Objeto Array

Em geral

O objeto **Array** é uma lista de elementos indexados nos quais pode-se guardar (escrever) dados ou as invocar (ler).

Atenção! O objeto Array é do Javascript 1.1

Array de uma dimensão

Para fazer um array, procede-se a duas etapas:

- primeiro construir a estrutura do array. Nesta fase, os elementos do array estão vazios.
- depois inserir valores nos elementos definidos.

Começa-se por definir o array :

nome_do_array = new Array (x);

Onde **x** é o número de elementos do array.

De notar que, o número de elementos é limitado a 255.

Depois, depois alimenta-se a estrutura definida :

nome_do_array [i] = "elementos";

onde **i** é um número compreendido entre 0 e **x** menos 1.

Exemplo: um caderno de endereço com 3 pessoas

construção do array :
caderno = new Array(3);

inserir dados:
caderno[0]="Sérgio";
caderno[1]="Paulo";
caderno[2]="Angelo";

para acessar a um elemento, emprega-se:
document.write(caderno[2]);

Nota-se também que os dados são bem visíveis ao leitor (view source).

Nota:

- Muitas vezes é prático carregar o array com um ciclo. Admitindo que temos que carregar 50 imagens. Ou as carregamos manualmente (0.gif, 1.gif, 2.gif...), ou utiliza-se um ciclo do estilo:

function gifs() {
gif = new Array(50);

```
for (var=i;i<50;i++)
{gif[i] =i+".gif";}
}
```

Propriedades e Métodos

ELEMENTOS	DESCRIÇÃO
length	Devolva o número de elementos do array.
join()	Junta todas os elementos do array numa única cadeia. Os diferentes elementos são separados por um caractere separador especificado no argumento. Por defeito, este separador é uma vírgula.
reverse()	Inversa a ordem dos elementos.
sort()	Devolva os elementos por ordem alfabético.

Usando os exemplo do quadro,

document.write(carnet.join()); dá como resultado : Sérgio,Paulo,Ângelo.
document.write(carnet.join("-")); dá como resultado : Sérgio-Paulo-Ângelo.
document.write(carnet.reverse().join("-")); dá como resultado : Ângelo-Paulo-Sérgio

Array de duas dimensões

Pode-se criar arrays de duas dimensões (e mais ainda).

Primeiro declara-se array de 1 dimenão da maneira clássica :

```
nome_do_array = new Array (x);
```

Depois, declara-se cada elemento do array como um array de 1 dimension :

```
nome_do_array[i] = new Array(y);
```

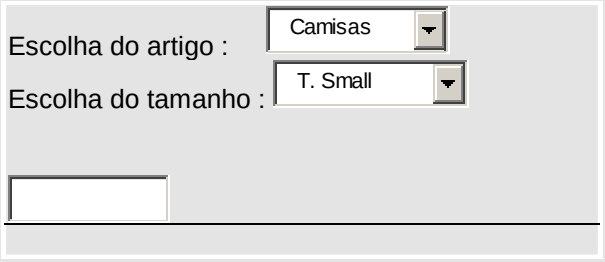
Para um array de 3 por 3 :

PREÇOS	T. SMALL	T. MEDIUM	T. LARGE
CAMISAS	1200	1250	1300
POLOS	800	850	900
T-SHIRT	500	520	540

```
nome = new Array(3);
nome[0] = new Array(3);
nome[1] = new Array(3);
nome[2] = new Array(3);
nome[0][0]="1200"; nome[0][1]="1250"; nome[0][2]="1300";
nome[1][0]="800"; nome[1][1]="850"; nome[1][2]="900";
```

nome[2][0]="500"; nome[2][1]="520"; nome[2][2]

Para explorar estes dados, aqui tem uma ilustração do que é possível:



O formulário escreva-se:

```
<FORM name="form" >
<SELECT NAME="liste">
<OPTION>Camisas
<OPTION>Polos
<OPTION>T-shirts
</SELECT>
<SELECT NAME="tamanho">
<OPTION>T. Small
<OPTION>T. Medium
<OPTION>T. Large
</SELECT>
<INPUT TYPE="button" VALUE="Obter preço " onClick="affi(this.form)">
<INPUT TYPE="TEXT" NAME="txt">
</FORM>
```

Onde a função **affi()** formula-se assim:

```
function affi() {
i = document.form.liste.selectedIndex;
j= document.form.taille.selectedIndex
document.form.txt.value=nome[i][j];
}
```

Base de Dados

Aqui está um título bastante pesado! Si o Javascript pode guardar dados e apresentar estas segundos os gostos do cliente, mas estamos mesmo assim muito longe das ferramentas específicas para este tipo de trabalho.
As bases de dados em Javascript serão sempre do tipo **estático** e sobretudo com uma codificação trabalhosa.

Até agora, definiu-se alguns caracteres nos elementos dos array. Nas strings, pode-se inserir muitos e sobretudo com as ferramentas de manipulação das strings do Javascript, pode-se guardar coisas do tipo:
Sérgio*Brandão*rua da Passagem,444*4440*Valongo*Portugal*
Ou seja neste caso, 6 dados.

Codificação tipo fixo

Retomando o exmplo do carderno de endereço. De maneira clássica, deve-se prever os difrrentes

campos e o número de posições consacradas a cada campo. Por exemplo:

Nome	20 posições
Apelido	10 posições
Endereço	20 posições
Código postal	10 posições
Cidade	10 posições
País	10 posições

Cada campo deve respeitar o número de posições predeterminados. As posições sobrenumerários devem ser completadas em branco. Com um risco de erro elevado a codificação e um tempo de carregamento da página sensivelmente aumentada para finalment transmetir que espaços vazios.

Nome 20 p.	**Apelido** 10 p.	**Endereço** 20 p.	**Postal** 10 p.	**Cidade** 10 p.	**País** 10 p.
Sérgio	Brandão	Rua da Passagem,444	4440	Valongo	Portugal

Para retomar os diferentes dados, utiliza-se a instrução **substring(x,y)**. Assim:

Nome	substring(0,19)
Apelido	substring(20,29)
Endereço	substring(30,49)
Código postal	substring(50,59)
Cidade	substring(60,69)
País	substring(70,79)

Codificação tipo variável

O princípio é de inserir um separador entre cada campo. E "pede-se" ao Javascript de retirar os caracteres compreendidos entre dois separadores.

Com o separador *****, os dados tornam-se:
str="Sérgio*Brandão*Rua da passagem,444*4440*Valongo*Portugal*"
ou seja 60 posições em vês de 80 tem-se um ganho de 25 %.

Para ler este diferentes dados, proceda-se em varias etapas:

- Para **pos=str.indexOf("*")**, nota-se a posição na string do primeiro separador encontrado.
- Para **str.substring(0,pos)**, tem-se o primeiro dado compreendido entre o início da string (posição 0) e a posição menos 1 do separador ou seja Sérgio.
- Para **str=str.substring(pos+1,str.length)**, recria-se uma string correspondendo à string do iníncio menos o dado parcial que acabou-se de extrair menos um separador. O que dá: **str="Brandão*Rua da passagem,444*4440*Valongo*Portugal*"**
- E isto, por um ciclo, até haver separadores menos 1.

Para guardar os dados, a maneira mais prática para os guardar é o controlador do formulário Hidden (encondido)em vez da string.

Pourquê? Bem com as string funciona, mas há o risco de chegar a limitação do comprimento das strings em Javascript, 50 à 80 caracteres. Por isso usa-se o símbolo + e a concatenação dos dados para que ele seja aceito pelo compilador Javascript.

O controlador do formulário Hidden não está sujeito a esta limitação de caracteres. Pode-se guardar 30 K (e mais) de dados. Basta assim entrar no controlador do formulário Hidden e dar atributo value="os dados a guardar".

O script completo será algo do tipo:

```
<HTML>
<BODY>
<FORM name="form">
<INPUT type=hidden name="data" value="Sérgio*Brandão*Rua da
passagem,444*4440*Valongo*Portugal*">
</FORM>
<SCRIPT LANGUAGE="javascript">
str=document.form.data.value; // extriar o string do controlador hidden
nsep=6 // número do separador
for (var i=0;i<nsep;i++){
pos=str.indexOf("*");
document.write(str.substring(0,pos)+ "<BR>");
str=str.substring(pos+1,str.length);
}
</SCRIPT>
</BODY>
</HTML>
```