as etiquetas <SCRIPT> , portanto não as interpretam, tal como fazem com qualquer etiqueta que desconhecem.

<NOSCRIPT>

Existe a possibilidade de indicar um texto alternativo para os navegadores que não entendem Javascript, para lhes informar de que nesse lugar deveria executar um script e que a página não está funcionando com 100% de suas capacidades. Também podemos sugerir aos visitantes que atualizem seu navegador a uma versão compatível com a linguagem. Para isso utilizamos a etiqueta <NOSCRIPT> e entre esta etiqueta e seu correspondente de fechamento podemos colocar o texto alternativo ao script.

```
<SCRIPT>
código javascript
</SCRIPT>
<NOSCRIPT>
Este navegador não compreende os scripts que se estão executando, você deve atualizar sua
versão de navegador a uma mais recente.
<br><br>
<a href=http://netscape.com>Netscape</a>.<br>
<a href=http://microsoft.com>Microsoft</a>.
</NOSCRIPT>
```

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

Mais sobre colocar scripts

Duas notas adicionais sobre como colocar scripts em páginas web.

Linguagem que estamos utilizando

A etiqueta <SCRIPT> tem um atributo que serve para indicar a linguagem que estamos utilizando, assim como a versão deste. Por exemplo, podemos indicar que estamos programando em Javascript 1.2 ou Visual Basic Script, que é outra linguagem para programar

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

scripts no navegador cliente que somente é compatível com Internet Explorer.

O atributo em questão é language e o mais habitual é indicar simplesmente a linguagem com a qual foram programados os scripts. A linguagem por padrão é Javascript, portanto se não utilizamos este atributo, o navegador entenderá que a linguagem com que se está programando é Javascript.

<SCRIPT LANGUAGE=javascript>

Arquivos externos de Javascript

Outra maneira de incluir scripts em páginas web, implementada a partir de Javascript 1.1, é incluir arquivos externos onde se podem colocar muitas funções que se utilizem na página. Os arquivos costumam ter extensão .js e se incluem desta maneira.

```
<SCRIPT language=javascript src="arquivo_externo.js">
//estou incluiendo o arquivo "arquivo_externo.js"
</SCRIPT>
```

Dentro das etiquetas <SCRIPT> pode ser escrito qualquer texto e será ignorado por o navegador, entretanto, os navegadores que não entendem o atributo SRC terão este texto por instruções, pelo qual é aconselhável botar um comentário Javascript antes de cada linha com o objetivo de que não respondam com um erro.

O arquivo que incluímos (neste caso arquivo_externo.js) deve conter somente sentenças Javascript. Não devemos incluir código HTML de nenhum tipo, nem sequer as etiquetas </SCRIPT> e </SCRIPT>.

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

Sintaxe Javascript

A linguagem **Javascript tem uma sintaxe muito parecida a de Java** por estar baseado nele. Também é muito parecida a da linguagem C, de modo que se o leitor conhece alguma destas duas linguagens poderá manejar com facilidade com o código. De qualquer forma, nos seguintes capítulos vamos descrever toda a sintaxe com detalhes, para que os novatos não tenham nenhum problema com ela.

Comentários

Um comentário é uma parte de código que não é interpretada pelo navegador e cuja utilidade radica em facilitar a leitura ao programador. O programador, a medida que desenvolve o script, vai deixando frases ou palavras soltas, chamadas comentários, que ajudam a ele ou a qualquer outro a ler mais facilmente o script na hora de modificá-lo ou depurá-lo.

Já foi visto anteriormente algum comentário Javascript, mas agora vamos revê-los de novo. Existem dois tipos de comentários na linguagem. Um deles, a barra dupla, serve para comentar uma linha de código. O outro comentário podemos utilizar para comentar várias linhas e se indica com os signos /* para começar o comentário e */ para terminá-lo. Vejamos uns exemplos.

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

```
<SCRIPT>
//Este é um comentário de uma linha
/*Este comentário pode se expandir
por várias linhas.
As que quiser*/
</SCRIPT>
```

Maiúsculas e minúsculas

Em javascript se deve respeitar as maiúsculas e as minúsculas. Se nos equivocamos ao utilizá-las o navegador responderá com uma mensagem de erro de sintaxe. Por convenção os nomes das coisas se escrevem em minúsculas, salvo que se utilize um nome com mais de uma palavra, pois nesse caso se escreverão com maiúsculas as iniciais das palavras seguintes à primeira. Também se pode utilizar maiúsculas nas iniciais das primeiras palavras em alguns casos, como os nomes das classes, apesar de que já veremos mais adiante quais são estes casos e o que são as classes.

Separação de instruções

As distintas instruções que contém nossos scripts devem ser separadas convenientemente para que o navegador não indique os correspondentes erros de sintaxe. Javascript tem duas maneiras de separar instruções. A primeira é através do caractere pontoe vírgula (;) e a segunda é através de uma quebra de linha.

Por esta razão, as sentenças Javascript não necessitam acabar em ponto e vírgula a não ser que coloquemos duas instruções na mesma linha.

De qualquer forma, não é uma idéia ruim se acostumar a utilizar o ponto e vírgula depois de cada instrução, pois outras linguagens como Java ou C obrigam a utilizá-las e estaremos nos acostumando a realizar uma sintaxe mais parecida à habitual em torno de programações avançadas.

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

Variáveis Javascript

Uma variável é um espaço em memória onde se armazena um dado, um espaço onde podemos salvar qualquer tipo de informação que necessitemos para realizar as ações de nossos programas. Por exemplo, se nosso programa realiza somas, será muito normal guardarmos em variáveis as distintas parcelas que participam na operação e o resultado da soma. O efeito seria algo parecido a isto.

```
parcela1 = 23
parcela2 = 33
soma = parcela1 + parcela2
```

Neste exemplo temos três variáveis, parcela1, parcela2 e soma, onde guardamos o resultado. Vemos que seu uso para nós é como se tivéssemos um apartado onde salvar um dado e que se pode acessa-los colocando somente seu nome.

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Os nomes das variáveis devem ser construídos com caracteres alfanuméricos e o caractere sublinhado (_). A parte disso, há uma série de regras adicionais para construir nomes para variáveis. A mais importante é que tem de começar por um caractere alfabético ou sublinhado. Não podemos utilizar caracteres raros como o signo +, um espaço ou um $. Nomes admitidos para as variáveis poderiam ser:

Idade
PaísDeNascimento
_nome

Também há que evitar utilizar nomes reservados como variáveis, por exemplo não poderemos chamar a nossa variável palavras como return ou for, que já veremos que são utilizadas para estruturas da própria linguagem. Vejamos agora alguns nomes de variáveis que não está permitido utilizar

12meses
seu nome
return
pe%pe

Declaração de variáveis

Declarar variáveis consiste em definir e de passo informar ao sistema de que se vai utilizar uma variável. É um costume habitual nas linguagens de programação definir as variáveis que serão utilizadas nos programas e para isso, seguem-se algumas regras restritas. Porém, javascript ultrapassa muitas regras por ser uma linguagem mais livre na hora de programar e um dos casos no qual outorga um pouco de liberdade é na hora de declarar as variáveis, já que não estamos obrigados a fazê-lo, ao contrário do que acontece na maioria das linguagens de programação.

De qualquer forma, é aconselhável declarar as variáveis, além de um bom costume e para isso Javascript conta com a palavra var. Como é lógico, utiliza-se essa palavra para definir a variável antes de utilizá-la.

```
var operando1
var operando2
```

Também pode-se atribuir um valor à variável quando se está declarando

```
var operando1 = 23
var operando2 = 33
```

Também se permite declarar várias variáveis na mesma linha, sempre que se separem por vírgulas.

```
var operando1,operando2
```

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

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Âmbito das variáveis

Chama-se âmbito das variáveis ao lugar onde estas estão disponíveis. Em geral, quando declaramos uma variável fazemos com que esteja disponível no lugar onde se foi declarado, isto ocorre em todas as linguagens de programação e como javascript se define dentro de uma página web, **as variáveis que declaremos na página estarão acessíveis dentro dela**. Deste modo, não poderemos acessar às variáveis que tenham sido definidas em outra página. Este é o âmbito mais habitual de uma variável e chamamos a este tipo de variáveis globais à página, mesmo que não seja o único, já que também poderemos declarar variáveis em lugares mais dimensionados.

Variáveis globais

Como dissemos, as variáveis globais são as que estão declaradas no âmbito mais amplo possível, que em Javascript é uma página web. Para declarar uma variável global à página simplesmente faremos em um script, com a palavra *var*.

```
<SCRIPT>
var variávelGlobal
</SCRIPT>
```

As variáveis globais são acessíveis desde qualquer lugar da página, ou seja, a partir do script onde foi declarado e todos os demais scripts da página, incluindo os que manejam os eventos, como o onclick, que já vimos que podia ser incluído dentro de determinadas etiquetas HTML.

Variáveis locais

Também poderemos declarar variáveis em lugares mais dimensionados, como por exemplo, uma função. A estas variáveis chamaremos de locais. Quando se declarem variáveis locais somente poderemos acessá-las dentro do lugar aonde tenha sido declaradas, ou seja, se havíamos declarado em uma função somente poderemos acessá-la quando estivermos nessa função.

As variáveis podem ser locais em uma função, mas também podem ser locais a outros âmbitos, como por exemplo, um loop. Em geral, são âmbitos locais qualquer lugar dimensionado por chaves.

```
<SCRIPT>
function minhaFuncao) (){
   var variavelLocal
}
</SCRIPT>
```

No script anterior declaramos uma variável dentro de uma função, pelo qual esta variável somente terá validade dentro da função. Pode-se ver as chaves para dimensionar o lugar onde está definida essa função ou seu âmbito.

Não há problema em declarar uma variável local com o mesmo nome que uma global, neste caso a variável será visível desde toda a página, exceto no âmbito onde está declarada a variável local já que neste lugar esse nome de variável está ocupado pela local e é ela quem tem validade. Resumindo, em qualquer lugar da página, a variável que terá validade é a global. Menos no âmbito onde está declarada a variável local, que será ela que vai ter

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

validade.

```
<SCRIPT>
var numero = 2
function minhaFuncao (){
   var numero = 19
   document.write(numero) //imprime 19
}
document.write(numero) //imprime 2
</SCRIPT>
```

Um conselho para os principiantes poderia ser não declarar variáveis com os mesmos nomes, para que nunca tenha confusão sobre qual variável é a que tem validade em cada momento.

Diferenças entre utilizar var ou no

Como dissemos, em Javascript temos liberdade para declarar ou não as variáveis com a palavra var, mas os efeitos que conseguiremos em cada caso serão distintos. Na verdade, quando utilizamos var, estamos fazendo com que a variável que estamos declarando seja local ao âmbito onde se declara. Por outro lado, se não utilizamos a palavra var para declarar uma variável esta será global a toda a página, seja qual for o âmbito no qual tenha sido declarada.

No caso de uma variável declarada na página web, fora de uma função ou de qualquer outro âmbito mais reduzido, é indiferente se se declara ou não com var, desde um ponto de vista funcional. Isto é devido a que qualquer variável declarada fora do âmbito global a toda a página. A diferença pode ser apreciada em uma função por exemplo, já que se utilizamos var a variável será local à função e se não o utilizamos , a variável será global à página. Esta diferença é fundamental na hora de controlar corretamente o uso das variáveis na página, já que se não o fazemos em uma função poderíamos sobrescrever o valor de uma variável, perdendo o dado que pudesse conter previamente.

```
<SCRIPT>
var numero = 2
function minhaFuncao (){
   numero = 19
   document.write(numero) //imprime 19
}
document.write(numero) //imprime 2
//chamamos a funcao
minhaFuncao()
document.write(numero) //imprime 19
</SCRIPT>
```

Neste exemplo, temos uma variável global à página chamada numero, que contém um 2. Também temos uma função que utiliza a variável numero sem a ter declarado com var, pelo que a variável numero da funcao será mesma variável global numero declarada fora da função. Em uma situação com esta, ao executar a função se sobrescreverá a variável numero e o dado que havia antes de executar a função se perderá.

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

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

O que podemos salvar em variáveis

Em uma variável podemos introduzir vários tipos de informação, por exemplo, texto, números inteiros ou reais, etc. A estas distintas classes de informação conhecemos como tipos de dados. Cada um tem características e usos distintos, vejamos quais são os tipos de dados de Javascript.

Números
Para começar temos o tipo numérico, para salvar números como 9 ou 23.6

Cadeias
O tipo cadeia de caractere salva um texto. Sempre que escrevemos uma cadeia de caracteres devemos utilizar as aspas (").

Boleanos
Também contamos com o tipo boleano, que salva uma informação que pode valer como sim (true) ou não (false).

Por último seria relevante assinalar aqui que nossas variáveis podem conter coisas mais complicadas, como poderia ser um objeto, uma função, ou vazio (null) mas já o vermos mais adiante.

Na verdade nossas variáveis não estão forçadas a salvar um tipo de dados em concreto e portanto, não especificamos nenhum tipo de dados para uma variável quando a estamos declarando. Podemos introduzir qualquer informação em uma variável de qualquer tipo, inclusive podemos ir mudando o conteúdo de uma variável de um tipo a outro sem nenhum problema. Vamos ver isto com um exemplo.

```
var nome_cidade = "Salvador"
var revisado = true
nome_cidade = 32
revisado = "no"
```

Esta agilidade na hora de atribuir tipos às variáveis pode ser uma vantagem à princípio, sobretudo para pessoas inexperientes, mas a longo prazo pode ser uma fonte de erros já que dependendo do tipo que são as variáveis se comportarão de um modo ou outro e se não controlamos com exatidão o tipo das variáveis podemos encontrar um texto somado a um número. Javascript operará perfeitamente, e devolverá um dado, mas em alguns casos pode que não seja o que estávamos esperando. Sendo assim, mesmo que tenhamos liberdade com os tipos, esta mesma liberdade nos faz estar mais atentos a possíveis desajustes difíceis de detectar ao longo dos programas. Vejamos o que ocorreria no caso de somar letras e números.

```
var parcela1 = 23
var parcela2 = "33"
var soma = parcela1 + parcela2
document.write(soma)
```

Este script nos mostraria na página o texto 2333, que não se corresponde com a soma dos dois números, e sim com sua combinação, um atrás do outro.

Veremos algumas coisas mais referentes aos tipos de dados mais adiante.

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

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

Tipos de dados em Javascript

Em nossos scripts vamos manejar variáveis de diversas classes de informação, como textos ou números. Cada uma destas classes de informação é o tipo de dados. Javascript distingue entre três tipos de dados e todas as informações que se podem salvar em variáveis vão estar encaixadas em algum destes tipos de dados. Vejamos detalhadamente quais são estes três tipos de dados.

Tipo de dados numérico

Nesta linguagem só existe um tipo de dados numérico, ao contrário do que ocorre na maioria das linguagens mais conhecidas. Todos os números são portanto, do tipo numérico, independentemente da precisão que tenham ou se são números reais ou inteiros. Os números inteiros são números que não têm vírgula, como 3 ou 339. Os números reais são números fracionários, como 2.69 ou 0.25, que também se podem escrever em nota científica, por exemplo, 2.482e12.

Com Javascript também podemos escrever números em outras bases, como a hexadecimal. As bases são sistemas de numeração que utilizam mais ou menos dígitos para escrever os números. Existem três bases com as que podemos trabalhar:

- Base 10, é o sistema que utilizamos habitualmente, o sistema decimal. Qualquer número, por padrão, se entende que está escrito em base 10.
- Base 8, também chamado sistema octal, que utiliza dígitos do 0 ao 7. Para escrever um número em octal basta simplesmente escrever este número precedido de um 0, por exemplo 045.
- Base 16 ou sistema hexadecimal, é o sistema de numeração que utiliza 16 dígitos, os compreendidos entre o 0 e o 9 e as letras da A à F, para os dígitos que faltam. Para escrever um número em hexadecimal devemos escrevê-lo precedido de um zero e um xis, por exemplo, 0x3EF.

Tipo boleano

O tipo boleano, boolean em inglês, serve para salvar ou sim ou um não, ou com outras palavras, um verdadeiro ou falso. Utiliza-se para realizar operações lógicas, geralmente para realizar ações se o conteúdo de uma variável é verdadeiro ou falso.

Se uma variável é verdadeira, então: Executo umas instruções Si no Executo outras

Os dois valores que podem ter as variáveis boleanas são true ou false.

```
minhaBoleana = true
minhaBoleana = false
```

Tipo de dados cadeia de caracteres

O último tipo de dados é o que serve para salvar um texto. Javascript só tem um tipo de dados para salvar texto e nele, se podem introduzir qualquer número de caracteres. Um texto pode

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

estar composto de números, letras e qualquer outro tipo de caracteres e signos. Os textos se escrevem entre aspas, duplas ou simples.

```
meuTexto = "Miguel vai pescar"
meuTexto = '23%%$ Letras & *--*'
```

Tudo o que se coloca entre aspas, como nos exemplos anteriores é tratado como uma cadeia de caracteres independentemente do que coloquemos no interior das aspas. Por exemplo, em uma variável de texto podemos salvar números e nesse caso temos que ter em conta que as variáveis de tipo texto e as numéricas não são a mesma coisa e que enquanto as de numéricas nos servem para fazer cálculos matemáticos, as de texto não servem.

Caracteres de escape em cadeias de texto.

Existe uma série de caracteres especiais que servem para expressar em uma cadeia de texto determinados controles como pode ser uma quebra de linha ou um tabulador. Estes são os caracteres de escape e se escrevem com uma nota especial que começa por uma contra-barra (uma barra inclinada ao contrário da normal '\') e logo se coloca o código do caractere a mostrar.

Um caractere muito comum é a quebra de linha, que se consegue escrevendo \n. Outro caractere muito habitual é colocar umas aspas, pois se colocamos umas aspas sem seu caractere especial nos fechariam as aspas que colocamos para iniciar a cadeia de caracteres. Temos então que introduzir as aspas com \" ou \' (aspas duplas ou simples). Existem outros caracteres de escape, que veremos na tabela abaixo mais resumidos, apesar de que também há que destacar como caractere habitual o que se utiliza para escrever uma contra-barra, para não confundi-la com o início de um caractere de escape, que é a dupla contra-barra \\.

Tabela com todos os caracteres de escape

Quebra de linha: \n
Aspas simples: \'
Aspas dupla: \"
Tabulador: \t
Enter: \r
Avance de página: \f
Retroceder espaço: \b
Contra-barra: \\

Alguns destes caracteres provavelmente não os chegará a utilizar nunca, pois sua função é um pouco rara e também, às vezes pouco clara.

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

Operadores Javascript I

Ao desenvolver programas em qualquer linguagem se utilizam os operadores. Estes servem para fazer os cálculos e operações necessárias para realizar seus objetivos. Um programa que não realiza operações somente se pode limitar a fazer sempre o mesmo. É o resultado destas operações que faz com que um programa varie seu comportamento segundo os dados que

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

obtenha. Existem operações mais simples ou mais complexas, que se podem realizar com operandos de distintos tipos de dados, como números ou textos, veremos neste capítulo de maneira detalhada todos estes operadores.

Exemplos de uso de operadores

Antes de começar a numerar os distintos tipos de operadores vamos ver dois exemplos destes para ajudar a termos uma idéia mais exata do que são. No primeiro exemplo, vamos realizar uma soma utilizando o operador soma.

```
3 + 5
```

Esta é uma expressão muito básica que não tem muito sentido por si só. Faz a soma entre os dois operadores número 3 e 5, porém, não serve muito porque não se faz nada com o resultado. Normalmente combinam-se mais de um operador para criar expressões mais úteis. A expressão seguinte é uma combinação entre dois operadores, um realiza uma operação matemática e o outro serve para salvar o resultado.

```
minhaVariavel = 23 * 5
```

No exemplo anterior, o operador * se utiliza para realizar uma multiplicação e o operador = se utiliza para atribuir o resultado em uma variável, de modo que salvamos o valor para seu posterior uso.

Os operadores podem ser classificados segundo o tipo de ações que realizam. A seguir veremos cada um destes grupos de operadores e descreveremos a função de cada um.

Operadores aritméticos

São os utilizados para a realização de operações matemáticas simples como a soma, diferença ou multiplicação. Em javascript são os seguintes:

+ Soma de dois valores
- Diferença de dois valores, também se pode utilizar para mudar o sinal de um número se o utilizamos com um só operando -23
* Multiplicação de dois valores
/ Divisão de dois valores
% O resto da divisão de dois números (3%2 devolveria 1, o resto de dividir 3 entre 2)
++ Incremento em uma unidade, se utiliza com um só operando
-- Decremento em uma unidade, utilizado com um só operando

Exemplos

```
preço = 128 //introduzo um 128 na variável preço
unidades = 10 //outra atribuição, logo veremos operadores de atribuição
fatura = preço * unidades //multiplico preço por unidades, obtenho o valor fatura
resto = fatura % 3 //obtenho o resto de dividir a variável fatura por 3
preço++ //incrementa em uma unidade o preço (agora vale 129)
```

Operadores de atribuição

Servem para atribuir valores às variáveis, já utilizamos em exemplos anteriores o operador de atribuição =, mas existem outros operadores deste tipo, que provém da linguagem C e que

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

muitos dos leitores já conhecerão.

= Atribuição. Atribui a parte da direita do igual à parte da esquerda. À direita se colocam os valores finais e à esquerda geralmente se coloca uma variável onde queremos salvar o dado.
+= Atribuição com soma. Realiza a soma da parte da direita com a da esquerda e salva o resultado na parte da esquerda.
-= Atribuição com diferença
*= Atribuição da multiplicação
/= Atribuição da divisão
%= Se obtém o resto e se atribui

Exemplos
poupança = 7000 //atribui um 7000 à variável poupança
poupança += 3500 //incrementa em 3500 a variável poupança, agora vale 10500
poupança /= 2 //divide entre 2 minha poupança, agora ficam 5250

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

Operadores Javascript II

Operadores de cadeias

As cadeias de caracteres, ou variáveis de texto, também têm seus próprios operadores para realizar ações típicas sobre cadeias. Apesar do javascript ter somente um operador para cadeias se podem realizar outras ações com uma série de funções pré-definidas na linguagem que veremos mais adiante.

+ Concilia duas cadeias, pega a segunda cadeia a seguir da primeira.

Exemplo
cadeia1 = "ola"
cadeia2 = "mundo"
cadeiaConciliada = cadeia1 + cadeia2 //cadeia conciliada vale "olamundo"

Um detalhe importante que pode ser visto neste caso, é que o operador + serve para dois usos distintos, se seus operandos são números, os soma, mas se se trata de cadeias, as concilia. Isto ocorre em geral com todos os operadores que se repetem na linguagem, javascript é suficientemente esperto para entender que tipo de operação realizar mediante uma comprovação dos tipos que estão implicados nela.

Um caso que seria confuso é o uso do operador + quando se realiza a operação com operadores texto e numéricos misturados. Neste caso javascript assume que se deseja realizar uma conciliação e trata aos dois operandos como se tratasse de cadeias de caracteres, inclusive se a cadeia de texto que temos for um número. Isto veremos mais facilmente com o seguinte exemplo.

meuNumero = 23
minhaCadeia1 = "pedro"
minhaCadeia2 = "456"
resultado1 = meuNumero + minhaCadena1 //resultado1 vale "23pedro"

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

resultado2 = meuNumero + minhaCadeia2 //resultado2 vale "23456"
minhaCadeia2 += meuNumero //minhaCadena2 agora vale "45623"

Como podemos ver, também no caso do operador +=, se estamos tratando com cadeias de texto e números misturados, tratará aos dois operadores como se fossem cadeias.

Operadores lógicos

Estes operadores servem para realizar operações lógicas, que são aquelas que dão como resultado um verdadeiro ou um falso, e se utilizam para tomar decisões em nossos scripts. Ao invés de trabalhar com números, para realizar este tipo de operações se utilizam operandos boleanos, que conhecemos anteriormente, que são o verdadeiro (true) e o falso (false). Os operadores lógicos relacionam os operandos boleanos para dar como resultado outro operando boleano, tal como podemos ver no seguinte exemplo.

Se tenho fome e tenho comida, então irei comer

Nosso programa javascript utilizaria neste exemplo um operando boleano para tomar uma decisão. Primeiro irá ver se tenho fome, se é certo (true) irá ver se disponho de comida. Se são os dois são certos, poderá comer. No caso de que não tenha comida ou de que não tenha fome não comeria, assim como se não tenho fome nem comida. O operando em questão é o operando Y, que valerá verdadeiro (true) no caso de que os dois operandos sejam verdadeiros.

! Operador NO ou negação. Se é true passa a false e vice-versa.
&& Operador Y, se são os dois verdadeiros vale verdadeiro.
|| Operador O, vale verdadeiro se pelo menos um deles for verdadeiro.

Exemplo
meuBoleano = true
meuBoleano = !meuBoleano //meuBoleano agora vale false
tenhofome = true
tenhoComida = true
comoComida = tenhoFome && tenhoComida

Operadores condicionais

Servem para realizar expressões condicionais mais complexas que desejarmos. Estas expressões se utilizam para tomar decisões em função da comparação de vários elementos, por exemplo, se um número é maior que outro ou se são iguais. Os operadores condicionais se utilizam nas expressões condicionais para tomar decisões. Como estas expressões condicionais serão objeto de estudo mais adiante será melhor descrever os operadores condicionais mais adiante. De qualquer forma, aqui podemos ver a tabela de operadores condicionais.

== Comprova se dois números são iguais
!= Comprova se dois números são distintos
> Maior que, devolve true se o primeiro operador for maior que o segundo
< Menor que, é true quando o elemento da esquerda for menor que o da direita
>= Maior igual.
<= Menor igual

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

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Operadores Javascript III

Operadores a nível de bit

Estes são muito pouco correntes e é possível que você nunca chegue a utilizá-los. Seu uso se realiza para efetuar operações com zeros e uns. Tudo o que maneja um computador são zeros e uns, a pesar de nós usarmos números e letras para nossas variáveis na verdade estes valores estão escritos internamente em forma de zeros e uns. Em alguns casos, poderemos necessitar realizar operações tratando as variáveis como zeros e uns, e para isso, utilizaremos estes operandos. Nesta manual se tornaria um pouco extenso demais se realizássemos uma discussão sobre este tipo de operadores, mas aqui você poderá ver estes operadores por acaso algum dia fizer falta.

& Y de bits
^ Xor de bits
| O de bits
<< >> >>> >>>= >>= <<= Várias classes de mudanças

Precedência dos operadores

A avaliação de uma sentença das que vimos nos exemplos anteriores é bastante simples e fácil de interpretar, mas quando em uma sentença entram em jogo uma infinidade de operadores diferentes pode haver uma confusão na hora de interpretá-la e avaliar quais operadores são os que se executam antes que outros. Para marcar umas pautas na avaliação das sentenças e que estas se executem sempre igual e com sentido comum existe a precedência de operadores, que não é mais que a ordem pela qual se irão executando as operações que eles representam. À princípio todos os operadores se avaliam da esquerda para a direita, mas existem umas normas adicionais, pelas quais determinados operadores se avaliam antes que outros. Muitas destas regras de precedência foram tiradas das matemáticas e são comuns a outras linguagens, podemos vê-las a seguir.

() [] . Parêntesis, colchetes e o operador ponto que serve para os objetos
! - ++ -- negação, negativo e incrementos
* / % Multiplicação, divisão e módulo
+- Soma e diferença
<< >> >>> Mudanças a nível de bit
< <= > >= Operadores condicionais
== != Operadores condicionais de igualdade e desigualdade
& ^ | Lógicos a nível de bit
&& || Lógicos boleanos
= += -= *= /= %= <<= >>= >>>= &= ^= != Atribuição

Nos seguintes exemplos podemos ver como as expressões poderiam chegar a ser confusas, mas com a tabela de precedência de operadores poderemos entender sem erros qual é a ordem pela qual se executam.

12 * 3 + 4 - 8 / 2 % 3

Neste caso, primeiro se executam os operadores * / y %, da esquerda a direita, com o qual se realizariam estas operações. Primeiro a multiplicação e logo a divisão por estar mais à

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

esquerda do módulo.

36 + 4 - 4 % 3

Agora o módulo.

36 + 4 - 1

Por último as somas e as diferenças da esquerda para direita.

40 - 1

39

De qualquer forma, é importante se dar conta que o uso dos parênteses pode nos economizar muitos quebra-cabeças e, sobretudo, a necessidade de sabermos de memória a tabela de precedência dos operadores. Quando virmos pouco claro a ordem com a qual se executarão as sentenças podemos utilizá-las e assim, forçar que se avalie antes o pedaço da expressão que se encontra dentro dos parênteses.

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

Controle de tipos

Vimos para determinados operadores que é importante o tipo de dados que estão manejando, visto que se os dados são de um tipo irão realizar operações distintas que se são de outro.

Assim, quando utilizávamos o operador +, se se tratava de números, os somava, mas se se tratava de cadeias de caracteres, os conciliava. Vemos então, que o tipo dos dados que estamos utilizando sim que importa e que teremos que estar pendentes a este detalhe se quisermos que nossas operações se realizem tal como esperávamos.

Para comprovar o tipo de um dado se pode utilizar outro operador que está disponível a partir de javascript 1.1, o operador typeof, que devolve uma cadeia de texto que descreve o tipo do operador que estamos comprovando.

```
var boleano = true
var numerico = 22
var numerico_flutuante = 13.56
var texto = "meu texto"
var data = new Date()
document.write("<br>O tipo de boleano é: " + typeof boleano)
document.write("<br>O tipo de numerico é: " + typeof numerico)
document.write("<br>O tipo de numerico_flutuante é: " + typeof numerico_flutuante)
document.write("<br>O tipo de texto é: " + typeof texto)
document.write("<br>O tipo de data é: " + typeof data)
```

Este script dará como resultado o seguinte:

O tipo de boleano é: boolean

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

O tipo de numerico é: number
O tipo de numerico_flutuante é: number
O tipo de texto é: string
O tipo de data é: object

Neste exemplo podemos ver que se imprime na página os distintos tipos das variáveis. Estes podem ser os seguintes:

- boolean, para os dados boleanos. (True ou false)
- number, para os numéricos.
- string, para as cadeias de caracteres.
- object, para os objetos.

Queremos destacar apenas mais dois detalhes:

1) Os números, já tendo ou não parte decimal, são sempre do tipo de dados numéricos.

2) Uma das variáveis é um pouco mais complexa, é a variável data que é um objeto da classe Date(), que se utiliza para o manejo de datas nos scripts. Mais adiante a veremos, assim como os objetos.

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

Estruturas de controle

Os scripts vistos até agora foram tremendamente simples e lineares: iam-se executando as sentenças simples uma atrás da outra desde o princípio até o fim. Entretanto, isto não tem porque ser sempre assim, nos programas geralmente necessitaremos fazer coisas distintas, dependendo do estado de nossas variáveis realizar um mesmo processo muitas vezes sem escrever a mesma linha de código uma e outra vez.

Para realizar coisas mais complexas em nossos scripts se utilizam as estruturas de controle. Utilizando-as podemos realizar tomadas de decisões e loops. Nos seguintes capítulos vamos conhecer as distintas estruturas de controle que existem em Javascript.

Tomada de decisões

Servem para realizar umas ações ou outras em função do estado das variáveis. Ou seja, tomar decisões para executar umas instruções ou outras dependendo do que esteja ocorrendo neste instante em nossos programas.

Por exemplo, dependendo se o usuário que entra em nossa página for maior de idade ou não, podemos lhe permitir ou não ver os conteúdos de nossa página.

Se idade é maior que 18 então:
  Deixo-lhe ver o conteúdo para adultos
Se não
  Mando-lhe fora da página

Em javascript podemos tomar decisões utilizando dois enunciados distintos.

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

- IF
- SWITCH

Loops

Os loops se utilizam para realizar certas ações repetidamente. São muito utilizados em todos os níveis na programação. Com um loop podemos por exemplo, imprimir em uma página os números de 1 ao 100 sem a necessidade de escrever cem vezes a instrução a imprimir.

```
Desde o 1 até o 100
  Imprimir o número atual
```

Em javascript existem vários tipos de loops, cada um está indicado para um tipo de repetição distinto e são os seguintes:

- FOR
- WHILE
- DO WHILE

Como já assinalamos as estruturas de controle são muito importantes em Javascript e em qualquer linguagem de programação. É por isso que nos seguintes capítulos veremos cada uma destas estruturas detalhadamente, descrevendo seu uso e oferecendo alguns exemplos.

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

Estrutura IF

IF é uma estrutura de controle utilizada para tomar decisões. É uma condicional que realiza umas ou outras operações em função de uma expressão. Funciona da seguinte maneira, primeiro se avalia uma expressão se o resultado dá positivo realizam-se as ações relacionadas com o caso positivo.

A sintaxe da estrutura IF é a seguinte:

```
if (expressão) {
  ações a realizar em caso positivo
  ...
}
```

Opcionalmente se podem indicar ações a realizar no caso de que a avaliação da sentença dê resultados negativos.

```
if (expressão) {
  ações a realizar em caso positivo
  ...
} else {
  ações a realizar em caso negativo
  ...
}
```

Observemos várias coisas: Para começar vemos como com umas chaves englobam as ações

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

que queremos realizar no caso de que se cumpram ou não as expressões. Estas chaves devem ser colocadas sempre, exceto no caso de que somente haja uma instrução como ações a realizar, que são opcionais.

Outro detalhe que está descarado é a margem que colocamos em cada um dos blocos de instruções para executar nos casos positivos e negativos. Esta margem é totalmente opcional, somente fizemos assim para que a estrutura IF se compreenda de uma maneira mais visual. As quebras de linhas também não são necessárias e foram assim colocadas para que se veja melhor a estrutura. Poderíamos perfeitamente colocar toda a instrução IF na mesma linha de código, mas isso não ajudará que as coisas estejam claras. Nós, e qualquer programador, aconselhamos que utilizem as margens e as quebras de linhas necessárias para que as instruções possam ser entendidas melhor, hoje e dentro de um mês, quando já não será tão fácil lembrar o que foi feitos em seus scripts.

Vejamos um exemplo de condicionais IF.

```
if (dia == "Segunda-feira")
   document.write ("Que tenha um ótimo começo de semana")
```

Se for segunda-feira nos desejará uma ótima semana. Não fará nada em caso contrário. Como neste exemplo somente indicamos uma instrução para o caso positivo, não será preciso utilizar as chaves. Observe também no operador condicional que consta de dois signos "igual".

Vamos ver agora outro exemplo, um pouco mais comprido.

```
if (credito >= preço) {
   document.write("comprou o artigo " + novoArtigo) //mostro compra
   carrinho += novoArtigo //coloco o artigo no carrinho da compra
   credito -= preço //diminuo o crédito conforme o preço do artigo
} else {
   document.write("acabou o seu crédito") //informo que lhe falta dinhero
   window.location = "carrinhodacompra.html" //vou à página do carrinho
}
```

Este exemplo é um pouco mais complexo, e também um pouco fictício. O que faço é comprovar se tenho crédito para realizar uma suposta compra. Para isso, vejo se o crédito é maior ou igual que o preço do artigo, se é assim, informa da compra, coloco o artigo no carrinho e subtraio o preço ao crédito acumulado. Se o preço do artigo é superior ao dinheiro disponível, informo a situação e mando ao navegador à página onde se mostra seu carrinho da compra.

Expressões condicionais

A expressão a avaliar se coloca sempre entre parêntesis e está composta por variáveis que se combinam entre si mediante operadores condicionais. Lembramos que os operadores condicionais relacionavam duas variáveis e devolviam sempre um resultado boleano. Por exemplo, um operador condicional é o operador "é igual" (==), que devolve true no caso de que os operandos sejam iguais ou false no caso de que sejam distintos.

```
if (idade > 18)
   document.write("pode ver esta página para adultos")
```

Neste exemplo, utilizamos em operador condicional "é maior" (>). Neste caso, devolve true se a variável idade é maior que 18, com o que se executaria a linha seguinte que nos informa de

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

que se pode ver o conteúdo para adultos.

As expressões condicionais podem ser combinadas com as expressões lógicas para criar expressões mais complexas. Lembramos que as expressões lógicas são as que têm como operandos aos boleanos e que devolvem outro valor boleano. São os operadores de negação lógica, E lógico e O lógico.

```
if (bateria == 0 && redeEletrica = 0)
   document.write("seu laptop vai se apagar em segundos")
```

O que fazemos é comprovar se a bateria de nosso suposto computador está a zero (acabada) e também comprovamos se o computador não tem rede elétrica (se está fora da tomada). Logo, o operador lógico os relaciona com um E, de modo que se está sem bateria E sem rede elétrica, informo que o ordenador vai se apagar.

A lista de operadores que se podem utilizar com as estruturas IF, podem ser vistos no capítulo de operadores condicionais e operadores lógicos.

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

Estrutura IF (parte II)

Sentenças IF aninhadas

Para fazer estruturas condicionais mais complexas podemos aninhar sentenças IF, ou seja, colocar estruturas IF dentro de outras estruturas IF. Com um só IF podemos avaliar e realizar uma ação ou outra segundo duas possibilidades, mas se temos mais possibilidades que avaliar devemos aninhar Ifs para criar o fluxo de código necessário para decidir corretamente.

Por exemplo, se desejo comprovar se um número é maior ou igual ao outro, tenho que avaliar três possibilidades distintas. Primeiro, posso comprovar se os dois números são iguais, se são, já está resolvido o problema, mas se não são iguais ainda terei que ver qual dos dois é o maior. Vejamos este exemplo em código Javascript.

```
var numero1=23
var numero2=63
if (numero1 == numero2){
   document.write("Os dois números são iguais")
}else{
   if (numero1 > numero2) {
      document.write("O primeiro número é maior que o segundo")
   }else{
      document.write("O primeiro número é menor que o segundo")
   }
}
```

O fluxo do programa é como comentávamos antes, primeiro se avalia se os dois números são iguais. No caso positivo se mostra uma mensagem o informando. No caso contrário, já sabemos que são distintos, mas ainda devemos averiguar qual dos dois é maior. Para isso, faz-se outra comparação para saber se o primeiro é maior que o segundo. Se esta comparação dá

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

resultados positivos mostramos uma mensagem dizendo que o primeiro é maior que o segundo, em caso contrário indicaremos que o primeiro é menor que o segundo.

Voltamos a ressaltar que as chaves neste caso são opcionais, pois só se executa uma sentença para cada caso. Ademais, as quebras de linhas e as margens são opcionais em todo caso e nos serve somente para ver o código de uma maneira mais ordenada. Manter o código bem estruturado e escrito de una maneira compreensível é muito importante, já que nos fará a vida mais agradável na hora de programar e mais adiante quando tenhamos que revisar os programas. Neste manual utilizarei uma anotação como a que pode ser vista nas linhas anteriores, e também será visto adiante, ademais manterei essa anotação em todo momento. Isto sem lugar á dúvidas fará com que os códigos com exemplos sejam compreensíveis mais rapidamente, se não fizéssemos assim, seria um verdadeiro sacrifício lê-los. Esta mesma receita é aplicável aos códigos que você irá criar e o principal beneficiado será você mesmo e os companheiros que cheguem a ler seu código.

Operador IF

Existe um operador que ainda não vimos e é uma forma mais esquemática de realizar alguns IF simples. Provém da linguagem C, onde se escrevem muitas poucas linhas de código que resulta muito elegante. Este operador é um claro exemplo de economia de linhas e caracteres ao escrever os scripts. Será visto rapidamente, pois a única razão pela qual o incluo é para que saiba que existe e se o encontra em alguma ocasião por aí, você saiba identificá-lo e como funciona.

Um exemplo de uso de operador IF pode ser visto a seguir:

Variável = (condição) ? valor1 : valor2

Este exemplo não só realiza uma comparação de valores, como também atribui um valor a uma variável. O que faz é avaliar a condição (colocada entre parênteses) e se é positiva atribui o valor1 à variável e no caso contrário lhe atribui o valor 2. Vejamos um exemplo:

```
momento = (hora_atual < 12) ? "Antes de meio-dia" : "Depois de meio-dia"
```

Este exemplo olha se a hora atual é maior que 12. Sendo assim, quer dizer que agora é antes de meio-dia, assim que atribui "Antes de meio-dia" à variável momento. Se a hora é maior ou igual a 12 é que é depois de meio-dia, com o que se atribui o texto "Depois de meio-dia" à variável momento.

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

Estrutura SWITCH

É a outra opção disponível em Javascript para tomar decisões em função de distintos estados das variáveis. Esta expressão se utiliza quando temos múltiplas possibilidades como resultado da avaliação de uma sentença.

A estrutura SWITCH se incorporou a partir da versão 1.2 de Javascript (Netscape 4 e Internet Explorer 4). Sua sintaxe é a seguinte:

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

```
switch (expressão) {
   case valor1:
     Sentenças a executar se a expressão tem como valor a valor1
     break
   case valor2:
     Sentenças a executar se a expressão tem como valor a valor2
     break
   case valor3:
     Sentenças a executar se a expressão tem como valor a valor3
     break
   default:
     Sentenças a executar se o valor não é nenhum dos anteriores
}
```

A expressão se avalia, se vale valor1 se executam as sentenças relacionadas com esse caso. Se a expressão vale valor2 se executam as instruções relacionadas com esse valor e assim, sucessivamente, por tantas opções como desejarmos. Finalmente, para todos os casos não contemplados anteriormente se executa o caso por padrão.

A palavra break é opcional, mas se não a colocamos a partir de que se encontre coincidência com um valor se executarão todas as sentenças relacionadas com este e todas as seguintes. Ou seja, se em nosso esquema anterior não tivesse nenhuma expressão que valesse valor1, se executariam sentenças relacionadas com valor1 e também as relacionadas com valor2, valor3 e default.

Também é opcional a opção default ou opção por padrão.

Vejamos um exemplo de uso desta estrutura. Suponhamos que queremos indicar que dia da semana é. Se o dia é 1 (segunda-feira) colocamos uma mensagem indicando, se o dia é 2 (terça) devemos colocar uma mensagem diferente e assim, sucessivamente para cada dia da semana, menos no 6 (sábado) e 7 (domingo) que queremos mostrar a mensagem "é fim de semana". Para dias maiores que 7 indicaremos que esse dia não existe.

```
Switch (dia_da_semana) {
   case 1:
     document.write("É segunda-feira")
     break
   case 2:
     document.write("É terça-feira")
     break
   case 3:
     document.write("É quarta-feira")
     break
   case 4:
     document.write("É quinta-feira")
     break
   case 5:
     document.write("É sexta-feira ")
     break
   case 6:
   case 7:
     document.write("É fim de semana")
```


© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

```
      break
   default:
      document.write("Esse dia não existe")
}
```

O exemplo é relativamente simples, somente pode ter uma pequena dificuldade, consistente em interpretar o que passa no caso 6 e 7, que havíamos dito que tínhamos que mostrar a mesma mensagem. No caso 6 na verdade não indicamos nenhuma instrução, mas como tampouco colocamos um break se executará a sentença ou sentenças do caso seguinte, que correspondem com a sentença indicada no caso 7 que é a mensagem que informa que é fim de semana. Se o caso é 7 simplesmente se indica que é fim de semana, tal como se pretendia.

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

Loop FOR

O loop FOR se utiliza para repetir mais instruções um determinado número de vezes. Entre todos os loops, o FOR **costuma ser utilizado quando sabemos ao certo o número de vezes que queremos que seja executada a sentença**. A sintaxe do loop se mostra a seguir:

```
for (iniciação;condição;atualização) {
   sentenças a executar em cada repetição
}
```

O loop FOR tem três partes incluídas entre os parênteses. A primeira é a iniciação, que se executa somente ao começar a primeira repetição do loop. Nesta parte costuma-se colocar a variável que utilizaremos para levar a conta das vezes que se executa o loop.

A segunda parte é a condição, que se avaliará cada vez que comece a repetição do loop. Contém uma expressão para comprovar quando se deve deter o loop, ou melhor dizendo, a condição que se deve cumprir para que continue a execução do loop.

Por último temos a atualização, que serve para indicar as mudanças que quisermos executar nas variáveis cada vez que termine a interação do loop, antes de comprovar se se deve seguir executando.

Depois do for se colocam as sentenças que queremos que se executem em cada repetição, limitadas entre chaves.

Um exemplo de utilização deste loop pode ser visto a seguir, onde se imprimirão os números do 0 ao 10.

```
var i
for (i=0;i<=10;i++) {
   document.write(i)
}
```

Neste caso se inicia a variável i a 0. Como condição para realizar uma repetição, tem de se cumprir que a variável i seja menor ou igual a 10. Como atualização se incrementará em 1 a

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

variável i.

Como se pode comprovar, **este loop é muito potente, já que em uma só linha podemos indicar muitas coisas distintas e muito variadas**.

Por exemplo, se queremos escrever os número do 1 ao 1.000 de dois em dois, será escrito o seguinte loop:

```
for (i=1;i<=1000;i+=2)
   document.write(i)
```

Se observarmos, em cada repetição atualizamos o valor de i incrementando-lhe em 2 unidades.

Outro detalhe: não utilizamos as chaves englobando as instruções do loop FOR porque só tem uma sentença e neste caso não é obrigatório, tal como acontecia com as instruções do IF.

Se quisermos contar decrescentemente do 343 ao 10 utilizaríamos este loop.

```
for (i=343;i>=10;i--)
   document.write(i)
}
```

Neste caso decrementamos em uma unidade a variável i em cada repetição.

Exemplo

Vamos fazer uma pausa para assimilar o loop for com um exercício que não implica nenhuma dificuldade se entendemos o funcionamento do loop.

Trata-se de fazer um loop que escreva em uma página web os cabeçalhos desde <H1> até <H6> com um texto que ponha "Cabeçalho de nível x".

O que desejamos escrever em uma página web mediante Javascript é o seguinte:

```
<H1>Cabeçalho de nível 1</H1>
<H2>Cabeçalho de nível 2</H2>
<H3> Cabeçalho de nível 3</H3>
<H4> Cabeçalho de nível 4</H4>
<H5> Cabeçalho de nível 5</H5>
<H6> Cabeçalho de nível 6</H6>
```

Para isso, temos que fazer um loop que comece em 1 e termine em 6 e em cada repetição escreveremos o respectivo cabeçalho.

```
for (i=1;i<=6;i++) {
   document.write("<H" + i + ">Cabeçalho de nível " + i + "</H" + i + ">")
}
```

Este script pode ser visto em funcionamento aqui.

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

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Loops WHILE e DO WHILE

Vejamos agora os dois tipos de loops WHILE que podemos utilizar em Javascript e os usos de cada um.

Loop WHILE

Estes loops se utilizam quando queremos repetir um número indefinido de vezes a execução de umas sentenças, sempre que se cumpra uma condição. Sim, que é mais simples de compreender que o loop FOR, pois não incorpora na mesma linha a iniciação das variáveis, sua condição para continuar executando e sua atualização. Somente se indica, como veremos a seguir, a condição que tem que se cumprir para que se realize uma repetição.

```
while (condição){
   sentenças a executar
}
```

Um exemplo de código onde se utiliza este loop pode ser visto a seguir:

```
var color = ""
while (color != "vermelho")
   color = me dá uma cor
}
```

Este é um exemplo do mais simples do que se pode fazer com um loop while. O que faz é pedir que o usuário introduza uma cor, mas que não seja a cor vermelha. Para executar um loop como este primeiro temos que iniciar a variável que vamos utilizar na condição de repetição do loop. Com a variável iniciada podemos escrever o loop, que comprovará para executar que a cor da variável seja diferente de "vermelha". Em cada repetição do loop pede-se uma nova cor ao usuário para atualizar a variável cor e termina-se a repetição, com o que retornamos ao princípio do loop, onde temos que voltar a avaliar se o que há na variável cor é "vermelha" e assim sucessivamente enquanto não seja introduzido como cor o texto "vermelho". Obviamente, a expressão "me dá uma cor" não é Javascript, mas como ainda não sabemos como escrever isso em Javascript, é melhor vê-lo mais adiante.

Loop DO...WHILE

É o último dos loops que há em Javascript. Utiliza-se geralmente quando não sabemos quantas vezes haverá de se executar o loop, assim como o loop WHILE, com a diferença de que sabemos ao certo que o loop pelo menos se executará uma vez.

Este tipo de loop se introduziu em Javascript 1.2, portanto, nem todos os navegadores o suportam, somente os de versão 4 ou superior. Em qualquer caso, qualquer código que se queira escrever com DO...WHILE pode ser escrito também utilizando um loop WHILE, com o qual em navegadores antigos deverá traduzir o loop DO...WHILE por um loop WHILE.

A sintaxe é a seguinte:

```
do {
   sentenças do loop
```


© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

```
} while (condição)
```

O loop se executa sempre uma vez e ao final se avalia a condição para dizer se se executa outra vez o loop ou se termina sua execução.

Vejamos o exemplo que escrevemos para um loop WHILE neste outro tipo de loop:

```
var color
do {
   color = me dá uma cor
} while (color != "vermelho")
```

Este exemplo funciona exatamente igual que o anterior, exceto que não tivemos que iniciar a variável cor antes de introduzirmos no loop. Pede uma cor contanto que a cor introduzida seja diferente de "vermelho".

Exemplo

Vamos ver a seguir um exemplo mais prático sobre como trabalhar com um loop WHILE. Como é muito difícil fazer exemplos práticos, com o pouco que sabemos sobre Javascript, vamos adiantar uma instrução que ainda não conhecemos.

Neste exemplo vamos declarar uma variável e iniciá-la a 0. Logo, iremos somando a essa variável um número aleatório do 1 ao 100 até somarmos 1.000 ou mais, imprimindo o valor da variável soma depois de cada operação. Será necessário utilizar o loop WHILE porque não sabemos exatamente o número de repetições que teremos que realizar.

```
var soma = 0
while (soma < 1000){
   soma += parseInt(Math.random() * 100)
   document.write (soma + "<br>")
}
```

Supomos que no que diz respeito ao loop WHILE não haverá problemas, mas onde sim que pode haver é na sentença utilizada para tomar um número aleatório. Entretanto, não é necessário explicar aqui a sentença porque já temos planejado fazer mais adiante.

Podemos ver uma página com o exemplo em funcionamento.

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

Break e Continue

De maneira adicional ao uso das distintas estruturas de loop se podem utilizar duas instruções para

- Deter a execução de um loop e sair dele
- Deter a repetição atual e voltar ao princípio do loop.

São as instruções break e continue.

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Break

Detém-se um loop utilizando a palavra break. Deter um loop significa sair dele e deixá-lo todo como está para continuar com o fluxo do programa imediatamente depois do loop.

```
for (i=0;i<10;i++){
   document.write (i)
   escribe = diga-me se continuo
   if (escribe == "no")
      break
}
```

Este exemplo escreve os números do 0 ao 9 e em cada repetição do loop, pergunta ao usuário se deseja continuar. Se o usuário diz qualquer coisa continua, exceto quando diz "não" que então se sai do loop e deixa a conta por onde havia deixado.

Continue

Serve para voltar ao princípio do bucle em qualquer momento, sem executar as linhas que haja por debaixo da palavra continue.

```
var i=0
while (i<7){
   incrementar = diga-me se incremento
   if (incrementar == "no")
      continue
   i++
}
```

Este exemplo, em condições normais contaria até desde i=0 até i=7, mas cada vez que se executa o loop pergunta ao usuário se deseja incrementar a variável ou não. Se se introduz "não" se executa a sentença continue, com o qual se volta ao princípio do loop sem chegar a incrementar em 1 a variável i, já que se ignoram as sentenças que hajam por debaixo do continue.

Exemplo

Um exemplo mais prático sobre estas instruções pode ser visto a seguir. Trata-se de um loop FOR planejado para chegar até 1.000, mas que vamos pará-lo com break quando chegarmos a 333.

```
for (i=0;i<=1000;i++){
   document.write(i + "<br>")
   if (i==333)
      break;
}
```

Podemos ver uma página com o exemplo em funcionamento.

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

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Loops aninhados em Javascript

Aninhar um loop consiste colocar esse loop dentro de outro. O aninhamento de loops é necessário para fazer determinados processamentos um pouco mais complexos dos que os que vimos nos exemplos anteriores e com certeza que em sua experiência como programador já deve ter utilizado ou senão, utilizará em um futuro.

Um loop aninhado tem a estrutura como a que segue abaixo. Vamos explicá-lo através destas linhas:

```
for (i=0;i<10;i++){
   for (j=0;j<10;j++) {
   document.write(i + "-" + j)
   }
}
```

A execução funcionará da seguinte forma. Para começar inicia-se o primeiro loop, com o que a variável i valerá 0 e a seguir inicia-se o segundo loop, com o que a variável j valerá também 0. Em cada repetição imprime-se o valor da variável i, um hífen ("-") e o valor da variável j, como as duas variáveis valem 0, se imprimirá o texto "0-0" na página web.

O loop que está aninhado (mais para dentro) é o que mais se executa, neste exemplo, para cada repetição do loop mais externo, o loop aninhado se executará por completo uma vez, ou seja, fará suas 10 repitições. Na página web se escreveriam estes valores, na primeira repetição do loop externo e desde o princípio:

0-0
0-1
0-2
0-3
0-4
0-5
0-6
0-7
0-8
0-9

Para cada repetição do loop externo se executarão as 10 repetições do loop interno ou aninhado. Vimos a primeira repetição, agora veremos as seguintes repetições do loop externo. Em cada uma acumula uma unidade na variável i, com o que sairiam estes valores.

1-0
1-1
1-2
1-3
1-4
1-5
1-6
1-7
1-8
1-9

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

E logo estes:

2-0
2-1
2-2
2-3
2-4
2-5
2-6
2-7
2-8
2-9

Assim até que terminem os dois loops, que seria quando se alcançasse o valor 9-9.

Vejamos um exemplo muito parecido ao anterior, embora um pouco mais útil. Trata-se de imprimir na página todas as tabelas multiplicar. Do 1 ao 9, ou seja, a tabela do 1, a do 2, do 3...

```
for (i=1;i<10;i++){
   document.write("<br><b>La tabla del " + i + ":</b><br>")
   for (j=1;j<10;j++) {
     document.write(i + " x " + j + ": ")
     document.write(i*j)
     document.write("<br>")
   }
}
```

Com o primeiro loop controlamos a tabela atual e com o segundo loop a desenvolvemos. No primeiro loop escrevemos um título, em negrito, indicando a tabela que estamos escrevendo, primeiro a do 1 e logo as outras em ordem crescente até o 9. Com o segundo loop escrevo cada um dos valores de cada tabela. Pode-se ver o exemplo em funcionamento neste link.

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

Funções em Javascript

Agora veremos um assunto muito importante, sobretudo para os que nunca programaram e estão dando seus primeiros passos no mundo da programação com Javascript, já que veremos um conceito novo, o de função, e os usos que têm. Para os que já conhecem o conceito de função também será um capítulo útil, pois também veremos a sintaxe e o funcionamento das funções em Javascript.

O que é uma função

Na hora de fazer um programa levemente grande existem determinados processos que se podem conceber de forma independente, e que são mais simples de resolver que o problema inteiro. Ademais, estes costumam ser realizados repetidas vezes ao longo da execução do programa. Estes processos podem se agrupar em uma função, definida para que não tenhamos que repetir uma vez ou outra esse código em nossos scripts, e sim, simplesmente

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

chamamos a função, e ela se encarrega de fazer tudo o que deve.

Portanto, podemos ver uma função como uma série de instruções que englobamos dentro de um mesmo processo. Este processo poderá logo ser executado desde qualquer outro site somente ao ser chamado. Por exemplo, em uma página web pode haver uma função para mudar a cor de fundo e de qualquer ponto da página poderíamos chamá-la para que nos mude a cor quando desejarmos.

As funções utilizam-se constantemente, não só as que você escreve como também as que já estão definidas no sistema, pois todas as linguagens de programação têm um montão de funções para realizar processos habituais como, por exemplo, obter a hora, imprimir uma mensagem na tela ou converter variáveis de um tipo a outro. Já vimos alguma função em nossos simples exemplos anteriores quando fazíamos um document.write() na verdade estávamos chamando à função write() associada ao documento da página que escreve um texto na página. Nos capítulos de funções vamos ver primeiro como realizar nossas próprias funções e como chamá-las logo. Ao longo do livro veremos muitas das funções definidas em Javascript que devemos utilizar para realizar distintos tipos de ações habituais.

Como se escreve uma função

Uma função deve-se definir com uma sintaxe especial que vamos conhecer a seguir:

```
function nomefuncao (){
  instruções da função
  ...
}
```

Primeiro escreve-se a palavra função, reservada para este uso. Seguidamente se escreve o nome da função, que como os nomes de variáveis podem ter números, letras e algum caractere adicional como um hífen abaixo. A seguir se colocam entre chaves as diferentes instruções da função. As chaves no caso das funções não são opcionais, ademais é útil colocá-las sempre como se vê no exemplo, para que seja visto facilmente a estrutura de instruções que engloba a função.

Vejamos um exemplo de função para escrever na página uma mensagem de boas vindas dentro de etiquetas <H1> para que fique mais ressaltado.

```
function escreverBoasvindas(){
  document.write("<H1>Olá a todos</H1>")
}
```

Simplesmente escreve na página um texto, é uma função tão simples que o exemplo não expressa suficientemente o conceito de função, mais já veremos outras mais complexas. As etiquetas H1 não se escrevem na página, e sim são interpretadas como o significado da mesma, neste caso que escrevemos uma cabeçalho de nível 1. Como estamos escrevendo em uma página web, ao colocar etiquetas HTML se interpretam como o que são.

Como chamar a uma função

Quando se chamam às funções: Para executar uma função temos que chamá-la em qualquer parte da página, com isso conseguiremos que se executem todas as instruções que tem a função entre as duas chaves. Para executar a função utilizamos seu nome seguido dos

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

parênteses.

NomeDaFuncao()

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

Onde colocamos as funções

À princípio, podemos colocar as funções em qualquer parte da página, é claro que sempre entre etiquetas <SCRIPT>. Não obstante, existe uma limitação na hora de colocá-la em relação aos lugares de onde for chamada. O mais normal é colocar a função antes de qualquer chamada à mesma e assim, certamente não iremos nos enganar.

Teoricamente, a função deve-se definir no bloco <SCRIPT> onde esteja a chamada à função, embora seja indiferente se a chamada se encontrar antes ou depois da função, dentro do mesmo bloco <SCRIPT>.

```
<SCRIPT>
minhaFuncao()
function minhaFuncao(){
   //faço algo...
   document.write("Isto está bem")
}
</SCRIPT>
```

Este exemplo funciona corretamente porque a função está declarada no mesmo bloco que sua chamada.

Também é válido que a função se encontre em um bloco <SCRIPT> anterior ao bloco onde está a chamada.

```
<HTML>
<HEAD>
  <TITLE>MINHA PÁGINA</TITLE>
<SCRIPT>
function minhaFuncao(){
   //faço algo...
   document.write("Isto está bem")
}
</SCRIPT>
</HEAD>
<BODY>

<SCRIPT>
minhaFuncao()
</SCRIPT>

</BODY>
</HTML>
```


© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Vemos um código completo sobre como poderia ser uma página web onde as funções estão no cabeçalho. Um lugar muito bom para colocá-las, porque se supõem que no cabeçalho ainda não vão utilizar e sempre poderemos desfrutar deles no corpo porque certamente já foram declarados.

Este último em compensação seria um erro:

O que será um erro é uma chamada a uma função que se encontra declarada em um bloco <SCRIPT> posterior.

```
<SCRIPT>
minhaFuncao()
</SCRIPT>

<SCRIPT>
function minhaFuncao(){
   //faço algo...
   document.write("Isto está bem")
}
</SCRIPT>
```

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

Parâmetros das funções

As estruturas que vimos anteriormente sobre funções não são as únicas que devemos aprender para manejá-las em toda a sua potência. As funções também têm uma entrada e uma saída, que se podem utilizar para receber e devolver dados.

Parâmetros

Os parâmetros se usam para mandar valores à função, com os quais ela trabalhará para realizar as ações. São os valores de entrada que recebem uma função. Por exemplo, uma função que realizasse uma soma de dois números teria como parâmetros a esses dois números. Os dois números são a entrada, assim como a saída seria o resultado, mais isso será visto mais tarde.

Vejamos um exemplo anterior no qual criávamos uma função para mostrar uma mensagem de boas vindas à página web, mas que agora passaremos um parâmetro que vai conter o nome da pessoa a qual se vai saudar.

```
function escreverBoasvindas(nome){
   document.write("<H1>Ola " + nome + "</H1>")
}
```

Como podemos ver no exemplo, para definir na função um parâmetro temos que por o nome da variável que vai armazenar o dado que passarmos. Essa variável, que neste caso se chama nome, terá como valor o dado que passarmos a esta função quando a chamarmos, alé disso, a variável terá vida durante a execução da função e deixará de existir quando a função terminar sua execução.

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Para chamar a uma função que tem parâmetros coloca-se entre parêntesis o valor do parâmetro. Para chamar à função do exemplo haveria que escrever:

escreverBoasvindas("Alberto Garcia")

Ao chamar à função assim, o parâmetro nome toma como valor "Alberto Garcia" e ao escrever a saudação na tela escreverá "Olá Alberto Garcia" entre etiquetas <H1>.

Os parâmetros podem escrever qualquer tipo de dados, numérico, textual, boleano ou um objeto. Realmente não especificamos o tipo do parâmetro, por isso devemos ter um cuidado especial ao definir as ações que realizamos dentro da função e ao passar valores à função para assegurarmos que tudo é conseqüente com os tipos de nossas variáveis ou parâmetros.

Múltiplos parâmetros

Uma função pode receber tantos parâmetros quanto quisermos e para expressá-lo colocam-se os parâmetros separados por vírgulas dentro dos parênteses. Vejamos rapidamente a sintaxe para que a função de antes receba dois parâmetros, primeiro, o nome a quem saudar e segundo, a cor do texto.

```
function escreverBoasvindas(nome,corTexto){
   document.write("<FONT color=" + corTexto + ">)
   document.write("<H1>Olá " + nome + "</H1>")
   document.write("</FONT>")
}
```

Chamaríamos à função com esta sintaxe. Entre parênteses colocaremos os valores dos parâmetros.

```
var meuNome = "Pedro"
var minhaCor = "red"
escreverBoasvindas(meuNome,minhaCor)
```

Coloquei entre parênteses, duas variáveis no lugar de dois textos entre aspas. Quando colocamos variáveis entre os parâmetros na verdade o que estamos passando à função são os valores que contém as variáveis e não as mesmas variáveis.

Parâmetros passam-se por valor

Para seguir a linha do uso de parâmetros em nossos programas Javascript, temos que indicar que os parâmetros das funções se passam por valor. Isto quer dizer que mesmo que modifiquemos um parâmetro em uma função a variável original que havíamos passado não mudará seu valor. Pode-se ver facilmente com um exemplo.

```
function passoPorValor(meuParametro){
   meuParametro = 32
   document.write("mudei o valor a 32")
}
var minhaVariavel = 5
passoPorValor(minhaVariavel)
document.write ("o valor da variavel e: " + minhaVariavel)
```


© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

No exemplo, temos uma função que recebe um parâmetro, e que modifica o valor do parâmetro atribuindo-lhe o valor 32. Também temos uma variável, que iniciamos a 5 e posteriormente chamamos à função passando esta variável como parâmetro. Como dentro da função modificamos o valor do parâmetro poderia acontecer da variável original mudasse de valor, mas como os parâmetros não modificam o valor original das variáveis, esta não muda de valor. Deste modo, ao imprimir na tela o valor de minhaVariavel se imprimirá o número 5, que é o valor original da variável, no lugar de 32 que era o valor col o que havíamos atualizado o parâmetro.

Em javascript somente se podem passar as variáveis por valor.

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

Valores de retorno

As funções também podem retornar valores, de modo que ao executar a função poderá se realizar ações e dar um valor como saída. Por exemplo, uma função que calcula o quadrado de um número terá como entrada -tal como vimos- a esse número e como saída terá o valor resultante de encontrar o quadrado desse número. Uma função que devolva o dia da semana teria como saída em dia da semana.

Vejamos un exemplo de função que calcula a média de dois números. A função receberá os dois números e retornará o valor da média.

```
function media(valor1,valor2){
   var resultado
   resultado = (valor1 + valor2) / 2
   return resultado
}
```

Para especificar o valor que retornará a função se utiliza a palavra return seguida do valor que se deseja devolver. Neste caso se devolve o conteúdo da variável resultado, que contém a média dos dois números.

Para receber os valores que devolve uma função se coloca o operador de atribuição =. Para ilustrar isto, pode-se ver este exemplo, que chamará à função média() e salvará o resultado da média em uma variável para logo, imprimi-la na página.

```
var minhaMedia
minhaMedia = media(12,8)
document.write (minhaMedia)
```

Múltiplos return

Em uma mesma função podemos colocar mais de um return. Logicamente só vamos poder retornar uma coisa, mas dependendo do que tenha acontecido na função poderá ser de um tipo ou de outro, com uns dados ou outros.

Nesta função podemos ver um exemplo de utilização de múltiplos return. Trata-se de uma

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

função que devolve um 0 se o parâmetro recebido era par e o valor do parâmetro se este era ímpar.

```
function multiploReturn(numero){
   var resto = numero % 2
   if (resto == 0)
     return 0
   else
     return numero
}
```

Para averiguar se um número é par encontramos o resto da divisão ao dividi-lo entre 2. Se o resto é zero é porque era par e devolvemos um 0, em caso contrário -o número é ímpar- devolvemos o parâmetro recebido.

Âmbito das variáveis em funções

Dentro das funções podemos declarar variáveis, inclusive os parâmetros são como variáveis que se declaram no cabeçalho da função e que se iniciam ao chamar a função. Todas as variáveis declaradas em uma função são locais a essa função, ou seja, somente terão validade durante a execução da função.

Podemos declarar variáveis em funções que tenham o mesmo nome que uma variável global à página. Então, dentro da função a variável que terá validade é a variável local e fora da função terá validade a variável global à página.

Em troca, se não declaramos as variáveis nas funções se entenderá por javascript que estamos fazendo referência a uma variável global à página, de modo que se não está criada, a variável a cria, mas sempre global à página no lugar de local à função.

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

Arrays em Javascript

Nas linguagens de programação existem estruturas de dados especiais que nos servem para salvar informações mais complexas do que simples variáveis. Uma estrutura típica em todas as linguagens é o Array, que é como uma variável onde podemos introduzir vários valores, ao invés de somente um como ocorre com as variáveis normais.

Os arrays nos permitem salvar várias variáveis e acessá-las de maneira independente, é como ter uma variável com distintos compartimentos onde podemos introduzir dados distintos. Para isso utilizamos um índice que nos permite especificar o compartimento ou posição ao qual estamos nos referindo.

Os arrays foram introduzidos em versões Javascript 1.1 ou superiores, ou seja, somente podemos utilizá-los a partir dos navegadores 3.0. Para navegadores antigos se pode simular o array utilizando sintaxe de programação orientada a objetos, mas dada a complexidade desta tarefa, pelo menos no momento em que nos encontramos e as poucas ocasiões que os deveremos utilizar, vamos ver como utilizar o autêntico array de Javascript.

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Criação de Arrays

O primeiro passo para utilizar um array é criá-lo. Para isso utilizamos um objeto Javascript já implementado no navegador. Veremos adiante um tema para explicar o que é a orientação a objetos, embora não será necessário para poder entender o uso dos arrays. Esta é a sentença para criar um objeto array:

```
var meuArray = new Array()
```

Isto cria um array na página que está se executando. O array se cria sem nenhum conteúdo, ou seja, não terá nenhum campo ou compartimento criado. Também podemos criar o array especificando o número de compartimentos que vai ter.

```
var meuArray = new Array(10)
```

Neste caso indicamos que o array vai ter 10 posições, ou seja, 10 campos onde salvar dados.

É importante observarmos que a palavra Array em código Javascript se escreve com a primeira letra em maiúscula. Como em Javascript as maiúsculas e minúsculas sim que importam, se escrevemos em minúscula não funcionará.

Podemos introduzir no array qualquer dado, tanto se indicamos ou não o número de campos do array. Se o campo está criado se introduz simplesmente e se o campo não estava ccriado se cria e logo, se introduz o dado, com o qual o resultado final será o mesmo. Esta criação de campos é dinâmica e se produz ao mesmo tempo, que os scripts se executam. Vejamos a seguir como introduzir valores em nossos arrays.

```
meuArray[0] = 290
meuArray[1] = 97
meuArray[2] = 127
```

Introduzem-se indicando entre colchetes o índice da posição onde queríamos salvar o dado. Neste caso introduzimos 290 na posição 0, 97 na posição 1 e 127 na 2. Os arrays começam sempre na posição 0, portanto, um array que tenha por exemplo 10 posições, terá campos do 0 ao 9. Para recolher dados de um array fazemos da mesma forma: colocando entre colchetes o índice da posição a qual queremos acessar. Vejamos como se imprimiria na tela o conteúdo de um array.

```
var meuArray = new Array(3)

meuArray[0] = 155
meuArray[1] = 4
meuArray[2] = 499

for (i=0;i<3;i++){
   document.write("Posição " + i + " do array: " + meuArray[i])
   document.write("<br>")
}
```

Criamos um array com três posições, logo introduzimos um valor em cada uma das posições do array e finalmente imprimimos. Em geral, o percurso por arrays para imprimir suas posições ou qualquer outra coisa se faz utilizando loops. Neste caso utilizamos um loop FOR

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

que vai desde o 0 até o 2.

Podemos ver o exemplo em funcionamento em outra página.

Tipos de dados nos arrays

Nos campos dos arrays podemos salvar dados de qualquer tipo. Podemos ver um array onde introduzimos dados de tipo caractere.

```
meuArray[0] = "Ola"
meuArray[1] = "a"
meuArray[2] = "todos"
```

Inclusive, em Javascript podemos salvar distintos tipos de dados nos campos de um mesmo array. Ou seja, podemos introduzir números em uns campos, textos em outros, boleanos ou qualquer outra coisa que desejarmos.

```
meuArray[0] = "criarweb.com"
meuArray[1] = 1275
meuArray[1] = 0.78
meuArray[2] = true
```

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

Longitude dos Arrays

Todos os arrays em javascript, além de armazenar o valor de cada uma de suas posições também armazenam o número de posições que têm. Para isso, utilizam uma propriedade do objeto array, a propriedade length. Já veremos o que é uma propriedade em objetos, mas para nosso caso podemos imaginar que é como uma variável, adicional às posições, que armazena um número igual ao número de campos do array.

Para acessar a uma propriedade de um objeto há que utilizar o operador ponto. Escreve-se o nome do array que queremos acessar ao número de posições que tem, sem colchetes nem parênteses, seguido de um ponto e a palavra length.

```
var meuArray = new Array()

meuArray[0] = 155
meuArray[1] = 499
meuArray[2] = 65

document.write("Longitude do array: " + meuArray.length)
```

Este código imprimiria na tela o número de posições do array, que neste caso é 3. Recordamos que um array com 3 posições abarca desde a posição 0 a 2.

É muito habitual que se utilize a propriedade length para poder percorrer um array por todas suas posições. Para ilustrá-lo vamos ver um exemplo de percurso por este array para mostrar seus valores.

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

```
for (i=0;i<meuArray.length;i++){
   document.write(meuArray[i])
}
```

Há que observar que o loop for se executa sempre que i valer menos que a longitude do array, extraída de sua propriedade length.

O seguinte exemplo nos servirá para conhecer melhor os percursos pelos arrays, o funcionamento da propriedade length e a criação dinâmica de novas posições. Vamos criar um array com 2 posições e preencher seu valor. Posteriormente, introduziremos um valor na posição 5 do array. Finalmente, imprimiremos todas as posições do array para ver o que acontece.

```
var meuArray = new Array(2)

meuArray[0] = "Colômbia"
meuArray[1] = "Estados Unidos"

meuArray[5] = "Brasil"

for (i=0;i<meuArray.length;i++){
   document.write("Posição " + i + " do array: " + meuArray[i])
   document.write("<br>")
}
```

O exemplo é simples. Pode-se apreciar que fazemos um percurso pelo array desde 0 até o número de posições do array (indicado pela propriedade length). No percurso vamos imprimindo o número da posição seguido do conteúdo do array nesta posição. Mas podemos ter uma dúvida ao perguntarmos qual será o número de elementos deste array, já que o havíamos declarado com 2 e logo lhe introduzimos um terceiro na posição 5. Ao ver a saída do programa poderemos contestar nossas perguntas. Será algo parecido a isto:

```
Posição 0 do array: Colômbia
Posição 1 do array: Estados Unidos
Posição 2 do array: null
Posição 3 do array: null
Posição 4 do array: null
Posição 5 do array: Brasil
```

Pode-se ver claramente que o número de posições é 6, da 0 a 5. O que ocorreu é que ao introduzir um dado na posição 5, todas os campos que não estavam criados até o quinto se criaram também.

As posições da 2 a 4 estão sem iniciar. Neste caso nosso navegador escreveu a palavra null para expressar isto, mas outros navegadores poderão utilizar a palavra undefined. Veremos mais adiante qual é este null e onde o podemos utilizar, o importante agora é compreender como trabalham os arrays e utiliza-los corretamente.

Podemos ver o efeito deste script em seu navegador em uma página a parte.

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

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Arrays multidimensionais

Os arrays multidimensionais são estruturas de dados que armazenam os valores em mais de uma dimensão. Os arrays que vimos até agora armazenam valores em uma dimensão, por isso para acessar às posições utilizamos somente um índice. Os arrays de 2 dimensões salvam seus valores de alguma forma como em filas e colunas e por isso, necessitaremos dois índices para acessar a cada uma de suas posições.

Com outras palavras, um array multidimensional é como um contêiner que guardará mais valores para cada posição, ou seja, como se os elementos do array fossem por sua vez outros arrays.

Em Javascript não existe um autêntico objeto array-multidimensinal. Para utilizar estas estruturas poderemos definir arrays onde, em cada uma de suas posições haverá outro array. Em nossos programas poderemos utilizar arrays de qualquer dimensão, veremos a seguir como trabalhar com de duas dimensões, que serão os mais comuns.

Neste exemplo vamos criar um array de duas dimensões onde teremos por um lado cidades e por outro a temperatura média que faz em cada uma durante os meses de inverno.

```
var temperaturas_medias_cidade0 = new Array(3)
temperaturas_medias_cidade0[0] = 12
temperaturas_medias_cidade0[1] = 10
temperaturas_medias_cidade0[2] = 11

var temperaturas_medias_cidade1 = new Array (3)
temperaturas_medias_cidade1[0] = 5
temperaturas_medias_cidade1[1] = 0
temperaturas_medias_cidade1[2] = 2

var temperaturas_medias_cidade2 = new Array (3)
temperaturas_medias_cidade2[0] = 10
temperaturas_medias_cidade2[1] = 8
temperaturas_medias_cidade2[2] = 10
```

Com as anteriores linhas criamos três arrays de 1 dimensão e três elementos, como os que já conhecíamos. Agora criaremos um novo array de três elementos e introduziremos dentro de cada um de seus campos os arrays criados anteriormente, com o qual teremos um array de arrays, ou seja, um array de 2 dimensões.

```
var temperaturas_cidades = new Array (3)
temperaturas_cidades[0] = temperaturas_medias_cidade0
temperaturas_cidades[1] = temperaturas_medias_cidade1
temperaturas_cidades[2] = temperaturas_medias_cidade2
```

Vemos que para introduzir o array inteiro fazemos referência ao mesmo sem parênteses nem colchetes, simplesmente com seu nome. O array temperaturas_cidades é nosso array bidimensinal.

Também é interessante ver como se realiza um percurso por um array de duas dimensões.

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Para iso temos que fazer um percorrido por cada um dos campos do array bidimensional e dentro destes fazer um novo percorrido para cada um de seus campos internos. Ou seja, um percorrido por um array dentro de outro.

O método para fazerum percorrido dentro de outro é colocar um loop dentro de outro, o que se chma de loop aninhado. Pode ser complicado fazer um loop aninhado, mas nós já tivemos a oportunidade de praticar em um capítulo anterior. Portanto, neste exemplo vamos colocar um loop FOR dentro de outro. Ademais, vamos escrever os resultados em uma tabela, o que complicará um pouco o script, mas poderemos ver como construir uma tabela de javascript, à medida que realizarmos o percurso aninhado ao loop.

```
document.write("<table width=200 border=1 cellpadding=1 cellspacing=1>");
for (i=0;i<temperaturas_cidades.length;i++){
   document.write("<tr>")
   document.write("<td><b>Cidade " + i + "</b></td>")
   for (j=0;j<temperaturas_cidades[i].length;j++){
     document.write("<td>" + temperaturas_cidades[i][j] + "</td>")
   }
   document.write("</tr>")
}
document.write("</table>")
```

Este script é um pouco mais complexo do que os que vimos anteriormente. A primeira ação consiste em escrever o cabeçalho da tabela, ou seja, a etiqueta <TABLE> junto com seus atributos. Com o primeiro loop realizamos um percorrido à primeira dimensão do array e utilizamos a variável i para levar a conta da posição atual. Por cada iteração deste loop escrevemos uma fila e para começar a fila abrimos a etiqueta <TR>. Ademais, escrevemos em um campo o número da cidade que estamos percorrendo nesse momento. Posteriormente, colocamos outro loop que vai percorrendo cada um dos campos do array em sua segunda dimensão e escrevemos a temperatura da cidade atual em cada um dos meses, dentro de sua etiqueta <TD>. Uma vez que acaba o segundo loop se imprimiram as três temperaturas e, portanto, a fila está terminada. O primeiro loop continua se repetindo até que todas as cidades estão impressas e uma vez terminado fechamos a tabela.

Podemos ver o exemplo em funcionamento e examinar o código do script inteiro.

Iniciação de arrays

Para terminar com o tema dos arrays vamos ver uma maneira de iniciar seus valores ao mesmo tempo que o declaramos, assim podemos realizar de uma forma mais rápida o processo de introduzir valores em cada uma das posições do array.

Vimos que o método normal de criar um array era através do objeto Array, colocando entre parênteses o número de campos do array ou não colocando nada, de modo que o array se crie sem nenhuma posição. Para introduzir valores a um array se faz igualmente, porém colocando entre parênteses os valores com os que desejamos preencher os campos separados por vírgula. Vejamos com um exemplo que cria um array com os nomes dos dias da semana.

```
var diasSemana = new
Array("Segunda","Terça","Quarta,","Quinta","Sexta","Sábado","Domingo")
```

O array se cria com 7 campos, do 0 ao 6 e em cada campo se escreve o dia da semana

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

correspondente (Entre aspas porque é um texto).

Agora vamos ver algo mais complicado, trata-se de declarar o array bidimensional que utilizamos antes para as temperaturas das cidades nos meses em uma só linha, introduzindo os valores de uma só vez.

var temperaturas_cidades = new Array(new Array (12,10,11), new Array(5,0,2),new Array(10,8,10))

No exemplo introduzimos em cada campo do array outro array que tem como valores as temperaturas de uma cidade em cada mês.

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

Pausa e conselhos Javascript

Até aqui vimos a maior parte da sintaxe e da forma de funcionar da linguagem Javascript. Agora podemos escrever scripts simples que façam uso de variáveis, funções, arrays, estruturas de controle e toda classe de operadores. Com tudo isto, conhecemos a linguagem, mas ainda temos muito chão pela frente para dominar Javascript e saber fazer todos os efeitos possíveis em páginas web, que definitivamente é o que nos interessa.

De qualquer forma, este manual foi feito com muita calma, com a intenção de que as pessoas que não estejam familiarizadas com o mundo da programação possam também ter acesso à linguagem e aprendam as técnicas básicas que permitirão afrontar futuros retos na programação. Esperamos então que o andamento deste manual tenha sido proveitoso para os mais inexperientes e que agora possam entender com facilidade as seguintes lições ou exemplos, já que conhecem as bases sobre as que estão implementados.

Antes de acabar, vamos dar uma série de conselhos a seguir na hora de programar nossos scripts que nos possam ajudar a tornar a vida mais fácil. Alguns conselhos são novos e importantes, outros foram assinalados anteriormente, mas sem dúvida não custa nada relembrar.

Distintos navegadores

A primeira coisa importante para assinalar é que Javascript é uma linguagem muito dinâmica e que foi implementada pouco a pouco, à medida que saiam novos navegadores. Se pensarmos no Netscape 2, o primeiro navegador que incluía Javascript, e o Netscape 6 ou Internet Explorer 6 daremos conta que já se passaram um mundo de novidades entre eles. Javascript mudou pelo menos tanto como os navegadores e isso, representa um problema para os programadores, porque têm que estar a par das distintas versões e a maneira de funcionar que têm.

As bases de javascript, sobre as que falamos até agora, nã mudaram praticamente nada. Somente em algumas ocasiões, que assinalamos segundo as conhecíamos -como os arrays, por exemplo - a linguagem evoluiu um pouco. Entretanto, à medida que novas tecnologias, como o DHTML, se desenvolveram, os navegadores variaram sua maneira de maneja-las.

Nosso conselho é que estejam a par das coisas que funcionam em uns e outros sistemas e que programem para que suas páginas sejam compatíveis com o maior número de navegadores. Para buscar este último é muito aconselhável provar as páginas em várias plataformas distintas. Também é muito útil deixar de lado os últimos avances, ou seja, não ir à última

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

novidade, e sim ser um pouco conservadores, para que as pessoas que atualizaram menos o browser possam também visualizar os conteúdos.

Conselhos para fazer um código simples e fácil de manter

Agora vamos dar uma série de conselhos para o código de nossos scripts seja mais claro e livre de erros. Muitos deles já assinalamos, e somos livres de aplicá-los ou não, mas se fazemos nossa tarefa como programadores pode ser muito mais agradável, na só hoje, como também o dia em que tenhamos que revisar os scripts em suas manutenções.

- Utilize comentários habitualmente para que o que estiver sendo feito nos scripts possa ser recordado por você e qualquer pessoa que tenha que lê-los mais adiante.
- Tenha cuidado com o âmbito das variáveis, lembre-se que existem variáveis globais e locais, que inclusive podem ter os mesmos nomes e conheça em cada momento a variável que tem validade.
- Escreva um código suficientemente claro, que se consegue com quebras de linhas depois de cada instrução, uma margem adequada (colocar margens a cada linha para indicar em que bloco estão incluídas), utilizar as chaves {} sempre, embora não sejam obrigatórias e em geral, manter sempre o mesmo estilo na hora de programar.
- Aplique um pouco de consistência ao manejo de variável. Declare as variáveis com var. Não mude o tipo do dado que contem (se era numérico não coloque logo texto, por exemplo). Dê nomes compreensíveis para saber em todo momento o que contém. Inclusive na hora de dar os nomes você pode aplicar uma norma, que se trata de que indiquem em seus nomes o tipo de dado que contém. Por exemplo, as variáveis de texto podem começar por uma s (de String), as variáveis numéricas podem começar por uma n ou as booleanas por uma b.
- Prove seus scripts aos poucos à medida que for programando. Você pode escrever um pedaço de código e provar antes de continuar para ver que tudo funciona corretamente. É mais fácil encontrar os erros de código em blocos pequenos do que em blocos grandes.

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

Tratamento de erros em Javascript

Para acabar a primeira parte do manual de javascript vamos explicar os erros comuns que podemos cometer e como evita-los e depura-los. Ademais veremos uma pequena conclusão da primeira parte do manual.

Erros comuns

Quando executamos os scripts podem ocorrer dois tipos de erros de sintaxe ou de execução, os vemos a seguir.

Erros de sintaxe ocorrem por escrever de maneira errônea as linhas de código, equivocar-se na hora de escrever o nome de uma estrutura, utilizar incorretamente as chaves ou os parênteses ou qualquer coisa similar. Javascript indica estes erros à medida que está carregando os scripts em memória, o que faz sempre antes de executa-loa, como foi indicado nos primeiros capítulos. Quando o analizador sintático de javascript detecta um erro destes se apresenta a

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

mensagem de erro.

Erros de execução ocorrem quando estão se executando os scripts. Por exemplo, podem ocorrer quando chamamos a uma função que não foi definida. Javascript não indica estes erros até que não se realize a chamada à função.

A maneira que têm javascript de mostrar um erro pode variar de um navegador a outro. Em versões antigas mostrava-se uma janelinha com o erro e um botão de aceitar, tanto em Internet Explorer como em Netscape. Atualmente, os erros de javascript permanecem um pouco mais ocultos ao usuário. Isto é bom, porque se nossas páginas têm algum erro em alguma plataforma não será muito incômodo para o usuário que em muitas ocasiões não perceberá. Entretanto, para o programador pode ser um pouco mais incômodo de revisar e se necessitará conhecer a maneira que se mostram os erros para que possam ser consertados.

Em versões de Internet Explorer maiores que a 4 mostra-se o erro na barra de estado do navegador e pode-se ver uma descrição maior do erro se clicamos duas vezes no ícone de alerta amarelo que aparece na barra de estado. Em Netscape aparece também uma mensagem na barra de estado que ademais nos indica que para mostrar mais informação devemos teclar "javascript:" na barra de endereços (onde escrevemos as URL para acessar às páginas). Com isso conseguimos que apareça a Consola javascript, que nos mostra todos os erros que se encontram nas páginas.

Também podemos cometer falhas na programação que façam com que os scripts não funcionem tal e como desejávamos e que javascript não detecta como erros e portanto, não mostra nenhuma mensagem de erro.

Para deixar claro, veremos um exemplo no qual nosso programa pode não funcionar como desejamos sem que javascript ofereça nenhuma mensagem de erro. Neste exemplo, trataríamos de somar duas cifras, mas se uma das variáveis é do tipo texto poderíamos encontrar com um erro.

```
var numero1 = 23
var numero2 = "42"
var soma = numero1 + numero2
```

Quanto vale a variável soma? Como muitos já sabem, a variável soma vale "2342" porque ao tentar somar uma variável numérica e outra textual, tratam-se as duas como se fossem do tipo texto e portanto, o operador + se aplica como uma concatenação de cadeias de caracteres. Se não estamos ao par desta questão poderíamos ter um erro em nosso script já que o resultado não é o esperado e ademais o tipo da variável que se soma não é numérico e sim, uma cadeia de caracteres.

Evitar erros comuns

Vamos ver agora uma lista dos erros típicos cometidos por inexperientes na programação em geral e em javascript em particular, e também, por não tão inexperientes.

Utilizar = em expressões condicionais no lugar de == é um erro difícil de detectar quando se comete, se utilizamos um só igual o que estamos fazendo é uma atribuição e não uma comparação para ver se dois valores são iguais.

Não conhecer a procedência de operadores e não utilizar parênteses para agrupar as

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

operações que se deseja realizar. Neste caso nossas operações podem dar resultados não desejados.

Usar aspas duplas e simples erroneamente. Lembre-se que podem se utilizar aspas duplas ou simples indistintamente, com a seguinte norma: dentro de aspas duplas devem se utilizar aspas simples e vice-versa.

Esquecer de fechar chave ou fechar uma chave a mais.

Colocar várias sentenças na mesma linha sem separá-las de ponto e vírgula. Isto costuma acontecer nos manipuladores de eventos, como onclick, que veremos mais adiante.

Utilizar uma variável antes de iniciá-la ou não declarar as variáveis com var antes de utiliza-las também são erros comuns. Lembre-se que se não a declaras poderá estar fazendo referência a uma variável global no lugar de uma local.

Contar as posições dos arrays a partir de 1. Lembre-se que os arrays começam pela posição 0.

Depurar erros javascript

Qualquer programa é suscetível de conter erros. Javascript nos informará de muitos dos erros da página: os que têm relação com a sintaxe e os que têm lugar no momento da execução dos scripts a causa de equivocarmos ao escrever o nome de uma função ou de uma variável. Entretanto, não são os únicos erros que podemos encontrar, também estão os erros que ocorrem sem que javascript mostre a correspondente mensagem de erro, como vimos anteriormente, mas que fazem com que os programas não funcionem como esperávamos.

Para todo tipo de erro, uns mais fáceis de detectar que outros, devemos utilizar alguma técnica de depuração que nos ajude a encontra-los. Linguagens de programação mais potentes que a que tratamos agora incluem importantes ferramentas de depuração, porém em javascript devemos nos contentar com uma rudimentar técnica. Trata-se de utilizar uma função pré-definida, a função alert() que recebe entre parênteses um texto e o mostra em uma pequena janela que tem um botão de aceitar.

Com a função alert() podemos mostrar em qualquer momento o conteúdo das variáveis que estamos utilizando em nossos scripts. Para isso colocamos entre parênteses a variável que desejamos ver seu conteúdo. Quando se mostra o conteúdo da variável o navegador espera que apertemos o botão de aceitar e quando o fazemos continua com as seguintes instruções do script.

Este é um simples exemplo sobre como se pode utilizar a função alert() para mostrar o conteúdo das variáveis.

```
var n_atual = 0
var soma = 0
while (soma<300){
   n_atual ++
   soma += soma + n_atual
   alert("n_atual vale " + n_atual + " e soma vale " + soma)
}
```


© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.

Com a função alert() se mostra o conteúdo das duas variáveis que utilizamos no script. Algo similar a isto é o que teremos que fazer para mostrar o conteúdo das variáveis e saber como estão funcionando nossos scripts, se vai tudo bem ou se há algum erro.

Conclusão

Até aqui conhecemos a sintaxe javascript em profundidade. Embora ainda faltam coisas importantes de sintaxe, a visão que se pode ter da linguagem será suficiente para enfrentar os problemas mais fundamentais. Mais adiante apresentaremos outras reportagens para aprender a utilizar os recursos com os quais contamos na hora de fazer efeitos em páginas web.

Veremos a hierarquia de objetos do navegador, como construir nossos próprios objetos, as funções pré-definidas de javascript, características do HTML Dinâmico, trabalho com formulários e outras coisas importantes para dominar todas as possibilidades da linguagem.

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

© Os manuais de CriarWeb.com têm copyright dos autores. Não reproduzir sem autorização.