Objeto Array

Las matrices o arrays son uno de los objetos predefinidos del núcleo de JavaScript. Recordemos que un objeto es simplemente una colección ordenada de otros tipos de datos (números, cadenas de caracteres, booleanos, otros objetos, o incluso otros objetos predefinidos).

La peculiaridad de los arrays es que están diseñados para facilitar el acceso a listas de datos y la realización de operaciones con esas listas; para referirse a un dato de la lista sólo hay que indicar la posición, denominada índice, que ocupa dentro de ella, teniendo en cuenta que el primer elemento tiene asociado el índice 0.

Por ejemplo, en el siguiente código se crea un array a la vez que se asignan varios datos a su colección, y luego nos referimos a uno de ellos por la posición que ocupa, es decir, por su índice:

Un array en javascript se puede inicializar de las siguientes formas

```
a = [];  // array vacio
a = [1, 'hola', true];  // array con elementos
a = new Array();     // array vacío
```

```
var frutas = new Array('pera','manzana','naranja');
alert(frutas[2]);//naranja
```

También podemos crear un array vacío y asignar datos a su lista posteriormente, incluso dejando huecos, como en el ejemplo siguiente:

```
var frutas = new Array();
frutas [0] ='naranja';
frutas [2] ='pera';
alert(frutas[1]);//undefined
```

O declarar la longitud de la lista del array aunque no asignemos ninguno de los datos o sólo algunos de ellos. Por ejemplo:

```
var frutas = new Array(3);
frutas [2] ='naranja';
alert(frutas[2]);//naranja
```

Los objetos de tipo Array poseen la propiedad length, que nos indica y establece qué capacidad tiene la lista del array (no cuántos elementos contiene expresamente, ni tampoco cuál es el índice del mayor de ellos), y, ¡atención!, es una propiedad de lectura pero también de escritura (es decir, podemos cambiar su valor). Si acortamos la longitud de la lista de un array a través de su propiedad length la estaremos truncando.

Metodos

Método	**Descripción**	**Ejemplo**

arrayObj.concat(*array*)	Devuelve un array cuya lista está compuesta por los elementos del array al que se aplica, seguidos de los de *array*.	var colores = new Array ('rojo','verde','azul');
		var numeros = new Array(1,2,3);
		alert(colores.concat(numeros));

arrayObj.forEach(*función*)	Para cada elemento del array llama a la función *función* enviándole como argumentos el valor del elemento, su índice y el array completo.	**var** miMatriz = new Array('rojo','verde','azul'); **function** mostrar**(**valor,indice,matriz**)**{ alert('El valor del elemento [' + indice+ '] de ' + matriz + ' es ' + valor);

		} miMatriz.forEach(mostrar);

arrayObj.indexOf (valor)	Nos devuelve la posición de un elemento en el array, -1 si no existe.	var a = new Array('hola','adios');
		a.indexOf('hola'); // devuelve 0
		a.indexOf('adios'); // devuelve 1
		a.indexOf('tres'); // devuelve -1

arrayObj.lastIndexOf(valor)	Nos devuelve la posición de un elemento en el array, -1 si no existe. Empieza a buscar por detrás	var a=new Array("a","b","a");
		a.indexOf("a"); // Devuelve 0
		a.lastIndexOf("a"); // Devuelve 2

arrayObj.join(*separador*)	Devuelve una cadena de caracteres compuesta por los elementos de la lista del array separados mediante la cadeja *separador*. Puede considerarse el método contrario de otro llamado split que está disponible en los datos de tipo String.	var colores = new Array ('rojo','verde','azul'); alert(colores.join('+'));
arrayObj.pop()	Elimina el último elemento del array y devuelve ese elemento.	var colores = new Array ('rojo','verde','naranja'); alert(colores.pop());
arrayObj.push(*elemento*[,*elemento,…*])	Añade uno o más elementos al final del array y devuelve la longitud resultante de la lista.	var colores = new Array ('rojo','verde','naranja'); alert(colores.push('amarillo'));

arrayObj.reverse()	Invierte el orden de los elementos del array y devuelve el array con el nuevo orden.	var colores = new Array('rojo','verde','azul');
		colores.reverse();
		alert(colores);

arrayObj.shift()	Elimina el primer elemento del array de devuelve ese elemento.	var colores = new Array ('rojo','verde','naranja'); alert(colores.shift());
arrayObj.slice(*inicio*[,*fin*])	Devuelve un array con los elementos comprendidos entre el índice *inicio* incluido y el índice *fin* excluido. Si se omite *fin* utiliza todos los elementos a partir de inicio. Puede resultar útil para asignar un array por valor en lugar de por referencia.	var colores = new Array ('rojo','verde','naranja'); alert(colores.slice(1,2));
arrayObj.sort()	Ordena los elementos de la lista (alfanuméricamente, con las mayúsculas antes que las minúsculas) y devuelve el array ordenado.	var colores = new Array('rojo','verde','Rojo',5); alert(colores.sort());

arrayObj.splice(*inicio*,*eliminar*,[*elemento1*, *elemento2*, …])	Sirve para eliminar o añadir elementos a partir de la posición *inicio* de un array. Si *eliminar* es distinto de 0 se elimina ese número de elementos, incluido *inicio*, y se inserta en su lugar *elemento1*, *elemento2*, … Si *eliminar* es 0, se insertan *elemento1*, *elemento2*, … en la posición *inicio* desplazando los elementos existentes hacia la derecha. Si se usa para eliminar devuelve un array con la lista de elementos eliminados; si se usa para insertar no devuelve nada.	var colores = new Array('rojo','verde','azul');
		colores.splice(1,2,'naranja','rosa','violeta');
		alert(colores);
		colores.splice(2,0,'amarillo');
		alert(colores);

arrayObj.unshift(*elemento 1*[,*elemento2, …*])	Añade uno o más elementos al principio del array y devuelve la longitud resultante de su lista.	var colores = new Array ('rojo','verde','naranja'); alert(colores.unshift('amarillo'));

array1.forEach(callbackfn)

Parámetros

Parámetro	Definición
array1	Obligatorio. Objeto de matriz.
callbackfn	Obligatorio. Función que acepta un máximo de tres argumentos. **forEach** llama a la función callbackfn una vez para cada elemento de la matriz.

Sintaxis de la función de devolución de llamada

La sintaxis de la función de devolución de llamada es la siguiente:

```
function callbackfn(value, index, array1)
```

Puedes declarar la función de devolución de llamada usando hasta tres parámetros.

Los parámetros de la función de devolución de llamada son los siguientes.

Argumento de devolución de llamada	Definición
value	Valor del elemento de la matriz.
index	Índice numérico del elemento de la matriz.
array1	Objeto Array que contiene el elemento.

Ejemplo

En el ejemplo siguiente se muestra el uso del método **forEach**.

```
// Se define la función de llamada.
function ShowResults(value, index, ar) {
   document.write("value: " + value);
   document.write(" index: " + index);
   document.write("<br />");
}

// Se crea el array.
var letters = new Array('ab', 'cd', 'ef');

// Se llama a la función ShowResults para cada
// elemento del array.
letters.forEach(ShowResults);
// Output:
//  valor: ab indice: 0
//  valor: cd indice: 1
//  valor: ef indice: 2
```

En el siguiente ejemplo, el argumento callbackfn incluye el código de la función de devolución de llamada.

JavaScript

```
// Crea el array.
var numbers = [10, 11, 12];
var sum = 0;
function addNumber(value) { sum += value; }
// Llama a la función addNumber para cada elemento del array.
numbers.forEach(addNumber);
document.write(sum);
// Output: 33
```

arrayobj.sort(sortFunction)
El método **sort** ordena el objeto **Array**; no se crea ningún objeto **Array** nuevo durante la ejecución. Si proporciona una función en el argumento sortFunction, debe ser una función que reciba como parámetros dos elementos del tipo con el que trabaja el array y debe devolver uno de los siguientes valores:

- Un valor negativo si el primer elemento debe ordenarse antes que el segundo.
- Cero si ambos elementos tienen igual orden.
- Un valor positivo si el segundo elemento debe ordenarse antes que el primero.

Ejemplo

En el siguiente ejemplo, se muestra cómo utilizar el método **sort**.

JavaScript

```
var a = new Array(4, 11, 2, 10, 3, 1);

var b = a.sort();
document.write(b);
document.write("<br/>");

// Este es el orden de los caracteres ASCII.
// Output: 1,10,11,2,3,4)

// Ordenación de los elementos del array mediante el uso de una función.
b = a.sort(CompareForSort);
document.write(b);
document.write("<br/>");
// Output: 1,2,3,4,10,11.

// Ordena los elementos del array en orden numérico ascendente.
function CompareForSort(first, second)
{
   if (first == second)
      return 0;
   if (first < second)
      return -1;
   else
      return 1;
}
```

Estructura for ... in

La estructura Javascript **for in** nos permite recorrer una lista de elementos de una forma sencilla. Javascript for in es una estructura en bucle que nos permite tratar los elementos indicados en la sentencia.

La estructura Javascript for in tiene la siguiente sintaxis:

```
    for (variable in objeto) {

        // instrucciones

}
```