La alegría de crecer juntos

Plaza de Castilla - Madrid
COLEGIO NUESTRA SEÑORA DEL PILAR
Salesianas

C.F.G.S.: DESARROLLO DE APLICACIONES WEB
Módulo: DESARROLLO WEB EN ENTORNO CLIENTE

07 ARRAYS

Objetos predefinidos del núcleo de JavaScript

- Un array es una lista de datos a los que se accede a través de un índice. El primer elemento tiene asociado el índice 0.

- Para crear un array:

  var nombre_del_array = new Array ();

  con lo que estamos creando una instancia del objeto Array

- Crear un array dándole valores:

```
var vocales = new Array ('a','e','i','o','u');
for (i=0;i<6;i++)
        document.write ('<br>'+i+vocales[i]);
0a
1e
2i
3o
4u
5undefined
```

- Crear un array vacío y darle valores posteriormente

```
var letras = new Array ();
letras[2]='c';
letras[4]='e';
for (i=0;i<=5;i++)
        document.write('<br>'+i+letras[i]);
0undefined
1undefined
2c
3undefined
4e
5undefined
```

- También podemos crear arrays así: **Notación literal de arrays**

```
var letras = ['a','b','c','d','e'];
for (i=0;i<letras.length;i++)
        document.write ('<br>'+i+letras[i]);
```

- Y ver todo el contenido de un array así:

```
alert (letras);
a,e,i,o,u
```

- **Longitud de un array**

  Establece la capacidad del array, no cuantos elementos contiene expresamente.

  - Podemos declarar la longitud del array al declararlo:

```
var letras = new Array (3);
alert (letras.length);  // 3
```

  - Después podemos asignar valores a posiciones fuera de la longitud, lo que provoca que se amplíe el array y cambie su longitud

```
letras[2]='a';
letras[4]='b';
for (i=0;i<=5;i++)
        document.write('<br>'+i+letras[i]);
alert (letras.length);    // 5

        0undefined
        1undefined
        2a
        3undefined
        4b
        5undefined
```

```
letras[0]='c';
letras[5]='d';
for (i=0;i<=5;i++)
        document.write('<br>'+i+letras[i]);
alert (letras.length);  // 6

        0c
        1undefined
        2a
        3undefined
        4b
        5d
```

- Si modificamos nosotros la longitud del array el valor de los elementos a los que habíamos asignado valor y que se salen de la nueva longitud quedan sin valor.

```
letras.length = 3;
for (i=0;i<=4;i++)
        document.write('<br>'+i+letras[i]);

        0c
        1undefined
        2a
        3undefined
        4undefined
```

```
var a = new Array ();
alert (a.length);    // 0
var b = new Array (3);
alert(b.length);  // 3
var c = new Array (5,9);
alert(c.length);  // 2
```

- En un mismo array pueden mezclarse distintos tipos de datos.

```
var array = new Array (5,'patata',true);
for (i=0;i<array.length;i++)
        document.write ('<br>'+i+' tipo '+typeof array[i]);

0 tipo number
1 tipo string
2 tipo boolean
```

- Podemos obtener arrays multidimensionales haciendo que un elemento del array sea otro array

```
var tabla = new Array (3);
tabla [0] = new Array (1,2,3);
tabla [1] = new Array (4,5,6);
tabla [2] = new Array (7,8,9);
for (i=0;i<3;i++) {
        for (j=0;j<3;j++)
                document.write (tabla[i][j]);
        document.write ('<br>');
        }
        123
        456
        789
```

- Con notación literal

```
var tabla = [[1,2,3],[4,5,6],[7,8,9]];
for (i=0;i<3;i++) {
        for (j=0;j<3;j++)
                document.write (tabla[i][j]);
        document.write ('<br>');
        }
```

En general, cuando trabajemos con arrays de 2 dimensiones, con "tablas", seguiremos los siguientes pasos:

1. **Definir la tabla**
2. **Cargar de datos la tabla**
3. **Construir a partir de la tabla el código HTML para mostrar sus datos**

Ejemplo

Definir una tabla de 3 filas y 5 columnas. Rellenarla después con los números del 1 al 15. Recorrerla para construir una tabla html que meteremos en un div

1. Definir la tabla

```
var fil=3;
var col=5;
var tabla =new Array(fil);
for (i=0;i<fil;i++)
                tabla[i]=new Array(col);
```

2. Cargar la tabla con datos

```
var n=1;
for (i=0;i<fil;i++)
        for (j=0;j<col;j++){
                tabla[i][j]=n;
                n++;
                }
```

3. Construir la tabla HTML

```
var codigo='<table border=1> ';
for (i=0;i<fil;i++){
        codigo+='<tr> ';
        for (j=0;j<col;j++){
                codigo+='<td>'+tabla[i][j]+'</td>' ;
                }
        codigo+='</tr>';
        }
codigo+='</table>';
document.getElementById("div1").innerHTML=codigo;
```

VER EJEMPLOS 3a, 3b, 3c

Métodos más importantes de los objetos de tipo Array

http://www.w3schools.com/js/js_array_methods.asp

```
var vocales = new Array ('a','e','i','o','u');
var letras = ['a','b','c'];
```

- **concat (array)**
  - Para concatenar los elementos de dos arrays
  - Devuelve un array cuya lista está compuesta por los elementos del array al que se aplica, seguidos de los del *array que recibe*

```
alert (vocales.concat (letras));  // a,e,i,o,u,a,b,c
var todas = vocales.concat (letras); // crea el array todas con la concatenación
```

- **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.

```
alert (vocales. join(' * '));//    a * e * i * o * u
```

- **indexOf (aguja,[inicio])**
  - Devuelve el índice que ocupa el elemento *aguja* en la lista del array buscando de izquierda a derecha y comenzando a partir de *inicio*.
  - En ausencia de *inicio*, busca desde el elemento 0.
  - Si no se encuentra *aguja* devuelve -1.

```
alert (vocales.indexOf ('o'));  // 3
```

- **lastIndexOf(aguja,[inicio])**
  - Devuelve el índice que ocupa el elemento *aguja* en la lista del array buscando de derecha a izquierda y comenzando a partir de *inicio*.
  - En ausencia de *inicio* busca desde el último elemento
  - Si no se encuentra *aguja* devuelve -1.

```
alert (todas.lastIndexOf ('a'));  //  5
```

- **shift()**
  - Elimina el primer elemento del array y devuelve ese elemento.

```
alert (vocales); // a,e,i,o,u
alert (vocales.shift());  // a
```

```
alert (vocales+' '+vocales.length);  // e,i,o,u 4
```

- **pop()**
  - Elimina el último elemento del array y devuelve ese elemento.

```
alert (vocales);  // a,e,i,o,u
alert (vocales.pop());  // u
alert (vocales+' '+vocales.length);  // a,e,i,o  4
```

- **unshift(*elemento*[*,elemento,…*])**
  - Añade nuevos elementos al inicio de un array y devuelve el número de elementos del array modificado

```
alert (vocales.unshift('a','a','a'));  // 8
alert (vocales);  // a,a,a,a,e,i,o,u
```

- **push(*elemento*[*,elemento,…*])**
  - Añade uno o más elementos al final del array y devuelve la longitud resultante de su lista.

```
alert (vocales.push('a','a','a'));  // 8
alert (vocales);  //  a,e,i,o,u,a,a,a
```

- **reverse()**
  - Invierte el orden de los elementos de un array.
  - El array se queda con el nuevo orden

```
alert (vocales); // a,e,i,o,u
alert (vocales.reverse());  // u,o,i,e,a
alert (vocales); // u,o,i,e,a
```

- **sort()**
  - Ordena los elementos de la lista (alfanuméricamente, con las mayúsculas antes que las minúsculas) y devuelve el array ordenado.

```
alert (todas); // a,e,i,o,u,a,b,c
alert (todas.sort()); // a,a,b,c,e,i,o,u
```

  - Si lo que queremos es ordenar un array de números hay que pasar en el sort un argumento que indique el “método que queremos que use” al ordenar.

    var num = [1,22,100,300,45,33]

    alert (num.sort()); // 1,100,22,300,33,45

    Numericamente ascendente:

    alert(num.sort(function(a,b){return (a-b)})); // 1,22,33,45,100,300

    Numéricamente descendente

```
alert(num.sort(function(a,b){return (b-a)}));// 300,100,45,33,22,1
```

- **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.

```
alert (todas); // a,e,i,o,u,a,b,c
alert (todas.slice(3,6));  // o,u,a
```

- **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

```
alert (vocales); // a,e,i,o,u
alert (vocales.splice(3,0,'s','r'));
alert (vocales); // a,e,i,s,r,o,u
alert (vocales.splice(3,2,'x','y')); // s,r
alert (vocales); //  a,e,i,x,y,o,u
```