

# La computadora es un mar de datos

*"Con orden y tiempo se encuentra el secreto de hacerlo todo, y de hacerlo bien."*

Pitágoras – Filósofo y matemático

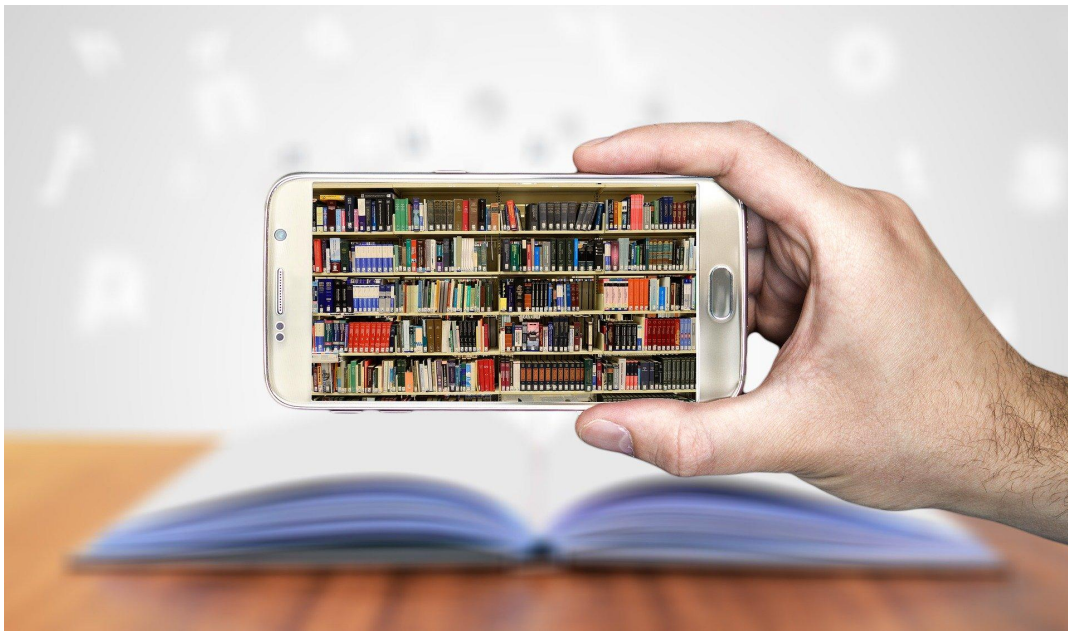


Imagen de [Gerd Altmann](#) en [Pixabay](#)

Es recomendable mantener un orden de las cosas que hacemos en nuestra vida diaria, más allá de las tareas que implementemos. Ser organizado/a tiene sus ventajas. Si manejas una estructura organizacional de tus tareas -como armar listas o calendarizar los eventos- sabrás con certeza qué tienes que hacer y podrás optimizar el tiempo con el que cuentas. Lo mismo sucede con tus archivos de la computadora: ordenarlos en carpetas con un cierto orden lógico (la lógica dependerá de tí mismo, de tu equipo u organización para la que trabajas), con nombres que luego puedas ubicar con facilidad, te simplificará mucho los procesos como developer.

Esta práctica de la organización de documentos tendrás que trasladarla al código. Como ya sabes, en programación tenemos distintos sistemas de almacenamiento tales como las [variables](#), que se pueden organizar por tipos. El problema es que a veces con eso no alcanza y tenemos que contar con otras



herramientas. En esta bitácora te encontrarás con una estructura -sí, ¡más compleja!- que te servirá para manipular datos que requerirán de tí una organización mayor.

## Etapa de organización

Si te encuentras por guardar y agrupar una gran cantidad de datos en un programa, una forma de hacerlo sería con variables, en las que se puede almacenar cada elemento de manera individual.

Pero, dado el problema que mencionamos anteriormente, ¿existe otra forma? La respuesta es: SIEMPRE HAY OTRA FORMA. La siguiente pregunta sería: ¿existe otra forma mejor? ¡Claro! Puedes usar [estructuras de datos](#), ya que es una manera más organizada.

Según la definición del blog [EDteam](#):

*“La estructura de datos es una rama de las ciencias de la computación que estudia y aplica diferentes formas de organizar información dentro de una aplicación, para manipular, buscar e insertar estos datos de manera eficiente.”*

[Array](#) es una [estructura de datos](#) que funciona igual a una variable como un espacio para almacenar información. Pero se diferencia en tanto que **admite almacenar más de un dato, o mejor dicho ¡un conjunto de datos!**

Javascript te permite guardar datos de diferentes tipos, ya sean de tipo booleano o números en un mismo array. Se podría decir que un **array funciona al igual que un archivo que permite guardar cualquier tipo de dato, de manera secuencial y no necesariamente en orden.**

Siguiendo el blog [EDteam](#): *“es recomendable usar arrays cuando el acceso a estos datos se realiza de manera aleatoria”.*

La forma de organización que elijas va a depender de lo que quieras hacer con los datos. Varían según cada lenguaje de programación. Sigamos el consejo del [blog](#) de [Benoit Vallon](#):

*“La clave para recordar es que cada estructura de datos tiene sus propias ventajas y desventajas. No hay ninguno de ellos*



*que pueda vencer a todos los demás, esa es la razón por la cual es importante conocerlos a todos.”*

## ¿Cómo construir un archivo de datos?

Para comenzar, ¿recuerdas el pseudocódigo que te presentamos en la bitácora anterior? Utilicemos el mismo concepto para entender cómo funcionan los arrays.

Imagina que estás situado/a en un e-commerce que se dedica a la venta de vehículos. El sitio web muestra los vehículos que tiene a la venta categorizados por su fabricante, por ejemplo: Chevrolet, Ford, Fiat, Volkswagen y Ferrari.

Podríamos definir una variable para cada uno de los fabricantes:

```
fabricante_1 = Chevrolet  
fabricante_2 = Ford  
fabricante_3 = Fiat  
fabricante_4 = Volkswagen  
fabricante_5 = Ferrari
```

Para definir este conjunto de datos de una manera más óptima y ordenada Javascript nos provee el tipo de variable Array:

```
fabricantes = Chevrolet, Ford, Fiat, Volkswagen, Ferrari
```

Como ves, cada uno de los fabricantes de vehículos se encuentra almacenado en una sola variable del tipo array llamada fabricantes. Nótese la redundancia en la palabra fabricantes: ¡es un buen síntoma! El nombre de la variable indica por sí sola qué valores estoy almacenando en ella.

Los valores almacenados dentro del array son denominados **elementos**. En el ejemplo, puedes observar 1 array con 5 elementos.

Ahora bien ¿es posible acceder a cada uno de los elementos de forma particular para obtener su valor? ¡Por supuesto! **Javascript asigna automáticamente un índice numérico a cada uno de los elementos** empezando desde el 0 e incrementando en 1 cada posición hasta su último elemento.



Siguiendo esta lógica podemos afirmar que el primer elemento de un array es el índice 0 y el último es "total de elementos - 1":

```
fabricantes índice 0 = Chevrolet
```

```
fabricantes índice 2 = Fiat
```

```
fabricantes índice 4 = Ferrari
```

¡Del dicho al hecho! pasemos del pseudocódigo al código y veamos cómo definirás Arrays en Javascript. Tienes 2 opciones:

- Mediante **corchetes**:

```
var fabricantes = ["Chevrolet", "Ford", "Fiat",  
"Volkswagen", "Ferrari"];
```

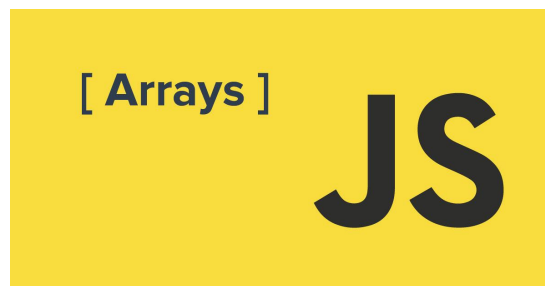
- Mediante **new Array**:

```
var fabricantes = new Array("Chevrolet", "Ford",  
"Fiat", "Volkswagen", "Ferrari");
```

Para acceder a cada uno de los elementos, utilizaremos corchetes con el número del índice:

```
console.log(fabricantes[1]); //imprime Ford
```

## Cinco acciones para hacer con los datos



Javascript permite manipular los Arrays de una forma sencilla con propiedades y métodos. Las **propiedades** contienen un valor, mientras que los **métodos** ejecutan alguna acción específica para proveer un resultado. Nos sirven para:

1. **Conocer la cantidad de elementos.** La propiedad `length`: nos devuelve el valor de cuántos elementos tengo en mi array.

```
console.log(fabricantes.length) //imprime 5
```

2. **Añadir elementos.** Se puede añadir elementos al inicio de un array previamente creado con `unshift`; para añadir un elemento al final del array utilizamos `push`.

```
fabricantes.unshift("Volvo");  
fabricantes.push("Porsche");  
console.log(fabricantes); //Imprime "Volvo", "Chevrolet",  
"Ford", "Fiat", "Volkswagen", "Ferrari", "Porsche"
```

3. **Buscar elementos.** Para buscar un elemento usamos `indexOf` y nos retorna el índice numérico como resultado de la búsqueda.

```
var indice = fabricantes.indexOf("Volkswagen");  
console.log(indice); //imprime 4
```

4. **Eliminar elementos.** Para eliminar el primer elemento de nuestro array utilizamos `shift()`. Si queremos eliminar el último elemento de nuestro array tenemos la función `pop()`.

```
fabricantes.shift(); //Elimina Volvo  
fabricantes.pop(); //Elimina Porsche
```

5. **Ordenar los elementos.** Mediante el método `sort` podemos ordenar los elementos ascendente - descendente.

Ten cuidado, cambiará cada uno de los índices en tu Array.

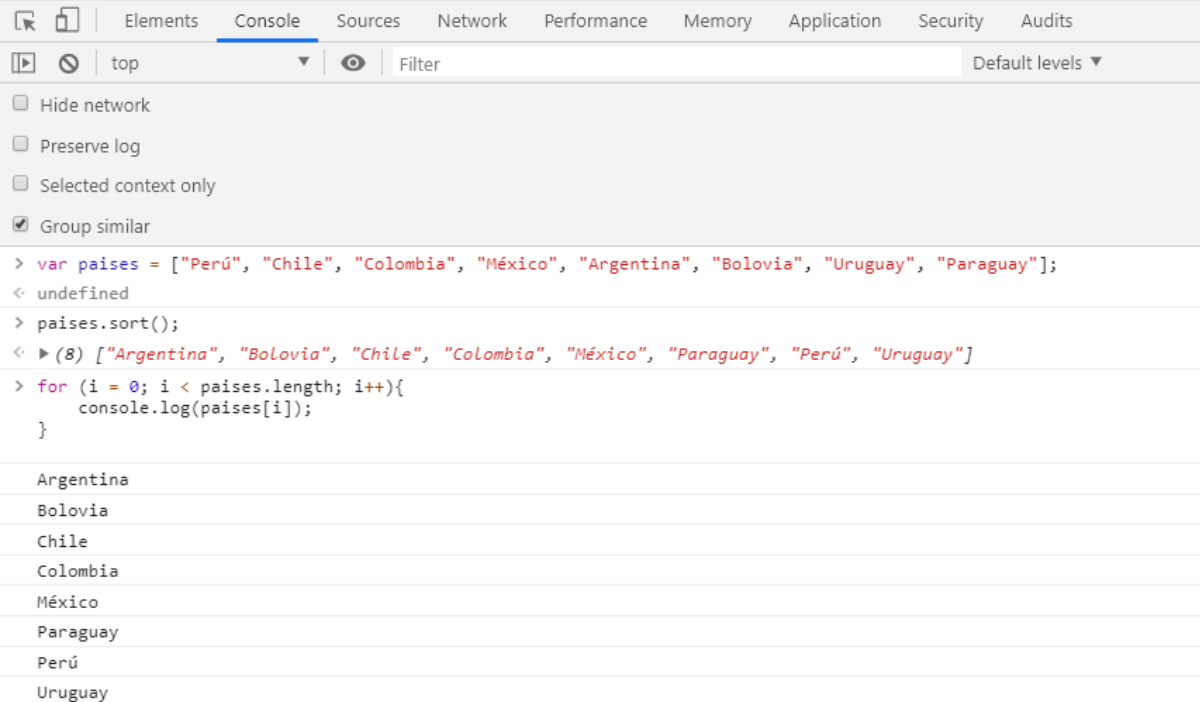
```
fabricantes.sort();  
console.log(fabricantes); //Imprime "Chevrolet",  
"Ferrari", "Fiat", "Ford", "Volkswagen"
```



## Recorre tus Arrays

¡Combina tus conocimientos adquiridos! Utiliza un ciclo para recorrer tus arrays: sabiendo que los índices comienzan en 0 y terminan en "elementos - 1", puedes utilizar una sentencia del tipo `for` para iterar cada uno de los elementos.

Ya no tendrás que preocuparte por la cantidad de elementos que contenga tu Array, porque tu ciclo dará la cantidad de vueltas necesarias hasta llegar al último elemento.



```
> var paises = ["Perú", "Chile", "Colombia", "México", "Argentina", "Bolivia", "Uruguay", "Paraguay"];
< undefined
> paises.sort();
< ▶ (8) ["Argentina", "Bolivia", "Chile", "Colombia", "México", "Paraguay", "Perú", "Uruguay"]
> for (i = 0; i < paises.length; i++){
  console.log(paises[i]);
}
```

Argentina  
Bolivia  
Chile  
Colombia  
México  
Paraguay  
Perú  
Uruguay

## Live Coding

¿Te quedaste con ganas de más? ¿Prefieres escuchar más que leer para aprender? ¿Quieres ver al director de la carrera - Daniel Segovia - codear lo que te contamos en la bitácora? Aquí te dejamos un video:

[JavaScript - Array](#)



## ¡Prepárate para el próximo encuentro!

### Profundiza



[Manejo de Arrays en JavaScript](#)



[JavaScript Arrays - tips, tricks and examples](#)

### Challenge



En la bitácora vimos cómo recorrer un array con la sentencia `for`.  
Te proponemos crear tu primer `array` de frutas y mostrar cada una de ellas mediante un ciclo tipo `while`.

