JavaScript - Utilizando Objetos String

Toda **string** no JavaScript é armazenada como um objeto String, normalmente não precisamos nos preocupar com isso, mas sim aproveitar os recursos que o objeto string nos proporciona.
Existem duas maneiras de se criar um objeto String. A primeira nós já a utilizamos nos módulos anteriores, enquanto a segunda, utiliza a sintaxe oficial de objetos:

```
Teste = "Isto é apenas um teste";
Teste = new String("Isto é apenas um teste");
```

As duas instruções acima são semelhantes, embora a segunda seja a maneira original de se declarar um novo objeto.

Calculando o Comprimento da String

Em algumas situações, veremos que irá ser muito útil sabermos quantos caracteres uma variável de string armazena. Podemos obter facilmente esta informação, utilizando a propriedade length dos objetos String.
Por exemplo, Teste.length refere-se ao comprimento da string Teste.
Eis um exemplo dessa propriedade:

```
Teste = "Isto é um teste";
document.write(Teste.length);
```

Convertendo Maiúsculas/Minúsculas

Dois métodos do objeto String permitem converter todo o conteúdo de uma string em letras maiúsculas ou em letras minúsculas:

- toUpperCase() Converte todos os caracteres na string em letras maiúsculas;
- toLowerCase() Converte todos os caracteres na string em letras minúsculas.

Por exemplo, a seguinte instrução exibe o valor da variável de string Teste em letras minúsculas:

```
document.write(Teste.toLowerCase());
```

Note que essa instrução NÃO altera o valor da variável Teste. Esses métodos retornam a versão de letras maiúsculas ou minúsculas da string, mas eles não alteram a própria string. Se quisermos alterar o valor da string, podemos fazê-lo da seguinte maneira:

```
Teste = Teste.toLowerCase();
```

Trabalhando com Substrings

Trabalhamos até aqui, utilizando apenas as strings inteiras, mas na maioria das vezes também nos é necessário saber o conteúdo de partes separadas de uma string, para isso o JavaScript nos permite manusear partes de uma string através do método substring().

O método substring() retorna uma string consistindo em uma parte da string original entre dois valores de índice, que devemos especificar entre parênteses. Por exemplo, a seguinte instrução exibe o quarto, o quinto e o sexto caracteres da string Teste:

```
document.write(Teste.substring(3,6);
```

A esse ponto, você provavelmente está se perguntando de onde vêm o 3 e o 6. Há três coisas que você precisa entender sobre os parâmetros de índice:
- A indexação inicia com 0 para o primeiro caractere da string, então o quarto caractere é realmente o 3.

- O segundo índice é não inclusivo. Um segundo índice de 6 inclui até o índice 5 (o sexto caractere).
- Podemos especificar os índices em qualquer forma, mas o JavaScript assumirá sempre que o índice menor é o primeiro índice.

Como outro exemplo, suponhamos que temos uma string chamada Alfabeto:

```
Alfabeto = "ABCDEFGHIJKLMNOPQRTUVWXYZ";
```

Os seguintes são exemplos do método substring() utilizando essa string:

· Alfabeto.substring(0,4) retorna ABCD;
· Alfabeto.substring(10,12) retorna KL;
· Alfabeto.substring(12,10) também retorna KL. Porque é menor 10, é utilizado como o primeiro índice;
· Alfabeto.substring(6,7) retorna G;
· Alfabeto.substring(24,26) retorna YZ;
· Alfabeto.substring(0,26) retorna o alfabeto inteiro;
· Alfabeto.substring(6,6) retorna o valor nulo, uma string vazia. Isso é verdadeiro sempre que os dois valores de índice são os mesmos.

Outro método interessante é o **charAt**, que é uma maneira fácil de se pegar um único caractere de uma string. Devemos apenas especificar o índice do caractere entre parênteses, lembrando que o primeiro caractere corresponde ao índice 0:

· Alfabeto.charAt(0) retorna A;
· Alfabeto.charAt(12) retorna M;
· Alfabeto.charAt(25) retorna Z;
· Alfabeto.charAt(27) retorna uma string vazia porque não há nenhum Alfabeto nessa posição.

Quanto à localização de substrings dentro de uma string, o objeto String ainda possui os seguintes métodos:
IndexOf - utilizado para localizar uma string dentro de outra string:

```
Resultado = Alfabeto.indexOf("JKL");
```

O valor retornado na variável Resultado é um índice na string, semelhante ao primeiro índice no método substring. O primeiro caractere da string tem índice 0. Podemos ainda especificar um índice opcional, a fim de indicarmos um ponto de partida dentro da string para que seja iniciada a busca. Por exemplo, a instrução abaixo procura a palavra "VACA" na string Alfabeto, iniciando com o 11o. caractere:

```
Resultado = Alfabeto.indexOf("VACA", 10);
```

Um outro método, o **lastIndexOf()**, trabalha da mesma maneira, só que ele retorna a última ocorrência da string. Ele pesquisa a string para trás, iniciando com o último caractere:

```
Resultado = alfabeto.LastIndexOf("VACA", 10);
```

Arrays Numéricos

Um **array**, ao contrário de outros tipos de dados no JavaScript, deve obrigatoriamente ser declarado antes de ser utilizado. A instrução abaixo cria um array com 10 elementos:

```
Notas = new Array(10);
```

Para atribuírmos valores a este array, utilizamos parênteses e um índice:

```
Notas[0] = 7;
Notas[1] = 7;
Notas[2] = 3;
Notas[3] = 8;
Notas[4] = 9;
Notas[5] = 5;
Notas[6] = 6;
Notas[7] = 6;
Notas[8] = 4;
Notas[9] = 10;
```

Assim como as strings, os arrays têm a propriedade length, que informa o número de elementos no array, normalmente o mesmo número que utilizamos ao criar o array. Podemos ainda acessar o valor de array, referenciando apenas o nome e o seu índice:

```
MostraNotas = "Notas obtidas: " + Notas[0] + "," + Notas[1] + "," + Notas[2];
document.write(MostraNotas);
```

Arrays de String

O método de criação e utilização de um array de strings é bem semelhante ao de um array numérico:

```
Nomes = new Array(3); //aqui o array foi criado
Nomes[0] = "José"; //daqui em diante, foi atribuído valores
Nomes[1] = "Maria";
Nomes[2] = "João";
```

Podemos ainda usar os métodos de string apresentados mais acima também em um array de strings.

Dividindo uma String

O JavaScript ainda possui um método chamado split, que divide uma string em suas partes componentes. Para utilizar esse método, especifique a string a dividir juntamente com um caractere para dividir as partes:

```
Nome = "José Francisco Silva";
Partes = Nome.split(" ");
```

A instrução acima, cria um novo array (Partes), contendo os pedaços da string Nome, como segue abaixo:

```
Partes[0] = "José";
Partes[1] = "Francisco";
Partes[2] = "Silva";
```

Podemos ainda fazer a operação contrária, ou seja, juntar partes de um array, utilizando o método join:

```
NomeCompleto = Partes.join(" ");
```

Script: Rolando uma mensagem na Barra de Status do Navegador

Agora que já dominamos perfeitamente o mundo dos arrays em JavaScript, está na hora de começar a colocar tudo isso em prática. Com certeza você enquanto navega pela internet, já deve ter se deparado em algum site, em que uma mensagem fica rolando na barra de status do navegador, pois bem, chegou o momento de fazermos a nossa própria mensagem.
Vamos começar atribuindo a mensagem a ser rolada em uma variável:

```
Mensagem = "Ibest.Masters - O melhor caminho para um Site de Sucesso!";
```

Em seguida, criamos também uma outra string, que será usada como espécie de um separador da mensagem:

```
Espaços = ">>>   <<<";
```

Precisaremos também de uma outra variável numérica que irá armazenar a posição atual da string, vamos chamá-la de Posição e seu valor inicial deverá ser 0:

```
Posicao = 0;
```

Quem irá fazer o trabalho de rolagem na barra de status, será uma função que iremos criar, chamada de RolaMensagem:

```
function RolaMensagem()
{
   window.status = Mensagem.substring(Posicao, Mensagem.length) + Espaços +
Mensagem.substring(0, Posicao);
   Posicao ++;

  if (Posicao > Mensagem.length) Posicao = 0;

  window.setTimeout("RolaMensagem()", 200);
}
```

Resumindo a função RolaMensagem, ela associa o resultado da Mensagem e dos Espaços mostrando na barra de status do navegador. Devemos ficar atentos que toda vez, esse resultado será diferente, já que a variável Posicao vai sendo incrementada. E por fim a instrução **setTimeout** que permite configurar uma instrução a ser executada depois de uma demora de tempo. O valor é descrito em milésimos de segundos.
A seguir, o código completo da função RolaMensagem já incluso em um documento de HTML:

```
<HTML>
<HEAD><TITLE>Rolando uma Mensagem</TITLE>
<script language="JavaScript">
<!--
Mensagem = "Ibest.Masters - O melhor caminho para um Site de Sucesso!";
Espacos = "..... .....";
Posicao = 0;

function RolaMensagem()
{
   window.status = Mensagem.substring(Posicao, Mensagem.length)
+ Espacos + Mensagem.substring(0, Posicao);     Posicao ++;

  if (Posicao > Mensagem.length) Posicao = 0;

   window.setTimeout("RolaMensagem()", 200);
}
RolaMensagem();
//-->
</script>
</HEAD>

<BODY BGCOLOR="#FFFFFF">
<H1>Exemplo de Rolagem de Mensagens na barra de Status do Navegador</H1>
</BODY>
</HTML>;
```