



# Arrays y Objetos

Arreglos

***Utilizar estructuras de tipo  
arreglo y sentencias  
iterativas para el control del  
flujo de un algoritmo que  
resuelve un problema simple  
acorde al lenguaje  
Javascript.***

- Unidad 1:  
Introducción al lenguaje JavaScript
- Unidad 2:  
Funciones y Ciclos
- Unidad 3:  
Arrays y Objetos
- Unidad 4:  
APIs



Te encuentras aquí



## ¿Qué aprenderás en esta sesión?

- *Identifica las características y utilidad de una estructura de tipo arreglo para la elaboración de un algoritmo.*

Para asignarle un evento  
a un elemento, ¿qué  
método debería usar?



Según lo estudiado, ¿qué es  
una expresión regular?



¿Qué hace el modificador 'i'  
en las expresiones  
regulares?



**`/* Estructura y utilidad de un Array */`**

# Estructura y utilidad de un Array

## ¿Qué es un Array?

- Un array o arreglo en JavaScript es un tipo de dato que nos permite almacenar de manera ordenada un conjunto de elementos, a los que podemos acceder desde una sola variable.



Ejemplo de un array basado en una analogía de un estacionamiento.



# Declaración y asignación de Arrays

- Para definir arrays en JavaScript puedes utilizar los editores de código [JSFiddle](#), [CodePen](#), [CodeSandbox](#) o un archivo HTML con código JavaScript en su interior.
- Los arrays en JavaScript son comúnmente definidos entre corchetes ("[" ]"), indicando en su interior los valores que necesitamos listar, aunque también existen otras formas de hacerlo.

```
let amigos = ["Erick", "Cristian", "Max", "Claudia"];  
console.log(amigos);
```

```
Array(4) [ "Erick", "Cristian", "Max", "Claudia" ]
```

# Demostración - “Lista de películas”



# Ejercicio guiado

## *Lista de películas*

Crear una lista de películas utilizando un array, las películas a agregar son: “It 2, Rambo 3, Halloween, Shaft”, sigamos los siguientes pasos:

- **Paso 1:** Crear e inicializar una variable para que contenga nuestra lista de películas.
- **Paso 2:** Dentro de la notación de corchetes y separadas por comas, se escribe cada película de la lista como string, es decir, dentro de las comillas dobles o sencillas.

```
let peliculas = ["It 2", "Rambo 3", "Halloween", "Shaft"];  
console.log(peliculas);
```



# Ejercicio guiado

## *Lista de películas*

- **Paso 3:** Ejecutar el código en la consola del navegador web, obteniendo el siguiente resultado:

```
Array(4) [ "It 2", "Rambo 3", "Halloween", "Shaft" ]
```

Como puedes ver, no sólo es posible crear arrays con la notación anterior, sino que también, podemos definirlos a través del objeto `Array()`. Para la lista de películas recién creada, podemos utilizar el objeto **Array** de la siguiente forma.

```
let peliculas = new Array("It 2", "Rambo 3", "Halloween", "Shaft");
```

```
var peliculas = Array("It 2", "Rambo 3", "Halloween", "Shaft");
```

# Arrays vacíos

- Existen casos en los que se necesita inicializar un array vacío, ya que se le asignan valores en otro momento. Estos Arrays son dinámicos o dependen de alguna otra rutina.

```
var peliculas = [];
```

- Por si te lo preguntas, al intentar visualizar el resultado de un array vacío dentro de un alert, se mostrará un mensaje en blanco en el navegador

# Arrays con distintos tipos de datos

Los arrays no tan sólo aceptan cadenas de textos o strings como revisamos en los casos anteriores, sino que también admiten otros tipos de datos y objetos.

- Lista de valores numéricos:

```
var pares = [2, 4, 6, 8, 10];
```

- Lista de valores decimales:

```
var decimales = [2.1, 4.4, 3.1, 7.7];
```

- Lista de valores booleanos:

```
var booleanos = [false, false, true];
```

- Lista de objetos dentro de un arreglo:

```
var objetos = [{nombre: 'juan'}, {nombre: 'carlos'}];
```

# Demostración - “Arrays con distintos tipos de datos”



## Ejercicio guiado

### *Arrays con distintos tipos de datos*

Crear un array que contenga distintos tipos de datos, en el espacio 0 debe ir almacenado tu primer nombre, en el siguiente espacio tu primer apellido, en el siguiente espacio las variables edad y fecha de nacimiento como objeto, y para finalizar indicar si eres hijo único mediante operadores booleanos.

- **Paso 1:** Declarar la variable que contendrá el arreglo con los corchetes pero vacía.

```
var datosPersonales = [,,{,},,];
```



# Ejercicio guiado

## Arrays con distintos tipos de datos

- **Paso 2:** Creado y distribuido el espacio en el arreglo, ahora agregamos los datos correspondientes.

```
var datosPersonales = ['juan', 'duran', {edad: 34, nacimiento: 1985}, false];  
console.log(datosPersonales);
```

- **Paso 3:** Al ejecutar el código el resultado sería:

```
0: "juan"  
1: "duran"  
2: {edad: 34, nacimiento: 1985}  
3: false
```



***/\* Referenciando Arrays \*/***

# Referenciando Arrays

- Para acceder a un valor específico de un array, podemos valernos de su índice asignado para realizar dicha tarea.
- Por ejemplo, para acceder al segundo elemento del array de la variable películas o al valor Rambo 3, podemos realizarlo de la siguiente manera:

**Considera que los índices parten desde el 0 hasta el número total de elementos menos 1.**

```
var peliculas = [  
  "It 2", // posición 0  
  "Rambo 3", // posición 1  
  "Halloween", // posición 2  
  "Shaft" // posición 3  
];  
console.log(peliculas[1]);
```

# Demostración - “Accediendo a los valores de un arreglo”



# Ejercicio guiado

## Accediendo a los valores de un arreglo

Acceder y mostrar los datos con el nombre de: "Vue JS" y BackEnd: "Node JS", del siguiente arreglo:

```
var programa_web = [  
  "JavaScript",  
  "React JS",  
  "Vue JS",  
  "Angular JS",  
  {BackEnd: "Node JS"}  
];
```



# Ejercicio guiado

## Accediendo a los valores de un arreglo

- **Paso 1:** Para acceder a la posición que contiene el valor de “Vue JS”, debemos referenciar la variable que contiene el arreglo con el número dos (2).
- **Paso 2:** Para acceder a la variable “BackEnd” con el valor de “Node JS”, se debe referenciar la variable programa\_web, que contiene el arreglo, con la posición 4. Quedando de la siguiente manera:

```
var programa_web = [  
  "JavaScript", // posición 0  
  "React JS", // posición 1  
  "Vue JS", // posición 2  
  "Angular JS", // posición 3  
  {BackEnd: "Node JS"} // posición  
  4  
];  
  
console.log(programa_web[2]);  
console.log(programa_web[4]);
```

**/\* Arreglos multidimensionales o  
matrices \*/**

# Arreglos multidimensionales o matrices

- Una matriz es un conjunto ordenado de elementos (igual que un arreglo).
- JavaScript no provee arrays multidimensionales (matrices), como otros lenguajes de programación pero puedes crear una matriz definiendo un array.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Fuente: [Wikipedia](https://es.wikipedia.org/wiki/Matriz)



# Arreglos multidimensionales o matrices

- Para crear una matriz es necesario reemplazar cada elemento de un array por otros arrays, de la siguiente forma

```
var array = [  
  "elemento1",  
  "elemento2",  
  "elemento3"  
]
```

```
var matriz = [  
  ["elemento1"],  
  ["elemento2"],  
  ["elemento3"]  
];  
console.log(matriz[0])  
console.log(matriz[1])  
console.log(matriz[2])
```

```
(0) ["elemento1"]  
(1) ["elemento2"]  
(2) ["elemento3"]
```

# Demostración - “Matrices”



# Ejercicio guiado

## Matrices

Realizar una matriz de 3x3 con números enteros desde el 1 hasta el 9, tres números por cada arreglo interno. En este caso, cada espacio del arreglo original contendrá otro arreglo dentro de él con los valores de la matriz, es decir, tendremos arreglos anidados dentro de la variable que ya es un arreglo. Resolvamos este ejemplo:

- **Paso 1:** Construir el arreglo principal y lo llamaremos matriz3x3, dejando planteados los tres arreglos internos:

```
var matriz3x3 = [  
  [elementos1],  
  [elementos2],  
  [elementos3],  
]
```

# Ejercicio guiado

## Matrices

- **Paso 2:** Al plantear el arreglo principal con los internos, se agregan los números correspondientes a cada arreglo interno, es decir, tres números por arreglos para poder formar la matriz de 3x3.

```
var matriz3x3 = [  
  [1,2,3],  
  [4,5,6],  
  [7,8,9],  
];
```

# Ejercicio guiado

## Matrices

- **Paso 3:** Mostramos el arreglo realizado en la consola del navegador web mediante un `console.log`:

```
console.log(matriz3x3);
```

- **Paso 4:** Al ejecutar y mostrar el código anterior, el resultado sería:

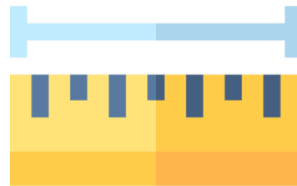
```
0: Array(3) [ 1, 2, 3 ]  
1: Array(3) [ 4, 5, 6 ]  
2: Array(3) [ 7, 8, 9 ]
```



***/\* Largo de Arrays y su manipulación \*/***

# Largo de Arrays y su manipulación

- Los arrays contienen una propiedad llamada **length**, la cual permite acceder a la cantidad de elementos que contienen.
- Se debe tener cuidado con el uso de la propiedad length, ya que de manera explícita (o intencional) puede contener un valor distinto al esperado, dado que su valor podemos reasignarlo, lo que conlleva incluso a eliminar o expandir un array.



# Demostración - “Lista de instrumentos musicales”





# Ejercicio guiado

## *Lista de instrumentos musicales*

Crear un nuevo arreglo con una lista de instrumentos musicales: “Guitarra Eléctrica, Piano de Cola, Violín, Trompeta”. En base a esta colección, modificaremos la cantidad de elementos disponibles, su largo y agregaremos nuevos elementos al arreglo (“Guitarra Clasica, Chelo”) en las posiciones 6 y 7 respectivamente.

- **Paso 1:** Crear la estructura básica del arreglo, es decir, inicializar la variable y dejar los corchetes para agregar los elementos dentro del arreglo.

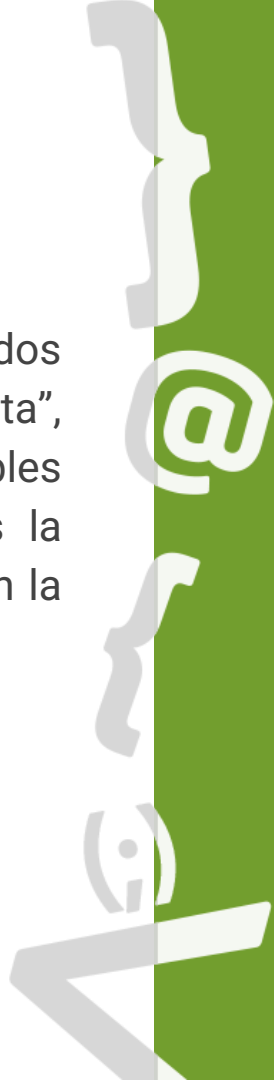
```
let instrumentos = [];
```

# Ejercicio guiado

## *Lista de instrumentos musicales*

- **Paso 2:** Es momento de agregar los elementos dentro del arreglo planteados en el enunciado, es decir: "Guitarra Eléctrica, Piano de Cola, Violín, Trompeta", por lo que tienen que ir dentro de los corchetes con comillas dobles o simples para dejarlos como string y separados por comas. Luego, mostramos la cantidad de elementos dentro del arreglo mediante la instrucción `length` en la consola del navegador:

```
let instrumentos = ["Guitarra Eléctrica", "Piano de  
cola", "Violín", "Trompeta", ];  
console.log(instrumentos);  
console.log(instrumentos.length);
```



# Ejercicio guiado

## *Lista de instrumentos musicales*

- **Paso 3:** Al ejecutar el código anterior, el resultado en la consola del navegador sería:

```
Array(4) [ "Guitarra Eléctrica", "Piano de cola", "Violín", "Trompeta" ]  
4
```

- **Paso 4:** Para poder modificar el largo original del arreglo, podemos indicarle mediante la propiedad `length`, la nueva cantidad de elementos totales del arreglo y mostramos en la consola el arreglo para poder visualizar la modificación realizada:

```
instrumentos.length = 2;  
console.log(instrumentos);  
console.log(instrumentos.length);
```

# Ejercicio guiado

## *Lista de instrumentos musicales*

- **Paso 5:** Al ejecutar el código anterior, el resultado en la consola del navegador sería:

```
Array [ "Guitarra Eléctrica", "Piano de cola" ]  
2
```

- **Paso 6:** Ahora para agregar los dos nuevos elementos que se solicitan en el enunciado, utilizamos la variable del arreglo y le indicamos en cual posición se desea agregar cada elemento.

```
instrumentos[6] = "Guitarra Clásica";  
instrumentos[7] = "Chelo";  
console.log(instrumentos);  
console.log(instrumentos.length);
```



# Ejercicio guiado

## *Lista de instrumentos musicales*

- **Paso 7:** Al ejecutar el código anterior, el resultado en la consola del navegador sería:

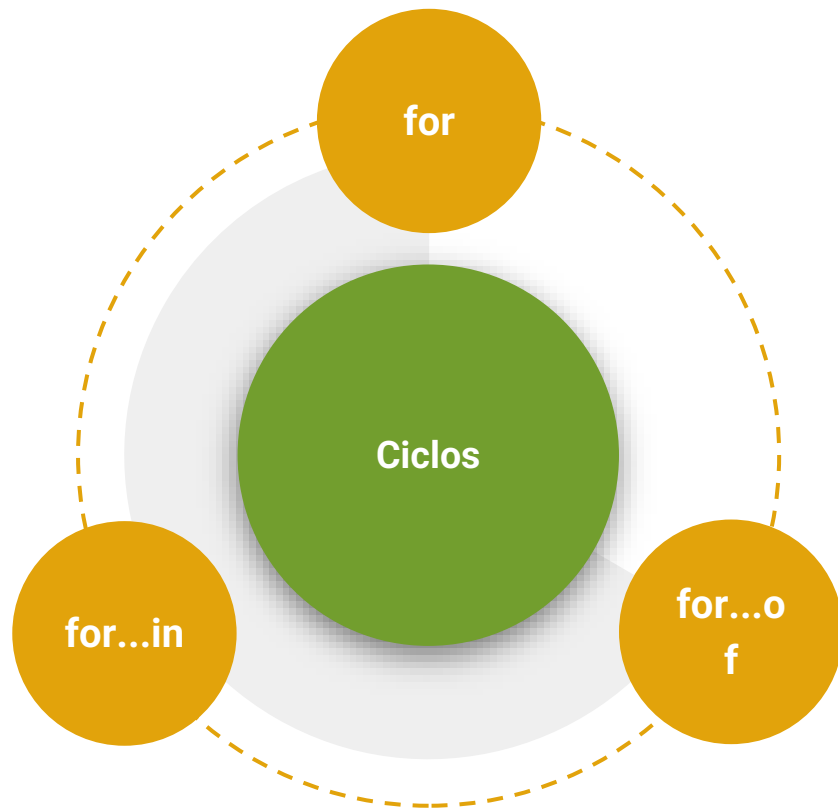
```
Array(8) [ "Guitarra Eléctrica", "Piano de  
cola",,,, "Guitarra Clásica", "Chelo" ]  
8
```



**/\* Recorriendo los valores de un Array \*/**

# Recorriendo los valores de un Array

- Una de las tareas más usuales dentro de la programación en JavaScript es recorrer arrays o colecciones de datos, y poder manipular cada uno de sus elementos.
- Para esta tarea, tenemos a disposición rutinas que nos permiten iterar, como las siguientes:



# Recorriendo los valores de un Array

## *Ciclo for*

Para recorrer cada una de las películas dentro del ejemplo de arreglo anterior, primero escribimos la palabra `for`, la cual entre paréntesis se compone de 3 bloques separados por punto y coma:

- Definición de una variable
- Evaluación de una variable
- Modificación de una variable

```
var peliculas = ["It 2", "Rambo 3", "Halloween", "Shaft"];  
for (var i = 0; i < peliculas.length; i++) {  
    document.write(peliculas[i] + '|');  
};
```



# Demostración - “Recorriendo un arreglo con un ciclo for”



# Ejercicio guiado

## Recorriendo un arreglo con un ciclo for

Mostrar cada uno de los elementos que componen el arreglo indicado, utilizando la estructura repetitiva “for” según el siguiente código:

```
var usuarios = [  
  'Manuel',  
  {nombre: 'Juan', edad: 34},  
  30,  
  {nombre: 'Rodrigo', edad: 55},  
  true,  
  {nombre: 'Paola', edad: 34},  
  'Pedro',  
  ['Maria', 'Eliana']  
];
```



# Ejercicio guiado

## Recorriendo un arreglo con un ciclo for

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.
- **Paso 2:** Copiamos la variable con el arreglo del enunciado y luego se crea la estructura del ciclo for implementando la instrucción “length” para tener disponible la longitud del arreglo e iterar en base a ese número:

```
var usuarios = [  
  'Manuel',  
  {nombre: 'Juan', edad: 34},  
  30,  
  {nombre: 'Rodrigo', edad: 55},  
  true,  
  {nombre: 'Paola', edad: 34},  
  'Pedro',  
  ['Maria', 'Eliana']  
];  
  
for (var i = 0; i < usuarios.length;  
i++) {  
};
```

# Ejercicio guiado

## Recorriendo un arreglo con un ciclo for

- **Paso 3:** Dentro de la estructura del ciclo for, se mostrará cada uno de los elementos encontrados, accediendo mediante la referencia dispuesta por la variable del ciclo for que lleva el conteo de las iteraciones, en este caso "i".

```
for (var i = 0; i < usuarios.length; i++) {  
    console.log(usuarios[i]);  
};
```

- **Paso 4:** Ejecutamos el código anterior en nuestro navegador mediante el archivo index.html y entramos a la consola para poder observar el resultado

# Recorriendo los valores de un Array

## *For ... in*

- La estructura for...in es utilizada para realizar ciclos “finitos” sobre objetos (Object), es decir, se recorren de inicio a fin sin necesidad de un inicializador y no existe una condición de salida, sino que es implícita y corresponde al final de los elementos iterables.

```
for (variable in objeto) {  
    // ...  
}
```

# Demostración - “Recorriendo objetos con for... in”



# Ejercicio guiado

## Recorriendo objetos con for... in

Se solicita recorrer, acceder y mostrar cada una de las llaves (variables) para un objeto que represente a una persona con nombre, edad y una función de saludo, este objeto es:

```
let persona = {  
  nombre: 'Marcelo Salas',  
  edad: 11,  
  saludar: () => {  
    return "Buena matador!";  
  }  
};
```



# Ejercicio guiado

## Recorriendo objetos con for... in

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.
- **Paso 2:** En el archivo script.js, se debe armar la estructura for...in, la cual, servirá para recorrer el objeto en cuestión, accediendo a la llave y mostrando en la consola del navegador cada propiedad o llave que tenga el objeto, más no el valor que acompaña a cada propiedad, por consiguiente, el código del for...in quedaría:

```
let persona = {  
  nombre: 'Marcelo  
Salas',  
  edad: 11,  
  saludar: () => {  
    return "Buena  
matador!";  
  }  
}  
  
for (propiedad in  
persona) {  
  
  console.log(propiedad);  
}
```



# Recorriendo los valores de un Array

## *For... of*

Ya conocimos cómo iterar sobre objetos, de igual manera también es posible hacer ciclos sobre “objetos iterables”, el más conocido y ya visto anteriormente en este curso son los Array. La estructura de **for...of** corresponde a:

```
for (variable of objeto) {  
    // ...  
}
```

# Demostración - “Recorriendo arreglos con for...of”



# Ejercicio guiado

## Recorriendo arreglos con for... of

A partir de un array con las edades de los miembros de una familia de 5 personas, se debe recorrer y mostrar los datos por consola, utilizando tanto for...in como for...of. El arreglo con las edades es el siguiente:

```
let edades = [49, 51, 21, 18, 15];
```

- **Paso 1:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

# Ejercicio guiado

## Recorriendo arreglos con for... of

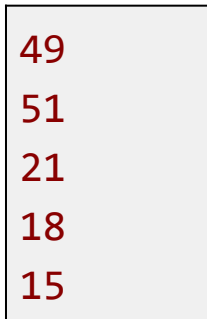
- **Paso 2:** Ahora se debe armar primeramente la estructura for...of, la cual, servirá para recorrer el arreglo en cuestión, accediendo a los valores del arreglo y mostrando el resultado en la consola del navegador, por consiguiente, el código del for...of quedaría:

```
let edades = [49, 51, 21, 18, 15];  
  
for (let edad of edades) {  
  console.log(edad);  
}
```

# Ejercicio guiado

*Recorriendo arreglos con for... of*

- **Paso 3:** Ejecuta el código en el navegador web, por lo que el resultado sería:



49  
51  
21  
18  
15

## Ejercicio guiado

### Recorriendo arreglos con for... of

- **Paso 4:** Ahora como complemento al ejercicio anterior, recorramos el mismo arreglo pero esta vez utilizando for..in, por lo que agregaremos esta última estructura dejando el for...of anterior para comparar ambos resultados en línea:

```
let edades = [49, 51, 21, 18, 15];

for (let otra of edades) {
  console.log(otra);    //
}

for (let otra in edades) {
  console.log(otra);    //
}
```

# Ejercicio guiado

## Recorriendo arreglos con *for... of*

- **Paso 5:** Al ejecutar el código anterior, el resultado mostrado en la consola del navegador quedaría:

```
49  
51  
21  
18  
15  
0  
1  
2  
3  
4
```

¿Existe algún concepto que no  
hayas comprendido?

Volvamos a revisar los  
conceptos que más te hayan  
costado antes de seguir  
adelante.







## Próxima sesión...

- *Identifica las características principales de los objetos en el lenguaje Javascript así como sus objetos preconstruidos más utilizados.*
- *Utiliza el objeto Math para la construcción de un algoritmo que resuelve un problema.*
- *Utiliza el objeto String para la construcción de un algoritmo que resuelve un problema.*

**{desafío}**  
**latam\_**

*Academia de  
talentos digitales*

