# TEMA 4.- ¿CÓMO ORGANIZAR DATOS EN JAVASCRIPT?

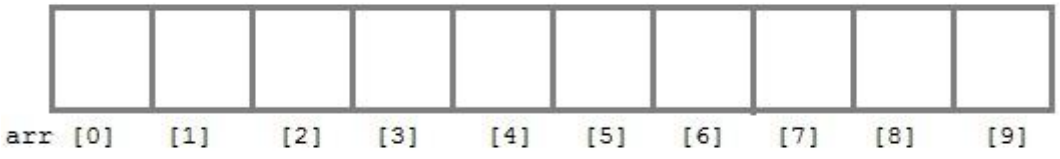
Anteriormente hemos tratado el tema de cómo crear variables y almacenar valores simples en ellas, ahora vamos a utilizar estructuras de datos especiales, que nos servirán para almacenar **información más compleja** que la que se puede almacenar en variables simples. De todas las estructuras de datos que ofrece este lenguaje como pueden ser (listas, pilas, colas, árboles, conjuntos,…), el array es la estructura más utilizada por su sencillez y versatilidad.

En el siguiente ejemplo se puede ver que puede surgir la necesidad de almacenar un número enorme de elementos, como por ejemplo los productos en venta en un supermercado:

```
var product1 = "Manzanas";
var product2 = "Leche";
var product3 = "Peras";
var product4 = "Naranjas";
var product5 = "Cereales";
...
var product180 = "Salsa barbacoa";
```

Hasta ahora no contamos con ninguna estructura de datos que nos permita almacenar toda esta información, sólo contamos con la posibilidad de almacenar cada uno de los datos en una variable simple. Por este motivo, nos surge la necesidad de utilizar estructuras de datos más complejas, como el **array**.

El **array** es una estructura que contiene un conjunto ordenado de valores relacionados, es decir una variable en la que se pueden **introducir varios valores**. Desde el punto de vista lógico, una matriz se puede ver como un conjunto de elementos ordenados en una fila o varias filas y columnas, en función del número de **dimensiones** que tenga.




Los arrays se denominan **vectores** en caso de que tengan una sola dimensión (como en la imagen anterior) y **matrices** en caso de que tengan varias dimensiones.

Los arrays son una estructura de datos, adecuada para situaciones en las que el acceso a los datos se realiza de forma aleatoria e impredecible. Por el contrario, si los elementos pueden estar ordenados y se va a utilizar acceso secuencial existen otras estructuras más adecuadas, como una lista, ya que esta estructura puede cambiar de tamaño fácilmente de forma dinámica durante la ejecución de un programa.

Los arrays nos permiten guardar un gran número de elementos y también nos permite acceder a ellos de manera independiente. Cada elemento es referenciado por la posición que ocupa dentro del array. Dichas posiciones se llaman **índices** y siempre son correlativos. La forma más común de indexado de un array es la llamada **indexación base-cero(0)** en la que el primer elemento del array tendrá el índice 0.

**¿Quieres saber más?**

Más información sobre tipos de estructuras de datos:
http://es.wikipedia.org/wiki/Estructura_de_datos

## ¿Cómo se crea un Array en JavaScript?

La declaración de un array en JavaScript se realiza de la siguiente forma:

- La palabra clave *var*.
- El nombre del array.
- El operador de asignación.
- La palabra clave para la creación de objetos *new*.
- El constructor *Array*.
- El paréntesis final

```
var arrayName = new Array();

var arrayName = new Array(numberOfElements);
```

Una vez declarado el array se puede comenzar el proceso de **inicializar el array** con los elementos que contendrá:

```
arrayName[index] = elementValue;
```

También es posible inicializar el array a la vez que se declara mediante el paso de los elementos que va a contener utilizando el **constructor**:

```
var products = new Array('Manzanas', 'Leche', 'Peras');
```

Se puede inicializar un Array de un determinado tamaño previo pero vacío, es decir en todos sus índices se almacenará **'null'**, de la siguiente forma:

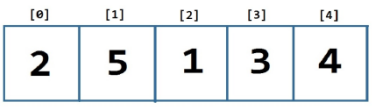
```
var products = new Array(20);
```

Otra forma de asignar elementos a las diferentes posiciones del array sería la siguiente:

```
var products = new Array();
products[0] = 'Manzanas';
products[1] = 'Leche';
products[2] = 'Peras';
```

**Otra forma diferente de inicializar el array**, con elementos asignados desde el inicio sería la siguiente:

```
products = ['Manzanas', 'Leche', 'Peras'];
```




Además de los tipos de array que hemos mencionado hasta ahora existe el **array mixto** (o también llamado **objeto literal**), en el que las posiciones son referenciadas con índices de tipo texto o números sin ningún criterio concreto. Si las posiciones índice están definidas por un texto, para acceder a ellas lo haremos utilizando su nombre y no su número (en este caso si usamos el número nos daría una posición undefined). El formato de creación sería: nombrearray = { "indice1" : valor , indice2 :

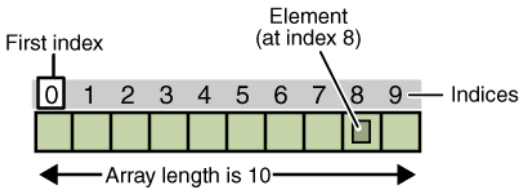
"valor" , …} (fíjate que en este caso para definir el array de tipo mixto tendremos que hacerlo comenzando y terminando con **llaves { } )**. Por ejemplo:

```
var data = {"number": 42, "month" : "June", "name" : "John", 3.14 : "PI"};
```

Si queremos obtener el tamaño que tiene el array, se puede realizar usando la propiedad ***length*** del objeto array, basándonos en el anterior ejemplo, el array *productos* tendrá tamaño *20*.

```
var size = products.lenght; //nos devolverá el valor 20
```

A diferencia de otros lenguajes de programación, en JavaScript hacer el dimensionamiento previo del array no nos dará ningún tipo de ventaja especial, ya que podremos asignar un valor a cualquier posición del array en cualquier momento, esté o no definida previamente. La propiedad length **se ajustará automáticamente al nuevo tamaño del array**. Por ejemplo podríamos hacer una asignación tal como products[33] y eso que nuestro array es de 20 posiciones. En este caso no ocurrirá ningún problema, ya que la propiedad length del array se ajustará automáticamente para poder almacenar la posición 33, con lo que la nueva longitud del array será 34, ya que la primera posición del array es la [0] y la última es la [33], tendremos por lo tanto 34 elementos en el array.




## ¿Cómo organiza JavaScript internamente la información?

JavaScript nos ofrece una gran **versatilidad** a la hora de almacenar y manipular los datos en los arrays ya que permite almacenar diferentes tipos de dato en cada posición del array. Por ejemplo, nos permite tener un array de arrays, proporcionando la equivalencia de arrays multidimensionales, pero adaptado a los tipos de datos que necesite nuestra aplicación.

JavaScript emplea un montón de arrays internamente para **gestionar los objetos HTML que aparecen en una web**. Si un documento contiene 10 enlaces, el navegador

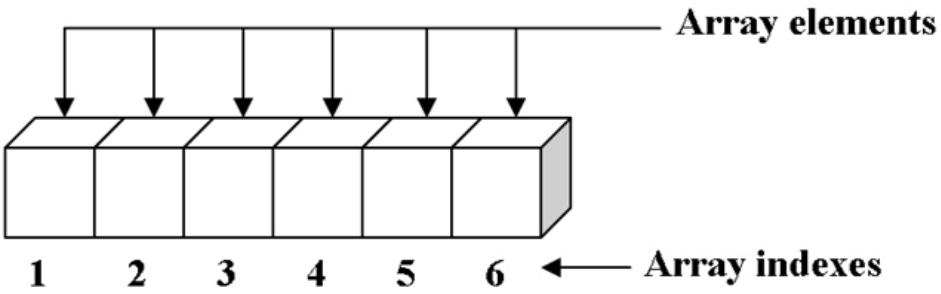
mantiene una tabla con todos esos enlaces, y se podrá acceder a cada uno de esos enlaces mediante un array de enlaces, donde el primer enlace se encontrará en el índice 0. Para acceder al primero de los enlaces se utilizará el nombre del array seguido del índice entre corchetes: document.links[0].

Como hemos visto en la unidad anterior, existen colecciones dentro del objeto document y cada una de esas colecciones (**anchors[], forms[], links[] e images[]**) será un array que contendrá las referencias de todas las anclas, formularios, enlaces e imágenes del documento.

## ¿Cómo se utiliza un Array?

Existen múltiples formas de **obtener los datos almacenados en los arrays**, un ejemplo simple puede ser el siguiente:

```
var solarSystem = new Array();
solarSystem[0] = "Mercury;
solarSystem[1] = "Venus";
solarSystem[2] = "Earth";
solarSystem[3] = "Mars";
solarSystem[4] = "Jupiter";
solarSystem[5] = "Saturn";
solarSystem[6] = "Uranus";
solarSystem[7] = "Neptune";
```




Esta forma sería la que utilizaríamos para acceder a una posición concreta de un array, aunque normalmente se utilizan bucles para recorrer el array independientemente de cuál sea su tamaño, ya que resulta en instrucciones mucho más limpias y eficientes que el acceso independiente a cada uno de los datos que se daba en el ejemplo anterior. A continuación se listan dos diferentes formas de recorrer un array:

```
for (i=0; i<solarSystem.length;i++)
      document.write(solarSystem[i] + "<br/>");
```

```
var i=0;
while (i < solarSystem.length)
{
      document.write(solarSystem[i] + "<br/>");
      i++;
}
```

Para recorrer un array mixto ser podría hacer de la siguiente forma:

```
//Dato el siguiente array mixto:
var data = { "number": 42, "month" : "June", "name" : "John", 3.14 : "PI"
};

// Imprimir todos los nombres de las posiciones del array mixto.
for (eachData in data)
      document.write(eachData +"<br/>");

//salida:
 > number
 > month
 > name
 > 3.14
```

```
// Imprimir todos los nombres de los contenidos de las posiciones      del
array mixto.
for (eachData in data)
      document.write(data[eachData]+"<br/>");

//salida:
 > 42
 > June
 > John
 > PI
```

## ¿Cómo se eliminan elementos de un Array?

Del mismo modo que se pueden insertar valores en un array, es posible eliminar cualquier valor del array. La forma más elegante es asignar el valor a *null*, pero también se podría asignar una cadena vacía *“”*.

Otra forma, para las versiones más recientes de los navegadores, es utilizar el operador *delete* que es una función de los arrays que permite explícitamente eliminar un elemento. Es importante tener en cuenta al usar este operador que al eliminar un elemento se perderá el valor que tenga almacenado en ese índice, pero la longitud del array será el mismo, es decir, ese índice continuará existiendo pero el valor se habrá borrado.

```
solarSystem[5]; // resultado: Jupiter
solarSystem.length; // resultado: 8
delete solarSystem[5]; //se elimina el planeta Jupiter
    solarSystem.length; // resultado: 8
    solarSystem[5]; // resultado: undefined
```

Como decíamos anteriormente el operador delete sólo elimina el dato, pero el array sigue teniendo el mismo tamaño, existe otro método llamado *splice* que permite eliminar uno de los elementos, incluida su posición, este método se comentará en el siguiente apartado.

## ¿Qué propiedades tienen los arrays en JavaScript?

El objeto Array de JavaScript cuenta con una serie de propiedades que nos permiten obtener información sobre los arrays más allá de la información que el usuario almacene en el array. Estas propiedades, que algunas ya las hemos visto, son las siguientes:

| Propiedad | Descripción | Utilización |
|---|---|---|
| constructor | Devuelve la función que creó el prototipo del objeto array. | var solarSystem = new Array(); |
| length | Establece o devuelve el número de elementos en un array. | solarSystem.length |
| prototype | Te permite añadir propiedades y métodos a un objeto. | Array.prototype.nombre_método |

Hasta ahora sólo hemos utilizado la propiedad *length* de los arrays para obtener el valor del tamaño del array pero es necesario recalcar que además de obtener el tamaño, es posible también establecer el tamaño de la siguiente manera: *solarSystem.length = 12*. A partir de este momento, el tamaño del array solarSystem será 12.

**¿Quieres saber más?**

Para aprender más sobre los prototipos, que son muy utilizados en algunas librebrías de JavaScript se puede consultar en el siguiente enlace: https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array/prototype

## ¿Qué métodos ofrecen los arrays en JavaScript?

| Método | Descripción |
|---|---|
| concat() | Une dos o más arrays, y devuelve una copia de los arrays unidos. |
| join() | Une todos los elementos de un array en una cadena de texto. |
| pop() | Elimina el último elemento de un array y devuelve ese elemento. |
| push() | Añade nuevos elementos al final de un array, y devuelve la nueva longitud. |
| reverse() | Invierte el orden de los elementos en un array. |
| shift() | Elimina el primer elemento de un array, y devuelve ese elemento. |
| slice() | Selecciona una parte de un array y devuelve el nuevo array. |
| sort() | Ordena los elementos de un array. |
| splice() | Añade/elimina elementos a un array. |
| toString() | Convierte un array a una cadena y devuelve el resultado. |
| unshift() | Añade nuevos elementos al comienzo de un array, y devuelve la nueva longitud. |
| valueOf() | Devuelve el valor primitivo de un array. |

A continuación se muestran algunos ejemplos de los métodos:

**concat()**

```
<script type="text/javascript">
var teamsA = new Array("Manchester United", "Chelsea", "Liverpool");
var teamsB = new Array("Arsenal", "Stoke City");
var allTeams = teamsA.concat(teamsB);
document.write("Equipos que juegan la copa: " + allTeams);
</script>
```

**join()**

```
<script type="text/javascript">
      var pizzas = new Array("Carbonara", "Quattro_Stagioni", "Diavola");
      document.write(pizzas.join(" - "));
</script>
```

### pop()

```
<script type="text/javascript">
var prizes = new Array("Car", "1000 Euros", "JavaScript Manual");
var thirdPrize = prizes.pop();
document.write("El tercer premio es: " + thirdPrize + "<br />");
document.write("Quedan los siguientes premios: " + prizes);
</script>
```

### push()

```
<script type="text/javascript">
var pizzas = new Array("Carbonara", "Quattro_Stagioni", "Diavola");
var newPizzas = pizzas.push("Margherita", "Boscaiola");
document.write("Número de pizzas disponibles: " + newPizzas + "<br />");
document.write(pizzas);
</script>
```

### reverse()

```
<script type="text/javascript">
      var fruits = ["Banana", "Orange", "Apple"];
      document.write(fruits.reverse());
      //Se imprimirá: Apple,Orange,Banana
</script>
```

### shift ()

```
<script type="text/javascript">
var pizzas = new Array("Carbonara", "Quattro_Stagioni", "Diavola");
var removedPizza = pizzas.shift();
document.write("Pizza eliminada de la lista: " + removedPizza + "<br/>");
document.write("Nueva lista de pizzas: " + pizzas);
</script>
```

### slice()

```
<script type="text/javascript">
      var fruits = ["Banana", "Orange", "Apple", "Plumb"];
      document.write(fruits.slice(0,1) + "<br/>");
            // imprimirá: Banana
      document.write(fruits.slice(1) + "<br />");
            // imprimirá: Orange,Apple,Plumb
      document.write(fruits.slice(-2) + "<br />");
            // imprimirá: Apple, Plumb
      document.write(fruits + "<br />");
            // imprimirá: Banana,Orange,Apple,Plumb
</script>
```

### sort()

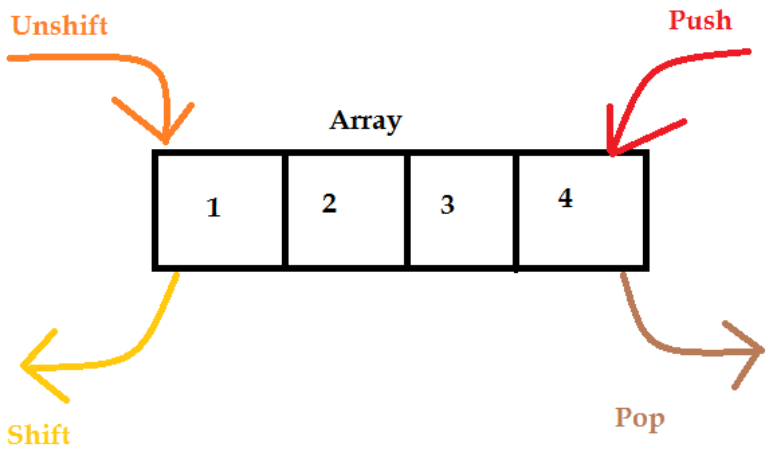
```
<script type="text/javascript">
var surnames = new Array("Pérez", "Guijarro", "Arias", "González");
surnames.sort();
document.write(surnames);
</script>
```

### splice()

```
<script type="text/javascript">
      var cars = new Array("Ferrari", "BMW", "Fiat");
      cars.splice(2,0,"Seat");
      document.write(cars);
</script>
```

### unshift()

```
<script type="text/javascript">
var olympicGamesHosts = new Array("Athens", "Sydney","Atlanta");
var numberOfHosts = olympicGamesHosts.unshift("Beijin");
document.write("Últimas " + numberOfHosts + " sedes olímpicas: " +
olympicGamesHosts);
</script>
```




**¿Quieres saber más?**

Más información y ejemplos sobre el objeto array.

http://www.w3schools.com/jsref/jsref_obj_array.asp

## ¿Cómo se trabaja con arrays multidimensionales?

Si bien es cierto que en JavaScript los arrays son unidimensionales, podemos crear arrays que en sus posiciones contengan otros arrays u otros objetos. Podemos crear de esta forma *arrays bidimensionales, tridimensionales,* etc.

Por ejemplo podemos crear un ***array bidimensional*** de la siguiente forma:

```
//definiendo el array y después asignando arrays en cada uno de sus
índices
var data = new Array();
data[0] = new Array("Cristina","Seguridad",24);
data[1] = new Array("Catalina","Bases de datos",17);
data[2] = new Array("Vieites", "Sistemas informáticos",28);
data[3] = new Array("Benjamin","Redes",26);

//o usando una definición completa:
var datos = [ ["Cristina","Securidad",24], ["Catalina","Bases de
datos",17], ["Vieites","Sistemas Informáticos",28],
["Benjamin","Networks",26] ];
```

Para acceder a un dato en particular de un array multidimensional de dos dimensiones, se requiere que hagamos una **doble referencia** utilizando dos índices. La primera referencia, será a un índice del array principal, y la segunda referencia, a un índice del array almacenado dentro de esa posición del array principal. Esto se hará escribiendo el nombre del array, y entre corchetes cada una de las referencias: **nombreArray[indice1][indice2]**. Siguiendo el mismo sistema se accederá a arrays de 3 dimensiones (**nombreArray[indice1][indice2][indice3]**), de 4 (**nombreArray[indice1][indice2][indice3][indice4]**)..., todo será cuestión de añadir el número de índices necesario. Podemos ver un ejemplo a continuación:

```
document.write("<br/>Quien imparte Bases de Datos? "+data[1][0]); //
Catalina
document.write("<br/>Asignatura de Vieites: "+data[2][1]); // ITs
document.write("<br/>Alumnos de Benjamin: "+data[3][2]); // 26
```

Si queremos **imprimir toda la información del array multidimensional**, tal y como hicimos en el apartado anterior podríamos hacerlo con un bucle *for*:

```
for (i=0;i<datos.length;i++) {
      document.write("<br/>"+data[i][0]+" del módulo de "+data[i][1]+",
tiene "+data[i][2]+" alumnos  en clase.");
}


//El resultado será:
> Cristina del módulo de Seguridad, tiene 24 alumnos en clase.
> Catalina del módulo de Bases de Datos, tiene 17 alumnos en clase.
> Vieites del módulo de Sistemas Informáticos, tiene 28 alumnos en clase.
> Benjamin del módulo de Redes, tiene 26 alumnos en clase.
```