Objeto String

Em geral

Voltemos ao objeto String para nos integrar da manipulação dos caracteres tão úteis para o aspecto da programação do Javascript.

INSTRUÇÃO	DESCRIÇÃO
length	É um inteiro que indica o comprimento da cadeia de caracteres.
charAt()	Método que permite acessar a um caractere isolado de uma cadeia.
indexOf()	Método que reenvia a posição de uma cadeia parcial a partir de uma posição determinada. (começando do início da cadeia principal seja na posição 0).
lastIndexOf()	Método que reenvia a posição de uma cadeia parcial a partir de uma posição determinada. (começando do FIM da cadeia principal seja na posição 1).
substring(x,y)	Método que reenvia uma String parcial situada entre a posição x e a posição y-1.
toLowerCase()	Transforma todas as letras em minúsculas.
toUpperCase()	Transforma todas as letras em Maiúsculas.

A propriedade length

A propriedade length devolve um inteiro que indica o número de elementos numa cadeia de caracteres. Se a cadeia está vazia (" "), o número é zero.

A sintaxe é simples :

```
x=variable.length;
x=("cadeia de caracteres").length;
```

A propriedade length só serve para as Strings, mas também para conhecer o comprimento e número de elementos :

- de formulários. Quantos formulários diferentes existem?
- de botões radio. Quantos botões radio existem num grupo?
- de checkbox. Quantas checkbox existem num grupo?
- de opções. Quantas opções existem num Select?
- de frames. Quantas frames existem?
- de âncoras, de links, etc

O método charAt()

Notamos que os caracteres são contados da esquerda para a direita e que a posição do primeiro caractere é 0. A posição do último caractere é, então, o comprimento (length) da cadeia de caractere menos 1;

cadeia :	`Javascript` (comprimento = 10)

posição :	|||||||||| 0123456789 (comprimento - 1)

Se a posição que se indica é inferior à zero ou maior que o comprimentos menos 1, Javascript devolve uma cadeia vazia.

A sintaxe de charAt() é:

```
cadeia_reposta = cadeia_partida.charAt(x);
```

Onde x é um inteiro compreendido entre 0 e o comprimento da cadeia a analisar menos 1.

Notar os seguintes exemplos:

var str="Javascript";
var chr=str.charAt(0);
var chr="Javascript".charAt(0);
ou var chr=charAt(str,0);
ou var chr=charAt("Javascript",0);
	A resposta é "J".

var str="Javascript";
var chr=str.charAt(9);
var chr=charAt(str,9);
	A resposta é "t".

var str="Javascript"; var chr=charAt(str,13);	A resposta é "" seja vazia.

O método indexOf

Este método reenvia à posição, seja x, de uma String parcial (letra única, grupo de letras ou palavra) numa cadeia de caracteres começando na posição indicado por y. Isto permite, por exemplo, de se ver se uma letra, um grupo de letras ou uma palavra existem numa frase.

```
variavel="cadeia_de_caracteres";
var="string_parcial";
x=variavel.indexOf(var,y);
```

Onde **y** é a posição à partir da qual a pesquisa (da esquerda para a direita) deve começar. Este pode ser qualquer inteiro compreendido entre 0 e o comprimento -1 da cadeia à analisar.
Se a posição não é especificada, a pesquisa começa por padrão da posição 0.

Se a string parcial não é encontrada na cadeia de caracteres analisada, o valor devolvido será igual a -1. exemplos :

variavel="Javascript" var="script" x=variable.indexOf(var,0);	x vale 4
variavel="www.bitplane.com.br" var="@" x=variable.indexOf(var);	x vale -1

O método lastIndexOf()

Este método é muito parecido com o indexOf() só que a pesquisa é feita da direita para a esquerda (começa pelo fim).

A sintaxe é idêntica só que o **y** representa uma posição em relação ao fim da cadeia de caracteres.

```
x=variavel.lastIndexOf(var,y);
```

Os seguintes exemplos mostrem a diferença entre index0f() e lastIndexOf() :

variável="Javascript"
var="a"
x=variavel.indexOf(var,0); aqui **x** vale **1** ou seja a posição do primeiro **a**.
x=variavel.lastIndexOf(var,9); ici **x** vale 3 ou seja a posição do segundo **a**.

Note que mesmo quando começa-se a ler a partir do fim da cadeia, a posição devolvida é contada desde o início da cadeia começando por zero.

O método substring()

O método **substring()** é do tipo indexOf(), lastIndexOf() e charAt() que acabamos de estudar. Este método será particularmente útil, por exemplo, para tomar diferentes dados numa longa cadeia de caracteres.

```
variavel = "cadeia de caracteres"
resultado=variavel.substring(x,y)
```

Os **x** e **y** são inteiros compreendidos entre 0 e o comprimento menos 1 da cadeia de caracteres.
Se **x** é inferior ao **y**, o valor devolvido começa na posição **x** e acaba na posição **Y-1**.
Se **x** é superior ao **y**, o valor devolvido começa na posição **y** e acaba na posição **X-1**.
Isso, dá o mesmo resultado e é equivalente escrever por exemplo substring(3,6) ou substring(6,3).

Se **x** é igual ao **y**, substring() devolva uma cadeia vazia (lógico, não?)

Aqui estão alguns exemplos :

Javascript
| | | | | | | | | |
0123456789
str="Javascript";

str1=str.substring(0,4);
str2="Javascript".substring(0,4);
str3=str.substring(6,9);

Os resultados são :

str1="Java"; seja as posições 0,1,2 e 3.
str2="Java"; seja as posições 0,1,2 e 3.
str3="rip"; seja as posições 6,7 e 8

O método toLowerCase()

Este método rescreve uma cadeia toda em caracteres minúsculos.

```
variavel2="cadeia de caracteres";
variavel1=variavel2.toLowerCase();
```

Exemplo :

```
str="JavaScript";
str1=str.toLowerCase();
str2="JavaScript".toLowerCase();
```

O resultado será :

```
str1="javascript";
str2="javascript";
```

O método toUpperCase()

Este método rescreve uma cadeia toda em minúsculas.

```
variavel2="cadeia de caracteres";
variavel1=variavel2.toUpperCase();
```

Exemplo :

```
str="JavaScript";
str1=str.toUpperCase();
str2="JavaScript".toUpperCase();
```

O resultado será :

```
str1="JAVASCRIPT";
str2="JAVASCRIPT";
```

Utilidade do toLowerCase() e do toUpperCase()

A utilidade destes 2 métodos não salta a vista. Mas é importante, visto que o Javascript é **case sensitive**. Assim uma pesquisa sobre Euro irá dar o mesmo resultado do que EURo.

Pode-se assim aconselhar de converter as bases de dado em minúsculas (ou toda em maiúscula).