



TEMA 5A

ARRAYS



1. ¿QUE ES UN ARRAY?

- Todos los lenguajes de programación disponen de un tipo de variable que es capaz de manejar conjuntos de datos. A este tipo de estructuras se las llama arrays de datos.
- También se las llama listas, vectores o arreglos, pero todos estos nombres tienen connotaciones que hacen que se puedan confundir con otras estructuras de datos.
- Por ello, es más popular el nombre sin traducir al castellano: array.
- Los arrays son variables que permiten almacenar, usando una misma estructura, una serie de valores. Para acceder a un dato individual dentro del array, hay que indicar su posición. La posición es un número entero conocido como **índice**

1. ¿QUE ES UN ARRAY?

nota

nombre del array.
permite referirnos
al array entero

7	nota[0]
8	nota[1]
6	nota[2]
6	nota[3]
5	nota[4]
4	nota[5]
3	nota[6]
9	nota[7]

↑ valores ↑ índices

2. CREACIÓN DE ARRAYS

- Hay muchas formas de declarar un array. Por ejemplo, si deseamos declarar un array vacío, la forma habitual es:

```
let a=[];
```

- Los corchetes vacíos tras la asignación, significan array vacío. Un array que se llama simplemente **a**
- Equivalente a ese código, podemos hacer:

```
let b=new Array();
```

2. CREACIÓN DE ARRAYS

- Lo cual ya nos anticipa que los arrays, en realidad, son objetos. El operador **new** sirve para crear objetos.
- Para colocar valores en el array se usa el índice que indica la posición del array en la que colocamos el valor. El primer índice es el cero, por ejemplo:

```
a[0]="Antonio";
```

- En este caso asignamos el texto Antonio al primer elemento del array.

2. CREACIÓN DE ARRAYS

- Podemos seguir:

```
a[1]="Luis";  
a[2]="Marta";  
a[3]="Sofia";
```

- Si quisiéramos mostrar un elemento concreto por consola haríamos:

```
console.log(a[2]); // Muestra Marta
```

2.CREACIÓN DE ARRAYS

- Podemos asignar valores en la propia declaración del array:

```
let a=[7,8,6,6,5,4,3,9];
```

- Disponemos también de esta otra forma de declarar:

```
let nota=new Array(7,8,6,6,5,4,3,9);
```

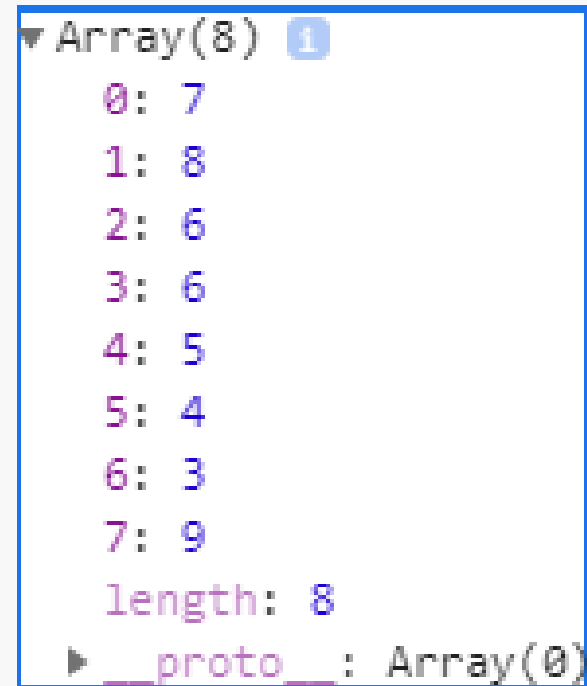
- En este caso, la variable nota es un array de diez elementos, todos números. Para mostrar el primer elemento:

```
console.log(nota[0]); // Muestra 7
```

2. CREACIÓN DE ARRAYS

- El método `console.log` tiene capacidad para mostrar todo un array:

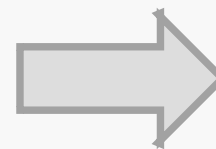
```
console.log(nota);
```



2. CREACIÓN DE ARRAYS

- Es habitual ver los arrays definidos con **const**, que sea constante no implica nada, ya que podemos cambiar los datos del array, y añadir y eliminar elementos sin problema.

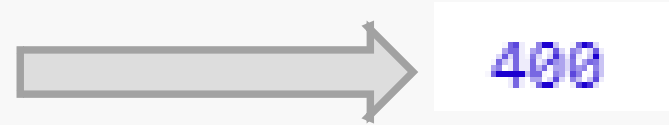
```
const datos=[4.5,6.78,7.12,9.123];  
datos[1]=9.18;  
datos[2]=3.25;  
console.log(datos);
```



```
▼ (4) [4.5, 9.18, 3.25, 9.123] ⓘ  
  0: 4.5  
  1: 9.18  
  2: 3.25  
  3: 9.123  
  length: 4  
  ▶ __proto__: Array(0)
```

3 . OPERACIÓN DE ASIGNACIÓN CON ARRAYS

```
const datos=[4.5,6.78,7.12,9.123];  
const datos2=datos;  
datos2[0]=400;  
console.log(datos[0]); //escribe 400
```



- Cuando asignamos la variable **datos2** al array **datos**, no se hace una copia del array. Tanto **datos** como **datos2** son dos variables que hacen referencia al mismo array.
- Por eso, cuando cambiamos el primer elemento del array **datos2**, también cambiamos el del array **datos**, ya que en realidad es el mismo array.

4. VALORES INDEFINIDOS

```
let a=["Raúl","Rocio"];  
a[3]="María ";  
console.log(a[2]);
```



undefined

- Veamos este código:
- En la definición del array: **let a: ["Saúl ", " Rocío"] ;** estamos creando el array llamado **a** y dando valor a los dos primeros elementos (**a[0]** y **a[1]**). Luego, damos valor al cuarto elemento del array (**a[3] =" María"**).
- En ningún momento hemos dado valor al elemento **a[2]** y por eso, el código provoca la escritura del texto **undefined**.

4. VALORES INDEFINIDOS

- El resumen es que podemos dejar elementos sin definir en un array. Es más, incluso en la propia declaración se pueden dejar elementos vacíos:

```
let a=["Saúl", "Rocío",,"María"];
```

- Las dos comas seguidas en la definición (después del valor "María") hacen que el tercer elemento (que sería **a[2]**) se ignore.

5. ELIMINAR ELEMENTOS DE UN ARRAY

- Para borrar un elemento se utiliza la palabra **delete** tras la cual se indica el elemento a eliminar.

```
const dias=["Lunes","Martes","Miércoles","Jueves",  
"Viernes","Sábado","Domingo"] ;  
delete dias[2];  
console.log(dias);
```

```
(7) ["Lunes", "Martes", empty, "Jueves", "Viernes", "Sábado", "Domingo"]  
i  
0: "Lunes"  
1: "Martes"  
3: "Jueves"  
4: "Viernes"  
5: "Sábado"  
6: "Domingo"  
length: 7  
__proto__: Array(0)
```

6. ARRAYS HETEROGENEOS

- Los arrays de JavaScript son heterogéneos. Admiten mezclar en el mismo array valores de diferente tipo:

```
const a=[3.4,"Hola",true,Math.random()];
```

- El array contiene dos números enteros (el 3 y el 4), un valor booleano (true) y un número decimal (resultado de la expresión Math.random()).
- Incluso podemos definir arrays de este tipo.

```
const b=[3.4,"Hola",[99,55,33]];
```

6. ARRAYS HETEROGENEOS

- El tercer elemento (b[2]) es un array.
- Podemos declarar arrays dentro de otros arrays.
- De modo que esta instrucción es posible:

```
console.log(b[2][1]);
```



55

7. RECORRER ARRAYS: LENGTH

- Una de las tareas fundamentales, en la manipulación de arrays, es recorrer cada elemento de un array para poderlo examinar.
- Esto es fácil de hacer mediante el bucle for y con el tamaño del array:

```
const notas = [5, 6, 7, 4, 9, 8, 9, 9, 7, 8];  
for (let i = 0; i < notas.length; i++) {  
    console.log("La nota " + i + " es: " + notas[i]);  
}
```


7. RECORRER ARRAYS: LENGTH

- En el bucle se utiliza el método `length` que nos permite saber el tamaño de un array.
- El contador `i` recorre todos los índices del array `notas`. El resultado del código es:

La nota 0 es: 5

La nota 1 es: 6

La nota 2 es: 7

La nota 3 es: 4

La nota 4 es: 9

La nota 5 es: 8

La nota 6 es: 9

La nota 7 es: 9

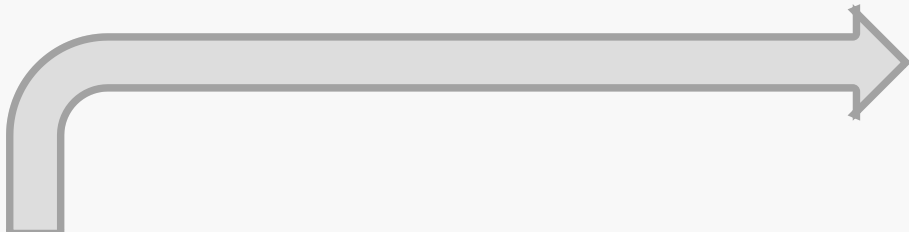
La nota 8 es: 7

La nota 9 es: 8

7. RECORRER ARRAYS: FOR OF

- El estándar E82015 incorporó un nuevo bucle llamado `for..of`. Pensado para simplificar aún más el recorrido de arrays,
- Es un bucle en el que la variable que se crea en el bucle va almacenando los diferentes valores del array
- Ejemplo:

7. RECORRER ARRAYS: LENGTH



```
const notas = [5, 6, 7, 4, 9, 8, 9, 9, 7, 8];  
for (let valor of notas) {  
  console.log("La nota es: " + valor);  
}
```

La nota es: 5

La nota es: 6

La nota es: 7

La nota es: 4

La nota es: 9

La nota es: 8

2 La nota es: 9

La nota es: 7

La nota es: 8

8. TAMANO DEL ARRAY

- La propiedad `length` nos permite obtener el tamaño de un array:

```
const dias = ["Lunes", "Martes", "Miércoles",  
"Jueves", "Viernes", "Sábado", "Domingo"];  
console.log(dias.length); //Escribe 7
```

- Es fácil, gracias a este método, añadir un elemento al final de un array:

```
nota[nota.length]=8;
```

- De esta forma, añadimos el número 8 al final del array `nota`.

9. METODOS PARA ANADIR ELEMENTOS

- El método **push** permite añadir un elemento al final del array. Requiere indicar qué valor le damos a ese elemento:

```
const cantantes=["Simon"];  
cantantes.push("Garfunkel");
```

- Ahora el array cantantes tiene dos elementos (Simon y Garfunkel).

9. METODOS PARA AÑADIR ELEMENTOS

- El método contrario es **pop**. Lo que hace es retirar del array el último elemento, podemos asignar dicho elemento a una variable:

```
const cantantes = ["Peter", "Paul", "Mary"];  
let x = cantantes.pop();  
console.log("El array ha quedado asi: " + cantantes);  
console.log("La variable x vale: " + x);
```

9. METODOS PARA AÑADIR ELEMENTOS

- El método `pop` no tiene argumento, pero requiere paréntesis (como todos los métodos). Tras el uso de **`pop`**, se quita el valor `Mary` y se asigna dicho valor a una variable llamada `x`. Por eso, lo que se escribe en pantalla es:

```
El array ha quedado así: Peter,Paul
```

```
La variable x vale: Mary
```

9. METODOS PARA AÑADIR ELEMENTOS

- Hay dos métodos más que permiten añadir y quitar elementos a un array.
- La diferencia es que lo hacen por delante, por el primer elemento del array. El método que quita el primer elemento del array se llama **shift**

```
const cantantes=["Crosby","Stills","Nash"];  
let x= cantantes.shift();  
console.log("El array ha quedado: "+cantantes);  
console.log("la variable x vale "+x);
```


9. METODOS PARA AÑADIR ELEMENTOS

- En este caso, es “Crosby” el elemento eliminado, por eso lo que se escribe en pantalla es:

```
El array ha quedado: Stills,Nash
```

```
la variable x vale Crosby
```

9. METODOS PARA AÑADIR ELEMENTOS

- El método contrario es **unshift**, al que hay que indicar como parámetro el valor a añadir por el lado izquierdo:

```
const cantantes=["Crosby","Stills","Nash"];  
cantantes.unshift("Young");  
console.log("El array ha quedado: "+cantantes);
```



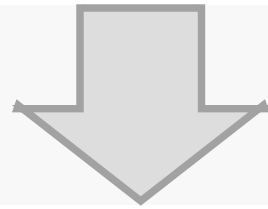
```
El array ha quedado: Young,Crosby,Stills,Nash
```

10. MEZCLAR ARRAYS

- El método **concat** permite añadir los elementos de un array al array que usa el método. El resultado de este método es un nuevo array, no se modifica ninguno de los dos arrays originales, que contiene los elementos unidos de ambos arrays.
- Por eso, el resultado de concat debe de ser asignado a un nuevo array.
- Ejemplo:

10. MEZCLAR ARRAYS

```
const planetas1=["Venus","Mercurio","La Tierra",  
"Marte"];  
const planetas2=["Júpiter", "Saturno","Urano",  
"Neptuno"];  
const planetas=planetas1.concat(planetas2);  
console.log(planetas);
```



```
► (8) ["Venus", "Mercurio", "La Tierra", "Marte", "Júpiter", "Saturno", "Uran  
o", "Neptuno"]
```

11. CONVERTIR ARRAY EN STRING

- El método **join** de los arrays permite convertir un array en un string. El string resultante contiene todos los elementos del array:

```
const dias=["Lunes","Martes","Miércoles","Jueves",  
"Viernes","Sábado","Domingo"];  
console.log(dias.join());
```



```
Lunes,Martes,Miércoles,Jueves,Viernes,Sábado,Domingo
```

10. MEZCLAR ARRAYS

- Cada elemento se separa con una coma. Es posible indicar otro separador ya que se puede indicar un texto como parámetro de join para que sea utilizado como separador:

```
const dias=["Lunes","Martes","Miércoles","Jueves",  
"Viernes","Sábado","Domingo"];  
console.log(dias.join(","));  
console.log(dias.join("-"));  
console.log(dias.join("<-->"));  
|
```

10. MEZCLAR ARRAYS

```
LunesMartesMiércolesJuevesViernesSábadoDomingo
```

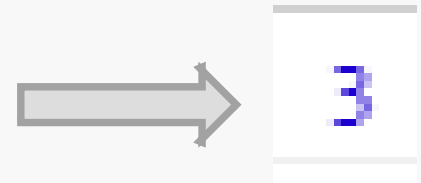
```
Lunes-Martes-Miércoles-Jueves-Viernes-Sábado-Domingo
```

```
Lunes<-->Martes<-->Miércoles<-->Jueves<-->Viernes<-->Sábado<-->Domingo
```

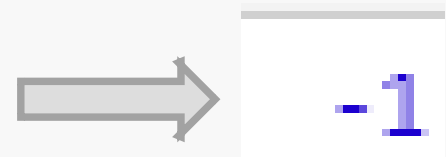
11. BÚSQUEDA DE ELEMENTOS EN UN ARRAY. METODO INDEXOF

- Este método permite buscar el valor elemento en un array. Si lo encuentra devuelve su posición y Si no, devuelve el valor -1. Ejemplo:

```
const dias=["Lunes","Martes","Miércoles","Jueves",  
"Viernes"];  
console.log(dias.indexOf('Jueves'));
```



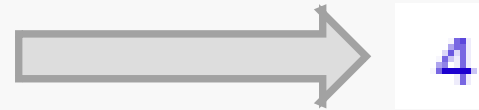
```
const dias=["Lunes","Martes","Miércoles","Jueves",  
"Viernes"];  
console.log(dias.indexOf('Sabado'));
```



12. BÚSQUEDA DE ELEMENTOS EN UN ARRAY. METODO LASTINDEXOF

- Método idéntico al anterior, solo que empieza a buscar desde el último elemento:

```
const numeros=[1,2,3,2,1,7];  
console.log(numeros.lastIndexOf(1));
```



13. BÚSQUEDA DE ELEMENTOS EN UN ARRAY. METODO INCLUDES

- Apareció en el estándar ES2015. Es un método que permite buscar un elemento y devuelve true si ese valor está en el array y false si no lo está.

```
const numeros=[1,2,3,2,1,7];  
console.log(numeros.includes(1));
```

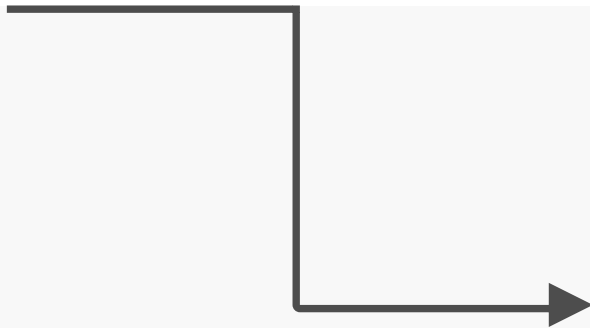


true

14. INVERTIR EL ORDEN

- El método `reverse` es capaz de dar la vuelta a los elementos de un array:

```
const numeros=[1,2,3,2,1,7];  
numeros.reverse();  
console.log(numeros);
```



```
(6) [7, 1, 2, 3, 2, 1] ⓘ  
  0: 7  
  1: 1  
  2: 2  
  3: 3  
  4: 2  
  5: 1  
  length: 6  
  __proto__: Array(0)
```

15. METODO SORT

- Sin duda, es uno de los métodos más interesantes de los arrays. Permite ordenar los elementos de un array. Ejemplo:

```
const dias=["Lunes","Martes","Miércoles","Jueves",  
"Viernes"];  
dias.sort();  
console.log(dias);
```

```
(5) ["Jueves", "Lunes", "Martes", "Miércoles", "Viernes"]
0: "Jueves"
1: "Lunes"
2: "Martes"
3: "Miércoles"
4: "Viernes"
length: 5
__proto__: Array(0)
```