Materia: Tecnologías para la Web.
Tema: JavaScript: Arreglos.

1. Introducción

- Los arreglos son estructuras de datos que consisten en elementos de datos relacionados (algunas veces se conocen como colecciones de elementos de datos).
- Los arreglos de JavaScript son entidades "dinámicas" en cuanto a que pueden cambiar su tamaño después de crearse.

2. Arreglos

- Un arreglo es un grupo de ubicaciones de memoria, en donde todas tienen el mismo nombre y por lo general son del mismo tipo (aunque este atributo no es obligatorio en JavaScript).
- Cada ubicación individual se conoce como elemento. Para hacer referencia a cualquiera de estos elementos hay que proporcionar el nombre del arreglo, seguido del número de posición (un entero que por lo general se conoce como el índice, del elemento en corchetes ([ ]).
- El primer elemento de cada arreglo es el elemento cero. El i-ésimo elemento del arreglo c se conoce como c [1 -1]. Los nombres de los arreglos siguen las mismas convenciones que otros identificadores.
- El nombre de un arreglo indexado puede usarse del lado izquierdo de una asignación para colocar un nuevo valor en un elemento del arreglo. También puede usarse del lado derecho de una operación de asignación para asignar su valor a otra variable.
- Todo arreglo en JavaScript conoce su propia longitud que almacena en su atributo `1ength`.

3. Declaración y asignación de arreglos

- En JavaScript los arreglos se representan mediante objetos `Array`.
- Los arreglos se crean con el operador `new`.

4. Ejemplos del uso de arreglos

- Para vincular un archivo de JavaScript a un documento de HTM L, use el atributo `src` del elemento script para especificar la ubicación del archivo de JavaScript.
- El conteo basado en cero se utiliza por lo general para iterar a través de arreglos.
- JavaScript reasigna un arreglo de manera automática cuando se asigna un valor a un elemento que se encuentra fuera de los límites del arreglo original. Los elementos entre el último elemento del arreglo original y el nuevo elemento tienen valores indefinidos.
- Los arreglos pueden crearse mediante el uso de una lista inicializadora separada por comas encerrada entre corchetes ([y]). El tamaño del arreglo se determina mediante el número de valores en la lista inicializadora.
- Los valores iniciales de un arreglo también pueden especificarse como argumentos en los paréntesis que van después de `new Array`. El tamaño del arreglo se determina mediante el número de valores entre paréntesis.
- La instrucción `for...in` de JavaScript permite a una secuencia realizar una tarea para cada demento en un arreglo. Este proceso se conoce como iterar sobre los elementos de un arreglo.

EJEMPLO 1.
Archivo InicArreglo.html:

```
<!-- Página Web para mostrar los resultados de inicializar arreglos . -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8">
            <title> Inicializar un arreglo</title>
            <link rel="stylesheet" type="text/css" href="estilotabla.css">
            <script src="InicArreglo.js"></script>
      </head>
      <body>
```

```
            <div id="salida1"></div>
            <div id="salida2"></div>
      </body>
</html>
```

Archivo InicArreglo.js:

```
// InicArreglo.js
function iniciar(){
      var n1 = new Array( 5 );      // asignar un arreglo de cinco elementos
      var n2 = new Array( );        // asignar el arreglo vacío
      // asignar valores a cada elemento del arreglo n1
      var longitud = n1.length;     // la longitud del arreglo una vez antes del ciclo
      for( var i = 0; i < n1.length; ++i ){
            n1[ i ] = i;
      } // fin de for
      for ( i = 0; i < 5; ++i ){    // crear e inicializar cinco elementos en el arreglo n2
            n2 [ i ] = i;
      } // fin de for
      imprimirArreglo( "Arreglo n1: ", n1, document.getElementById( "salida1" ) ) ;
      imprimirArreglo( "Arreglo n2: ", n2, document.getElementById( "salida2" ) ) ;
} // fin de la función iniciar
// imprime el encabezado seguido de una tabla de dos columnas
// que contienen los índices y elementos de "elArreglo"
function imprimirArreglo( encabezado, elArreglo, salida ){
      var contenido = "<h2>" + encabezado + "</h2><table>" +
            "<thead><th>Indice</th><th>Valor</th></thead><tbody>";
      // imprime el índice y el valor de cada elemento del arreglo
      var longitud = elArreglo.length; // la longitud del arreglo una vez antes del ciclo
      for (var i = 0; i < longitud; ++i ){
            contenido += "<tr><td>" + i + "</td><td>" + elArreglo[ i ] + "</td></tr>";
      } // fin de for
      contenido += "</tbody></table>";
      salida.innerHTML = contenido; // coloca la tabla en el elemento salida
} // fi n de la función imprimirArreglo
window.addEventListener( "load", iniciar, false ) ;
```

Archivo estilotabla.css:

```
table         { width: 300px;
                border-collapse:collapse;
                background-color: lightblue; }
table, td, th { border: 1px solid black;
                padding: 4px; }
td            { width: 50%; }
th            { text-align: left;
                color: white;
                background-color: darkblue; }
tr.oddrow     { background-color: white; }
```

EJEMPLO 2.

Archivo InicArreglo2.html:

```
<!DOCTYPE html>
<!-- Página Web para mostrar los resultados de inicializar arreglos. -->
<html>
	<head>
		<meta charset="utf-8">
		<title>Inicializar un arreglo</title>
		<link rel = "stylesheet" type = "text/css" href = "estilotabla.css">
		<script src = "InicArreglo2.js" ></script>
	</head>
	<body>
		<div id = "salida1" ></div>
		<div id = "salida2" ></div>
		<div id = "salida3" ></div>
	</body>
</html>
```

Archivo InicArreglo2.js:

```
// Inicialización de arreglos con listas inicializadoras.
function iniciar(){
	// La lista inicializadora especifica el número de elementos y valor para cada uno.
	var colores = new Array( "cyan", "magenta", "amarillo", "negro" );
	var enteros1 = [ 2, 4, 6, 8 ];
	var enteros2 = [ 2, , , 8 ];
	imprimirArreglo( "El arreglo colores contiene" , colores,
		document.getElementById( "salida1" ) );
	imprimirArreglo( "El arreglo enterosl contiene" , enteros1,
		document.getElementById( "salida2" ) );
	imprimirArreglo( "El arreglo enteros2 contiene", enteros2,
		document.getElementById( "salida3" ) );
} // fin de la función iniciar
// imprimir el encabezado seguido de una tabla de dos columnas
// que contiene índices y elementos de "elArreglo"
function imprimirArreglo( encabezado, elArreglo, salida ){
	var contenido = "<h2>" + encabezado + "</h2><table>" +
		"<thead><th>Index</th><th>Value</th></thead><tbody>";
	// imprime el índice y el valor de cada elemento del arreglo
	var longitud = elArreglo.length; // obtiene tamaño del arreglo después del ciclo
	for ( var i = 0; i < longitud; ++i ){
		contenido += "<tr><td>" + i + "</td><td>" + elArreglo[ i ] + "</td></tr>";
	} // fin de for
	contenido += "</tbody></table>";
```

```
      salida.innerHTML = contenido; // colocar la tabla en el elemento salida
} // fin de la función imprimirArreglo
window.addEventListener( "load", iniciar, false );
```

Archivo estilotabla.css:

```
table         { width: 300px;
                border-collapse:collapse;
                background-color: lightblue; }
table, td, th { border: 1px solid black;
                padding: 4px; }
td            { width: 50%; }
th            { text-align: left;
                color: white;
                background-color: darkblue; }
tr.oddrow     { background-color: white; }
```

EJEMPLO 3.

Archivo SumaArreglo.html:

```
<!-- Documento de HTML5 que muestra la suma de los elementos de un arreglo. -->
<!DOCTYPE html>
<html>
      <head>
            <neta charset = "utf-8">
            <title>Suma de los elementos de un arreglo</title>
            <script src = "SumaArreglo.js" ></script>
      </head>
      <body>
            <div id = "salida"></div>
      </body>
</html>
```

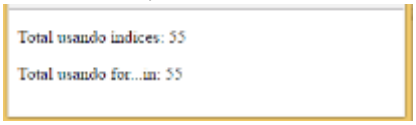
Archivo SumaArreglo.js:

```
// Suma de los elementos de un arreglo con for y for...in
function iniciar(){
      var elArreglo = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ];
      var total1 = 0, total2 = 0;
      // itera a través de los elementos del arreglo en orden y suma
      // el valor de cada elemento a total1
      var longitud = elArreglo.length; // la longitud del arreglo una vez antes del ciclo
      for ( var i = 0; i < elArreglo.length; ++i ){
            total1 += elArreglo[ i ];
```

```
      } // fin de for
      var resultados = "<p>Total usando &iacute;ndices: " + total1 + "</p>";
      // itera a través de los elementos del arreglo usando una instrucción
      // for...in para sumar el valor de cada elemento a total2
      for ( var elemento in elArreglo ){
            total2 += elArreglo[ elemento ];
      } // fin de for
      resultados += "<p>Total usando for...in: " + total2 + "</p>";
      document.getElementById( "salida" ).innerHTML = resultados;
} // fin de la función iniciar
window.addEventListener( "load", iniciar, false );
```

Archivo estilotabla.css:

```
table          { width: 300px;
                 border-collapse:collapse;
                 background-color: lightblue; }
table, td, th { border: 1px solid black;
                 padding: 4px; }
td             { width: 50%; }
th             { text-align: left;
                 color: white;
                 background-color: darkblue; }
tr.oddrow      { background-color: white; }
```



5. Generador de imágenes al azar mediante el uso de arreglos

- Crear un generador de imágenes al azar más elegante, ya que no requiere que los nombres de archivos de tas imágenes sean enteros, mediante el uso del arreglo Imágenes para almacenar los nombres de los archivos de imágenes como cadenas y accediendo al arreglo mediante un índice distribuido al azar.

EJEMPLO 4.
Archivo TirarDados.html:

```
<!-- Documento de HTML5 para el ejemplo de tiro de dados. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8">
            <title>Frecuencias de tiro de dado</title>
            <link rel = "stylesheet" type = "text/css" href = "estilo.css">
            <script src = "TirarDados.js" ></script>
      </head>
      <body>
            <p>   <img id = "dado1" src = "blanco.png" alt = "imagen dado 1">
                  <img id = "dado2" src = "blanco.png" alt = "imagen dado 2">
                  <img id = "dado3" src = "blanco.png" alt = "imagen dado 3">
                  <img id = "dado4" src = "blanco.png" alt = "imagen dado 4">
                  <img id = "dado5" src = "blanco.png" alt = "imagen dado 5">
                  <img id = "dado6" src = "blanco.png" alt = "imagen dado 6"></p>
            <p>   <img id = "dado7" src = "blanco.png" alt = "imagen dado 7">
                  <img id = "dado8" src = "blanco.png" alt = "imagen dado 8">
                  <img id = "dado9" src = "blanco.png" alt = "imagen dado 9">
                  <img id = "dado10" src = "blanco.png" alt = "imagen dado 10">
                  <img id = "dado11" src = "blanco.png" alt = "imagen dado 11">
```

```
                  <img id = "dado12" src = "blanco.png" alt = "imagen dado 12"></p>
            <form action = "#">
                  <input id = "botonTirar" type = "button" value = "Tirar dados">
            </form>
            <div id = "divTablaFrecuencias"></div>
      </body>
</html>
```

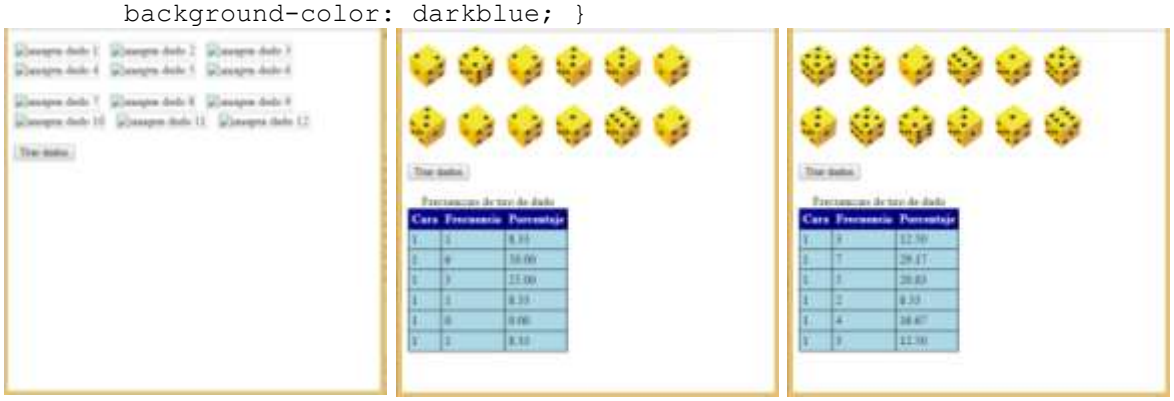
Archivo TirarDados.js:

```
// Resumen de las frecuencias de tiro de dado con un arreglo en vez de una instrucción
switch
var frecuencia = [ , 0, 0, 0, 0, 0, 0 ];  // frecuencia[0] sin inicializar
var totalDados = 0;
var imagenesDado = new Array(12); // arreglo para almacenar elementos img
// obtener elementos img de dado
function iniciar(){
      var boton = document.getElementById( "botonTirar" );
      boton.addEventListener( "click" , tirarDados, false );
      var longitud = imagenesDado.length; // longitud del arreglo una vez antes del ciclo
      for ( var i = 0 ; i < longitud; ++i ){
            imagenesDado[ i ] = document.getElementById( "dado" + (i + 1) );
      } // fin de for
} // fin de la función iniciar
// tirar los dados
function tirarDados(){
      var cara; // cara que se tiró
      var longitud = imagenesDado.length;
      for ( var i = 0; i < longitud; ++i ){
            cara = Math.floor( 1 + Math.random() * 6 );
            calcularTiros( cara ); // incrementar un contador de frecuencia
            establecerImagen( i, cara ); // mostrar imagen de dado apropiada
            ++totalDados; // incrementar total
      } // fin de for
      actualizarTablaFrecuencias();
} // fin de la función tirarDados
// incrementar contador de frecuencia apropiado
function calcularTiros( cara ){
      ++frecuencia[ cara ]; // incrementar contador apropiado
} // fin de la función calcularTiros
// establecer el origen de la imagen para un dado
function establecerImagen( numeroDado, cara ){
      imagenesDado[ numeroDado ].setAttribute( "src" , "dado" + cara + ".png" );
      imagenesDado[ numeroDado ].setAttribute( "alt" , "dado con " + cara + "punto(s)" );
} // fin de la función establecerImagen
// actualizar la tabla de frecuencias en la página
function actualizarTablaFrecuencias(){
      var resultados = "<table><caption>Frecuencias de tiro de dado</caption>" +
            "<thead><th>Cara</th><th>Frecuencia</th>" +
            "<th>Porcentaje</th></thead><tbody>";
      var longitud = frecuencia.length;
      // crear filas de tabla para frecuencias
      for ( var i = 1; i < longitud; ++i ){
            resultados += "<tr><td>1</td><td>" + frecuencia[ i ] + "</td><td>" +
            formatoPorciento(frecuencia [ i ] / totalDados) + "</td></tr>";
      } // fin de for
      resultados += "</tbody></table>";
      document.getElementById( "divTablaFrecuencias" ).innerHTML = resultados;
```

```
} // fin de la función actualizarTablaFrecuencias
// formato de porcentaje
function formatoPorciento( valor ){
      valor *= 100;
      return valor.toFixed(2);
} // fin de la función formatoPorciento
window.addEventListener( "load", iniciar, false );
```

Archivo estilo.css:

```
img            { margin-right: 10px; }
table          { width: 200px;
                 border-collapse:collapse;
                 background-color: lightblue; }
table, td, th { border: 1px solid black;
                 padding: 4px;
                 margin-top: 20px; }
th             { text-align: left;
                 color: white;
                 background-color: darkblue; }
```



EJEMPLO 5.

Archivo ImagenAleatoria.html:

```
<!-- Documento de HTML5 que muestra imágenes seleccionadas al azar. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8">
            <title>Generador de im&aacute;genes al azar</title>
            <script src = "ImagenAleatoria.js"></script>
      </head>
      <body>
            <img id = "imagen" src = "ECP.png" alt = "Error usual de programaci&oacute;n">
      </body>
</html>
```

Archivo ImagenAleatoria.js:

```
// Selección de imágenes al azar usando arreglos
var imgIcono;
var imagenes = [ "CPE", "EPT", "GPP", "GUI", "PERF", "PORT", "SEO" ];
var descripciones = [ "Error com&uacute;n de programaci&oacute;n",
      "Tip para prevenir errores" , "Buena pr&aacute;ctica de programaci&oacute;n",
      "Observaci&oacute;n de apariencia visual" , "Tip de rendimiento", "Tip de
portabilidad",
      "Observaci&oacute;n de ingenier&iacute;a de software" ];
```

```
// elegir una imagen al azar y su descripción correspondiente, después modificar
// el elemento img en el cuerpo del documento
function elegirImagen(){
      var indice = Math.floor( Math.random() * 7 );
      imgIcono.setAttribute( "src", imagenes[ indice ] + ".png" );
      imgIcono.setAttribute( "alt", descripciones[ indice ] );
} // fin de la función elegirImagen
// registra el manejador de eventos de clic de imgIcono
function iniciar(){
      imgIcono = document.getElementById( "imagen" );
      imgIcono.addEventListener( "click" , elegirImagen, false );
} // fin de la función iniciar
window.addEventListener( "load", iniciar, false );
```

6. Referencias y parámetros de referencia

- Dos formas de pasar argumentos a las funciones (o métodos) en muchos lenguajes de programación son: paso por valor y paso por referencia.
- Cuando un argumento se pasa a una función por valor, se crea una copia del valor del argumento y se pasa a la función invocada.
- En JavaScript los números, valores booleanos y cadenas se pasan a las funciones por valor.
- Con el paso por referencia, la función que hace la llamada concede a la función invocada el acceso a sus datos y le permite modificarlos si así lo desea. El paso por referencia puede mejorar cl rendimiento debido a que es posible eliminar la sobrecarga de copiar grandes cantidades de datos, pero puede debilitar la seguridad debido a que la función invocada puede acceder a los datos de la función que hizo la llamada.
- En JavaScript, todos los objetos (y por ende, todos los arreglos) se pasan a las funciones por referencia.
- El nombre de un arreglo es en realidad una referencia a un objeto que contiene los elementos del arreglo y la variable `length`, que indica cl número de elementos en el arreglo.

7. Paso de arreglos a funciones

- Para pasar un argumento tipo arreglo a una función, hay que especificar el nombre del arreglo (una referencia al arreglo) sin llaves.
- Aunque los arreglos completos se pasan por referencia, los elementos numéricos y booleanos individuales de un arreglo se pasan por valor, justo igual que como se pasan las variables numéricas y booleanas simples. Dichas piezas simples e individuales de datos se conocen como escalares o cantidades escalares. Para pasar el elemento de un arreglo a una función, use el nombre indexado del elemento como un argumento en la llamada a la función.
- El método `join` recibe como argumento una cadena que contiene el separador que debe usarse para separar los elementos del arreglo en la cadena que se devuelve. Si no se especifica el argumento, se utiliza la cadena vacía como el separador.

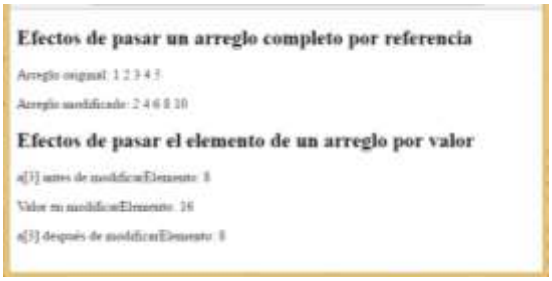
EJEMPLO 6.
Archivo PasoArreglo.html:

```
<!-- Documento de HTML que demuestra el paso de arreglos y -->
<!-- elementos individuales de arreglos a funciones. -->
<!DOCTYPE html>
<html>
      <head>
```

```
            <meta charset="utf-8">
            <title>Arreglos como argumentos</title>
            <link rel = "stylesheet" type = "text/css" href = "estilo.css">
            <script src = "PasoArreglo.js"></script>
      </head>
      <body>
            <h2>Efectos de pasar un arreglo completo por referencia</h2>
            <p id = "arregloOriginal"></p>
            <p id = "arregloModificado"></p>
            <h2>Efectos de pasar el elemento de un arreglo por valor</h2>
            <p id = "elementoOriginal"></p>
            <p id = "enModificarElemento"></p>
            <p id = "elementoModificado"></p>
      </body>
</html>
```

Archivo PasoArreglo.js:

```
// Paso de arreglos y elementos individuales de arreglos a funciones.
function iniciar(){
      var a = [ 1, 2 , 3 , 4, 5 ];
      // pasar todo un arreglo
      imprimirArreglo( "Arreglo original: ", a,
            document.getElementById( "arregloOriginal" ) );
      modificarArreglo( a ); // el arreglo a pasado por referencia
      imprimirArreglo( "Arreglo modificado: ", a,
            document.getElementById( "arregloModificado" ) );
      // pasar un elemento individual del arreglo
      document.getElementById( "elementoOriginal" ).innerHTML =
            "a[3] antes de modificarElemento: " + a[ 3 ];
      modificarElemento( a[ 3 ] ); // elemento del arreglo a[ 3 ] pasado por valor
      document.getElementById( "elementoModificado" ).innerHTML =
            "a[3] despu&eacute;s de modificarElemento: " + a[ 3 ];
} // fin de la función iniciar()
// Imprime encabezado seguido del contenido de "elArreglo"
function imprimirArreglo( encabezado, elArreglo, salida ){
      salida.innerHTML = encabezado + elArreglo.join( " " );
} // fin de la función imprimirArreglo
// función que modifica los elementos de un arreglo
function modificarArreglo( elArreglo ){
      for ( var j in elArreglo ){
            elArreglo[ j ] *= 2;
      } // fin e for
} // fin de la función modificarArreglo
// función que modifica el valor que se pasó
function modificarElemento( e ){
      e *= 2; // escala el elemento e sólo mientras dure la función
      document.getElementById( "enModificarElemento" ).innerHTML =
            "Valor en modificarElemento: " + e;
} // fin de la función modificarElemento
window.addEventListener( "load", iniciar, false );
```



8. Ordenamiento de arreglos con el método `sort` de `Array`

- El ordenamiento de datos (poner datos en un orden especifico, como ascendente o descendente) o una de las funciones de cómputo más importantes.
- El objeto `Array` en JavaScript tiene un método integrado llamado `sort` para ordenar arreglos.
- De manera predeterminada, el método `sort` de `Array` (sin argumentos) usa las comparaciones de cadena para determinar el orden de los elementos del arreglo.
- El método `sort` recibe como argumento opcional el nombre de una función (que se conoce como función comparadora, que compara sus dos argumentos y devuelve un valor negativo, cero o un valor positivo, si el primer argumento es menor, igual o mayor que cl segundo, respectivamente.
- Las funciones en JavaScript se consideran como datos. Por lo tanto, pueden asignarse a variables, almacenarse en arreglos y pasarse a funciones justo igual que otros tipos de datos.

EJEMPLO 7.

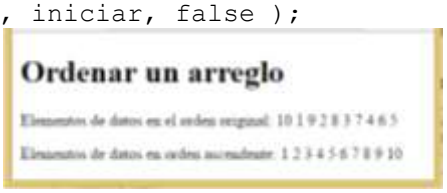
Archivo Ordenar.html:

```
<!-- Documento de HTML5 que muestra los resultados de ordenar un arreglo. -->
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title> El m&eacute;todo sort de Array</title>
		<link rel = "stylesheet" type = "text/css" href = "estilo.css">
		<script src = "Ordenar.js" ></script>
	</head>
	<body>
		<h1>Ordenar un arreglo</h1>
		<p id = "arregloOriginal"></p>
		<p id = "arregloOrdenado"></p>
	</body>
</html>
```

Archivo Ordenar.js:

```
// Ordenar un arreglo mediante sort.
function iniciar(){
	var a = [ 10, 1, 9, 2, 8, 3, 7, 4, 6, 5 ];
	imprimirArreglo( "Elementos de datos en el orden original: ", a,
		document.getElementById( "arregloOriginal" ) );
	a.sort( compararEnteros ); // ordenar el arreglo
	imprimirArreglo( "Elementos de datos en orden ascendente: ", a,
		document.getElementById( "arregloOrdenado" ) );
} // fin de la función iniciar
// imprimir el encabezado seguido del contenido de elArreglo
function imprimirArreglo( encabezado, elArreglo, salida ){
	salida.innerHTML = encabezado + elArreglo.join( " " );
} // fin de la función imprimirArreglo
// función de comparación para usar consort
```

```
function compararEnteros( valor1, valor2 ){
      return parseInt( valor1 ) - parseInt( valor2 );
} // fin de la función compararEnteros
window.addEventListener( "load", iniciar, false );
```



9. Búsqueda en arreglos con el método `indexOf` de `Array`

- El método `indexOf` de `Array` busca la primera ocurrencia de un valor y, si lo encuentra, devuelve el índice del arreglo de ese valor; en caso contrario devuelve `-1`. El método `lastIndexOf` busca la última ocurrencia.

EJEMPLO 8.

Archivo buscar.html:

```
<!-- Documento de HTML5 para buscar en un arreglo mediante indexOf. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8">
            <title>Buscar en un arreglo</title>
            <script src = "buscar.js" ></script>
      </head>
      <body>
            <form action="#">
                  <p> <label>Introduzca la clave de b&uacute;squeda entera:
                  <input id = "valEntrada" type = "number"></label>
                  <input id = "botonBuscar" type = "button" value = "Buscar">
                  </p>
                  <p id = "resultado" ></p>
            </form>
      </body>
</html>
```

Archivo buscar.js:

```
// Buscar en un arreglo mediante indexOf.
var a = new Array( 100 ); // crear un arreglo
// llenar el arreglo con valores enteros pares de 0 a 198
for ( var i = 0 ; i < a.length; ++i ){
      a [ i ] = 2 * i;
} // fin de for
// la función se invoca cuando se presiona el botón "Buscar"
function botonPresionado(){
      // obtener el campo de texto de entrada
      var valEntrada = document.getElementById( "valEntrada" );
      // obtener el párrafo de resultado
      var resultado = document.getElementById( "resultado" );
      // obtener la clave de búsqueda del campo de texto de entrada y luego realizar la
búsqueda
      var claveBusqueda = parseInt( valEntrada.value );
      var elemento = a.indexOf( claveBusqueda );
      if ( elemento != -1 ){
            resultado.innerHTML = "Se encontr&oacute; el valor en el elemento " + elemento;
      } // fin de if
      else{
            resultado.innerHTML = "No se encontr&oacute; el valor";
```

```
      } // fin de else
} // fin de la función botonPresionado
// registra el manejador de eventos de clic de botonBuscar
function iniciar(){
      var botonBuscar = document.getElementById( "botonBuscar" );
      botonBuscar.addEventListener( "click" , botonPresionado, false );
} // fin de la función iniciar
window.addEventListener( "load", iniciar, false );
```

10. Arreglos multidimensionales

- Para identificar un elemento específico de un arreglo multidimensional bidimensional, debemos especificar los dos índices; por convención, el primero identifica la fila del elemento y el segundo la columna del elemento.
- En general, un arreglo con m filas y n columnas se conoce como arreglo de m por n.
- Para acceder a cualquier demento en un arreglo bidimensional se usa el nombre de un elemento de la forma a[fila][columna]; a es el nombre del arreglo, fila y columna son las índices que identifican de manera única a la fila y columna, respectivamente, de cada elemento en a.
- Los arreglos multidimensionales se mantienen como arreglos de arreglos.

EJEMPLO 9.

Archivo InicArreglo3.html:

```
<!-- Documento de HTML5 que muestra la inicialización de un arreglo multidimensional. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8">
            <title>Arreglos multidimensionales</title>
            <link rel = "stylesheet" type = "text/css" href = "estilo.css">
            <script src = "InicArreglo3.js" ></script>
      </head>
      <body>
            <h2>Valores en arreglo1 por fila</h2>
            <div id = "salida1"></div>
            <h2>Valores en arreglo2 por fila</h2>
            <div id = "salida2" ></div>
      </body>
</html>
```

Archivo InicArreglo3.js:

```
// Inicialización de arreglos multidimensionales.
function iniciar(){
      var arreglo1 = [ [ 1, 2 , 3 ],      // fila 0
                  [ 4, 5, 6 ] ];    // fila 1
      var arreglo2 = [ [ 1, 2 ],    // fila 0
                  [ 3 ],            // fila 1
                  [ 4, 5, 6 ] ];    // fila 2
      imprimirArreglo( "Valores en arreglo1 por fila", arreglo1,
            document.getElementById( "salida1" ) );
      imprimirArreglo( "Valores en arreglo2 por fila", arreglo2,
            document.getElementById( "salida2" ) );
} // fin de la función iniciar
// mostrar el contenido del arreglo
function imprimirArreglo( encabezado, elArreglo, salida ){
```

```
      var resultados = "";
      // itera a través del conjunto de arreglos unidimensionales
      for ( var fila in elArreglo ){
            resultados += "<ol>";
            // itera a través de los elementos de cada arreglo unidimensional
            for ( var columna in elArreglo [ fila ] ){
                  resultados += "<li>" + elArreglo[ fila ] [ columna ] + "</li>";
            } // f in de for interno
            resultados += "</ol>"; // fin de lista ordenada
      } // fin de for externo
      salida.innerHTML = resultados;
} // fin de la función imprimirArreglo
window.addEventListener( "load", iniciar, false );
```

Archivo estilo.css:

```
img         { margin-right: 10px; }
table       { width: 200px;
                  border-collapse:collapse;
                  background-color: lightblue; }
table, td, th     { border: 1px solid black;
                  padding: 4px;
                  margin-top: 20px; }
th          { text-align: left;
                  color: white;
                  background-color: darkblue; }
li          { display: inline; padding: 4px }
```