Propiedades de JavaScript

JavaScript incorpora una serie de herramientas y utilidades para el manejo de las variables. De esta forma, muchas de las operaciones básicas con las variables, se pueden realizar directamente con las utilidades que ofrece JavaScript.

Funciones y propiedades útiles para cadenas de Texto

A continuación, se muestran algunas de las funciones más útiles para el manejo de cadenas de texto:

length, calcula la longitud de una cadena de texto (el número de caracteres que la forman).

```
var mensaje = "Hola Mundo";
var numeroLetras = mensaje.length; // numeroLetras = 10
```

El símbolo +, se emplea para concatenar varias cadenas de texto.

```
var mensaje1 = "Hola";
var mensaje2 = " Mundo";
var mensaje = mensaje1 + mensaje2; // mensaje = "Hola Mundo"
```

Además del operador +, también se puede utilizar la función concat():

```
var mensaje1 = "Hola";
var mensaje2 = mensaje1.concat(" Mundo"); // mensaje2 = "Hola Mundo"
```

Las cadenas de texto también se pueden unir con variables numéricas:

```
var variable1 = "Hola ";
var variable2 = 3;
var mensaje = variable1 + variable2; // mensaje = "Hola 3"
```

Cuando se unen varias cadenas de texto es habitual olvidar añadir un espacio de separación entre las palabras:

var mensaje1 = “Hola”;

var mensaje2 = “Mundo”;

var mensaje = mensaje1 + mensaje2; *// mensaje = “HolaMundo”*

Los espacios en blanco se pueden añadir al final o al principio de las cadenas y también se pueden indicar forma explícita:

var mensaje1 = “Hola”;

var mensaje2 = “Mundo”;

var mensaje = mensaje1 + ” ” + mensaje2; *// mensaje = “Hola Mundo”*

toUpperCase(), transforma todos los caracteres de la cadena a sus correspondientes caracteres en mayúsculas:

var mensaje1 = “Hola”;

var mensaje2 = mensaje1.toUpperCase(); *// mensaje2 = “HOLA”*

toLowerCase(), transforma todos los caracteres de la cadena a sus correspondientes caracteres en minúsculas:

var mensaje1 = “HolA”;

var mensaje2 = mensaje1.toLowerCase(); *// mensaje2 = “hola”*

charAt(posicion), obtiene el carácter que se encuentra en la posición indicada: var mensaje = “Hola”;

var letra = mensaje.charAt(0); *// letra = H*

letra = mensaje.charAt(2); *// letra = l*

indexOf(caracter), calcula la posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si el carácter se incluye varias veces dentro de la cadena de texto, se devuelve su primera posición empezando a buscar desde la izquierda. Si la cadena no contiene el carácter, la función devuelve el valor -1:

var mensaje = “Hola”;

var posicion = mensaje.indexOf(‘a’); *// posicion = 3*

```
posicion = mensaje.indexOf('b'); // posicion = -1
```

Su función análoga es **lastIndexOf()**:

lastIndexOf(caracter) calcula la última posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si la cadena no contiene el carácter, la función devuelve el valor -1:

```
var mensaje = "Hola";
var posicion = mensaje.lastIndexOf('a'); // posicion = 3
posicion = mensaje.lastIndexOf('b'); // posicion = -1
```

La función **lastIndexOf()** comienza su búsqueda desde el final de la cadena hacia el principio, aunque la posición devuelta es la correcta empezando a contar desde el principio de la palabra.

substring(inicio, final) extrae una porción de una cadena de texto. El segundo parámetro es opcional. Si sólo se indica el parámetro inicio, la función devuelve la parte de la cadena original correspondiente desde esa posición hasta el final:

```
var mensaje = "Hola Mundo";
var porcion = mensaje.substring(2); // porcion = "la Mundo"
porcion = mensaje.substring(5); // porcion = "Mundo"
porcion = mensaje.substring(7); // porcion = "ndo"
```

Si se indica un inicio negativo, se devuelve la misma cadena original:

```
var mensaje = "Hola Mundo";
var porcion = mensaje.substring(-2); // porcion = "Hola Mundo"
```

Cuando se indica el inicio y el final, se devuelve la parte de la cadena original comprendida entre la posición inicial y la inmediatamente anterior a la posición final (es decir, la posición inicio está incluida y la posición final no):

```
var mensaje = "Hola Mundo";
var porcion = mensaje.substring(1, 8); // porcion = "ola Mun"
```

```
porcion = mensaje.substring(3, 4); // porcion = "a"
```

Si se indica un final más pequeño que el inicio, JavaScript los considera de forma inversa, ya que automáticamente asigna el valor más pequeño al inicio y el más grande al final:

```
var mensaje = "Hola Mundo";
var porcion = mensaje.substring(5, 0); // porcion = "Hola "
porcion = mensaje.substring(0, 5); // porcion = "Hola "
```

split(separador), convierte una cadena de texto en un array de cadenas de texto. La función parte la cadena de texto determinando sus trozos a partir del carácter separador indicado:

```
var mensaje = "Hola Mundo, soy una cadena de texto!";
var palabras = mensaje.split(" ");
// palabras = ["Hola", "Mundo,", "soy", "una", "cadena", "de", "texto!"];
```

Con esta función se pueden extraer fácilmente las letras que forman una palabra:

```
var palabra = "Hola";
var letras = palabra.split(""); // letras = ["H", "o", "l", "a"]
```

Funciones útiles para Arrays

A continuación, se muestran algunas de las funciones más útiles para el manejo de arrays: la propiedad **length** calcula el número de elementos de un array.

```
var vocales = ["a", "e", "i", "o", "u"];
var numeroVocales = vocales.length; // numeroVocales = 5
```

La función **concat()** se emplea para concatenar los elementos de varios arrays.

```
var array1 = [1, 2, 3];
array2 = array1.concat(4, 5, 6); // array2 = [1, 2, 3, 4, 5, 6]
array3 = array1.concat([4, 5, 6]); // array3 = [1, 2, 3, 4, 5, 6]
```

join(separador), es la función contraria a **split()**. Une todos los elementos de un array para formar una cadena de texto. Para unir los elementos se utiliza el caracter separador indicado.

```
var array = [“hola”, “mundo”];
var mensaje = array.join(“”); // mensaje = “holamundo”
mensaje = array.join(” “); // mensaje = “hola mundo”
```

pop() elimina el último elemento del array y lo devuelve. El array original se modifica y su longitud disminuye en 1 elemento.

```
var array = [1, 2, 3];
var ultimo = array.pop();
// ahora array = [1, 2], ultimo = 3
```

push() añade un elemento al final del array. El array original se modifica y aumenta su longitud en 1 elemento. (También es posible añadir más de un elemento a la vez).

```
var array = [1, 2, 3];
array.push(4);
// ahora array = [1, 2, 3, 4]
```

shift() elimina el primer elemento del array y lo devuelve. El array original se ve modificado y su longitud disminuida en 1 elemento.

```
var array = [1, 2, 3];
var primero = array.shift();
// ahora array = [2, 3], primero = 1
```

unshift() añade un elemento al principio del array. El array original se modifica y aumenta su longitud en 1 elemento. (También es posible añadir más de un elemento a la vez).

```
var array = [1, 2, 3];
array.unshift(0);
```

```
// ahora array = [0, 1, 2, 3]
```

reverse() modifica un array colocando sus elementos en el orden inverso a su posición original:

```
var array = [1, 2, 3];
array.reverse();
// ahora array = [3, 2, 1]
```

Funciones útiles para Números

A continuación, se muestran algunas de las funciones y propiedades más útiles para el manejo de números.

NaN: (del inglés, *"Not a Number"*) JavaScript emplea el valor NaN para indicar un valor numérico no definido (por ejemplo, la división 0/0).

```
var numero1 = 0;
var numero2 = 0;
alert(numero1/numero2); // se muestra el valor NaN
```

isNaN() permite proteger a la aplicación de posibles valores numéricos no definidos

```
var numero1 = 0;
var numero2 = 0;
if(isNaN(numero1/numero2)) {
alert("La división no está definida para los números indicados");
}
else {
 alert("La división es igual a => " + numero1/numero2);
}
```

Infinity hace referencia a un valor numérico infinito y positivo (también existe el valor – Infinity para los infinitos negativos).

```
var numero1 = 10;

var numero2 = 0;

alert(numero1/numero2); // se muestra el valor Infinity
```

toFixed(digitos) devuelve el número original con tantos decimales como los indicados por el parámetro digitos y realiza los redondeos necesarios. Se trata de una función muy útil por ejemplo para mostrar precios.

```
var numero1 = 4564.34567;

numero1.toFixed(2); // 4564.35

numero1.toFixed(6); // 4564.345670

numero1.toFixed(); // 4564
```