JavaScript - Objetos predefinidos - String

Es un objeto predefinido de JavaScript para manejo de cadenas de caracteres.

Propiedades del objeto String

`length`	Longitud de la cadena de caracteres asociada al objeto

Se crea con el constructor String, pero cualquier variable que contiene una cadena de texto puede ser tratada como String (JavaScript se encarga de hacer las conversiones necesarias):

```
cadena1 = new String ("Un String");
longitud = cadena1.length;
cadena2 = "Una cadena de caracteres";
longitud = cadena2.length;
// también se puede poner
longitud="Una cadena de caracteres".length
```

Métodos del objeto String

`charAt(posición)`	Devuelve el caracter correspondiente a la posición indicada (teniendo en cuenta que las cadenas comienzan con índice cero). var cualquierCadena="Brave new world"; "El carácter en el índice 0 es '" + cualquierCadena.charAt(0) + "'" "El carácter en el índice 3 es '" + cualquierCadena.charAt(3) + "'" "El carácter en el índice 999 es '" + cualquierCadena.charAt(999) + "'" Resultado: El carácter en el índice 0 es 'B' El carácter en el índice 3 es 'v' El carácter en el índice 999 es ''

`charCodeAt(caracter)`	Devuelve el el valor Unicode correspondiente al carácter situado en la posición indicada (las cadenas comienzan con índice cero).
	"ABC".charCodeAt(0) // returns 65
	"AaSdas".charCodeAt(2) // returns 83
	teniendo en cuenta que 2 es la posicion de la letra. Si `S` fuera minuscula, el Unicode es diferente
	"Aasdas".charCodeAt(2) // returns 115

`concat(cadena1,...,cadenan)`	Concatena dos o más cadenas y devuelve como resultado esa unión. cadena1="Oh "

cadena2="qué maravillosa "
cadena3="mañana'."
cadena4=cadena1.concat(cadena2,cadena3) // devuelve "Oh qué maravillosa mañana'."

`endsWith(cadenaDeBusqueda[, posicion])`

*

Este método determina cuando una cadena termina en otra cadena.
cadenaDeBusqueda Los caracteres a buscar hasta el final de la cadena.
posicion
Opcional. Indicada la longitud de la cadena; por defecto, su valor no puede ser superior al de la cadena en la que desea buscar.
Valor retornado
true si la cadena de texto contiene la cadena buscada; en caso contrario, *false*.

var str = 'To be, or not to be, that is the question.';
str.endsWith('question.'); // true
str.endsWith('to be'); // false
str.endsWith('to be', 19); // true

`fromCharCode(cod1,...,codn)`

Convierte valores Unicode en caracteres.
Este método devuelve una cadena y no un objeto String.
Debido a que fromCharCode es un método estático de String, siempre se usará como String.fromCharCode.

String.fromCharCode(65,66,67)
devuelve la cadena "ABC".

`includes(`*`cadenaDeBusqueda`* `[, `*`posicion`*`])`

*

Este método permite determinar si una cadena de texto se encuentra incluida dentro de la otra.
cadenaDeBusqueda Una cadena a buscar en el texto
posicion
Opcional. La posición dentro de la cadena en la cual empieza la búsqueda del searchString. Por defecto este valor es 0.
Valor retornado
true si la cadena de texto contiene la cadena buscada; en caso contrario, *false*.
El método includes() es "case sensitive" (tiene en cuenta mayúsculas y minúsculas)
var str = 'To be, or not to be, that is the question.';
str.includes('To be'); // true
str.includes('question'); // true
str.includes('nonexistent'); // false
str.includes('To be', 1); // false
str.includes('TO BE'); // false

`indexOf(valor[,inicio])`

Devuelve el índice de la primera aparición del valor

especificado dentro de la cadena desde la posición inicio, si se omite inicio empieza a buscar desde el primer carácter empezando desde la izquierda de la cadena. Si no se encuentra devuelve -1:

```
cadena = "uno dos tres cuatro";
posicionDeDos = cadena.indexOf ("dos");
// posicionDeDos = 4
```

`lastIndexOf(valor[,fin])` Devuelve el índice de la primera aparición del valor especificado dentro de la cadena desde la posición fin, si se omite inicio empieza a buscar desde el primer carácter empezando desde la derecha de la cadena. Si no se encuentra devuelve -1:

```
cadena = "uno dos tres cuatro dos";
posicionDeDos = cadena.lastIndexOf ("dos");
// posicionDeDos = 20
```

```
var anyString="Brave new world"

// Displays 8
document.write("<P>The index of the first w from the beginning is " +
   anyString.indexOf("w"))
// Displays 10
document.write("<P>The index of the first w from the end is " +
   anyString.lastIndexOf("w"))
// Displays 6
document.write("<P>The index of 'new' from the beginning is " +
   anyString.indexOf("new"))
// Displays 6
document.write("<P>The index of 'new' from the end is " +
   anyString.lastIndexOf("new")
```

`match(patrón)` Busca una coincidencia en una cadena y devuelve todas las coincidencias encontradas en un array.

`replace(patrón, nuevaCadena)` Busca una coincidencia en una cadena y si existe, la remplaza por otra cadena pasada como parámetro.
flags: g - global, i - ignorar mayúsculas.
Ejemplo 1:
var re = /apples/gi;
var str = "Apples are round, and apples are juicy.";
var newstr = str.replace(re, "oranges");
El primer ejemplo devuelve "oranges are round, and

oranges are juicy."
Ejemplo 2:
var str = "Apples are round, and apples are juicy.";
var newstr = str.replace("Apples", "oranges");
El Segundo ejemplo devuelve "oranges are round, and apples are juicy."

`search(patrón)`

Busca una coincidencia en una cadena y devuelve la posición de la coincidencia o -1 si no existe ninguna.

`slice(inicio, ultimo)`

Extrae un trozo de una cadena desde inicio y último sin incluirlo, ambos de índice cero, o el final de la cadena si se omite este segundo:

```
cadena = "dos trozos";
trozo = cadena.slice (4,9); // trozo='trozo'
```

Si el segundo carácter es negativo, indica la posición comenzando por la parte derecha de la cadena original:

```
trozo = cadena.slice (4,-1); // trozo='trozo'
```

El siguiente ejemplo usa `slice()` para crear una nueva cadena.

```
var cadena1 = "La mañana se nos echa encima.";
var cadena2 = cadena1.slice(3, -2);
```

Esto escribe:

```
mañana se nos echa encim
```

`split(separador[,limite])`

Divide una cadena en trozos y los devuelve en un array. Se pasa como parámetro la cadena utilizada como separador:

```
cadena = "uno dos tres cuatro dos";
trozos = cadena.split (" "); // el separador es el
espacio
document.write (trozos[0]); // trozos[0] = 'uno'
```

Se puede especificar un segundo parámetro que indica el número máximo de trozos:

```
trozos = cadena.split (" ",2);
// se crea un array de 2 elementos
```

`startsWith(cadenaDeBusqueda[, posicion])`

*

Este método indica si una cadena comienza con los caracteres de otra cadena,
devolviendo true o false según exista o no.
cadenaDeBusqueda Los caracteres que buscamos al inicio de la cadena.
posicion Opcional. La posición de la cadena en la cual debe comenzar la búsqueda
de *cadenaDeBusqueda*; el valor predeterminado es 0.

```
var str = 'Ser, o no ser. ¡Esa es la cuestión!';
str.startsWith('Ser');          // true
str.startsWith('no ser');       // false
str.startsWith('Esa es la', 16); // true
```

`substr(inicio[,cantidad])`

Devuelve una subcadena a partir de los parámetros 'inicio' y 'longitud':

```
cadena = "uno dos tres cuatro dos";
subcadena = cadena.substr (4, 3); // subcadena = 'dos'
```

Si no se especifica la longitud, se considera hasta el final de la cadena. Si 'inicio' es un valor negativo se cuenta la posición inicial a partir del final de la cadena.

```
var cadena = "abcdefghij";

"(1,2): "    + cadena.substr(1,2); // '(1, 2): bc'
"(-3,2): "   + cadena.substr(-3,2);  // '(-3, 2): hi'
"(-3): "     + cadena.substr(-3);    // '(-3): hij'
"(1): "      + cadena.substr(1);     // '(1): bcdefghij'
"(-20, 2): " + cadena.substr(-20,2)); // '(-20, 2): ab'
"(20, 2): "  + cadena.substr(20,2);  // '(20, 2): '
```

Este script muestra:

```
(1,2): bc
(-3,2): hi
(-3): hij
(1): bcdefghij
(-20, 2): ab
(20, 2):
```

`substring(indiceA[,indiceB])`

Devuelve una subcadena que está comprendida entre dos índices de la cadena actual, sin incluir el carácter del indiceB, ambos de índice cero:

```
cadena = "uno dos tres cuatro dos";
subcadena = cadena.substring (4, 7); // subcadena =
'dos'
```

Si no se especifica indiceB, se considera hasta el final de la cadena.

Si indiceA es mayor que indiceB, entonces el efecto de substring es como si los dos argumentos se intercambiasen; por ejemplo, *cadena*.substring(1, 0) == *cadena*.substring(0, 1).

```
// asume una función print ya definida
var cualquierCadena = "Mozilla";

// Muestra "Moz"
print(cualquierCadena.substring(0,3));
print(cualquierCadena.substring(3,0));

// Muestra "lla"
print(cualquierCadena.substring(4,7));
print(cualquierCadena.substring(7,4));

// Muestra "Mozilla"
print(cualquierCadena.substring(0,7));
print(cualquierCadena.substring(0,10))
```

`toLowerCase()`

Devuelve una cadena con los caracteres convertidos a minúsculas.

`toUpperCase()`

Devuelve una cadena con los caracteres convertidos a mayúsculas.

`trim()`
`trimLeft()`
`trimRight()`

Eliminan los espacios al principio y al final de la cadena, al principio solamente o al final solamente, respectivamente.

** Los métodos pueden no funcionar correctamente en todos los navegadores. Habrá que comprobar la compatibilidad.*

Métodos desaconsejados

anchor(nombreDelAncla) crea un anclaje HTML:

```
referencia = new String ("Zona inferior");
document.write (referencia.anchor ("inferior"));
```

Sería equivalente a:
 Zona inferior

big() Devuelve una cadena que representa el texto en formato BIG de HTML.

blink() Crea el efecto de una cadena parpadeante.

bold() Devuelve una cadena que representa el texto en negrita (...) en HTML.

fixed() Devuelve una cadena que representa el texto en formato TT con fuente monoespaciada de HTML.

fontcolor(color) Modifica el color de la fuente de la cadena con el color pasado como parametro.

fontsize(número) Modifica el tamaño de la fuente de una cadena con el valor pasado como parametro.

italics() Devuelve una cadena que representa el texto en formato <I>...</I> (cursiva) de HTML.

link(href) Construye una cadena que se corresponde con la marca de enlace (<a>) de HTML:

```
enlace = new String ("Epsilon Eridani");
marcaEnlace = enlace.link ("http://www.epsilon-
eridani.com");
// marcaEnlace = <a href='http://...'>Epsilon Eridani</a>
```

small() Devuelve una cadena que se corresponde con el formato SMALL de HTML.

strike() Muestra una cadena techada.

sub() Devuelve una cadena con el formato SUB (subíndice) de HTML.

sup() Devuelve una cadena con el formato SUP (superíndice) de HTML.