

Arrays/arreglos

Introducción

Los arrays suelen ser definidos como grupo de elementos o también o lista de elementos. ¿Qué significa esto? Vamos a verlo con un ejemplo.

Supongamos que queremos guardar la edad de 5 personas. Una primera solución podría ser que hagamos cinco variables y en cada una se guarda una edad.

```
1  let edad1 = 18;
2  let edad2 = 35;
3  let edad3 = 16;
4  let edad4 = 15;
5  let edad5 = 12;
6
```

Esto es válido, pero en este caso estamos pensando en 5 variables.

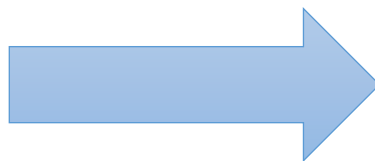
Ahora ¿qué pasa si el ejercicio nos pide guardar **70 variables**?

¿Será buena idea escribir 70 variables?
No, es tedioso y además consume recursos y memoria RAM.

¿Cómo hago para guardar 70 números en una sola variable?

Escribir `let edades = 18, 35, 16, 15, 12;` es ilegal.

Solución

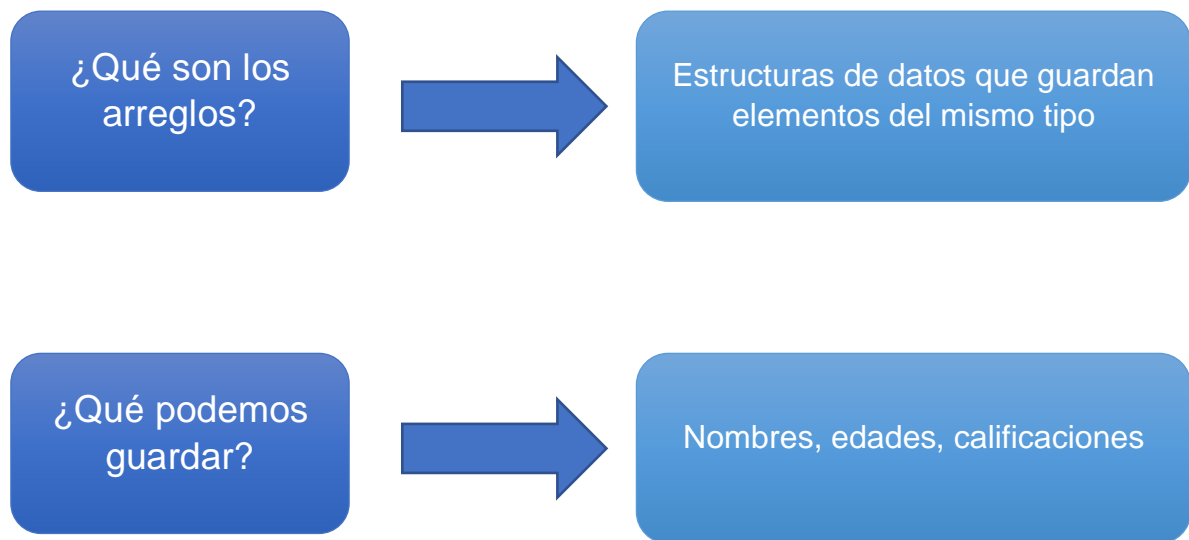


Arreglos

Concepto de arreglos

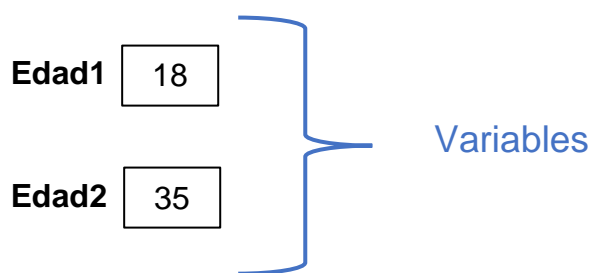
Antes que nada, a los arreglos se los suelen llamar con distintos nombres: arrays, arreglos, vectores, listas (dependiendo del lenguaje de programación, las listas son otra estructura distinta a los arreglos, pero no viene al caso).

La idea de los arreglos es la de solucionarnos ese problema, de almacenar muchos datos.



Es decir, los arrays los ocupamos cuando queremos guardar varios datos.

Cómo visualizamos esto

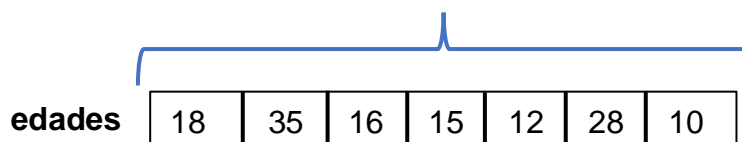


Como se puede observar, los arrays son muy convenientes cuando queremos guardar varios datos en un solo lugar.

Es como si estuviéramos usando una variable para guardar varios datos.

(Ojo, no estamos usando una variable, estamos usando un array, que no se confunda)

Arreglo de edades



Crear arrays en JavaScript

Array vacío

```
const array = [];
```

En lugar de **let**,
escribimos **const**

Nombre del
arreglo

Dentro de los corchetes []
Se escriben los elementos
separados por coma. Si se
dejan vacíos, el array queda
vacío, sin elemento.

Podemos agregar los
elementos en otra línea del
código

Crear arrays con elementos por defecto

```
const array = [1, 2, 3, 4, 5];
```

Como se puede ver,
aparecen los elementos
separados por coma

Este sería un arreglo de
números

Distintos tipos de datos

```
const arrayDeNumeros = [1, 2, 3, 4, 5];
```

Array de números enteros con
5 elementos

```
const arrayDeFlotantes = [3.14, 6.456, 1.2342];
```

Array de números flotantes con
tres elementos

```
const arrayDeTextos = ["hola", "mundo", "jeje"];
```

Array de cadenas de texto
(string) con tres elementos

```
const arrayDeLetras = ['p', 'a', 'b', 'l', 'o'];
```

Array de cadenas de letras
(caracteres) con 5 elementos

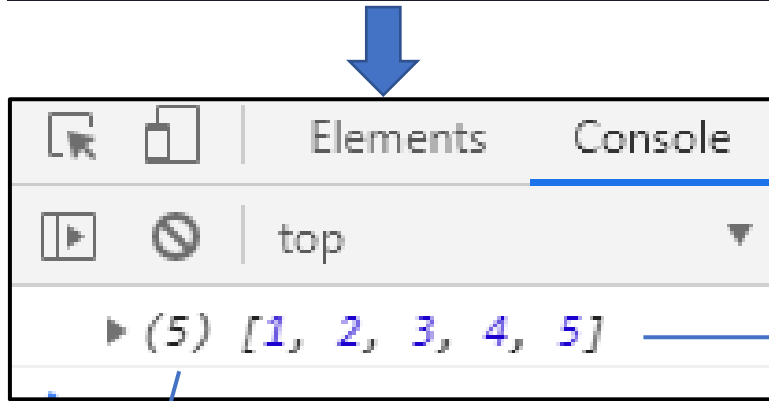
```
const arrayDeBooleanos = [true, false, false, false];
```

Array de cadenas de booleanos
con 4 elementos

mostrar los elementos por pantalla

Si nosotros mandamos a imprimir en un `console.log` el array, nos mostrará todos los elementos (este método lo usarás para cosas específicas).

```
const array = [1, 2, 3, 4, 5];  
console.log(array);
```



Nos muestra la cantidad de elementos que tiene el array

Nos muestra por consola todos los elementos que contiene

Mostrar elementos específicos

¿Qué pasa si no quiero mostrar todos los elementos?

¿Cómo hago si quiero mostrar solo dos números primeros números?

En ese caso tenemos que entender **cómo se guardan los elementos en los arrays**.

Cada elemento del array tiene un **índice** asignado, índice por el cual se lo identifica, como un DNI. Esos índices son las posiciones.

Básicamente al elemento se lo identifica por la posición en el que fue guardado.

Como se puede ver el número 35 está en la posición 1.

1 es el índice

El número 12 está en la posición 4.

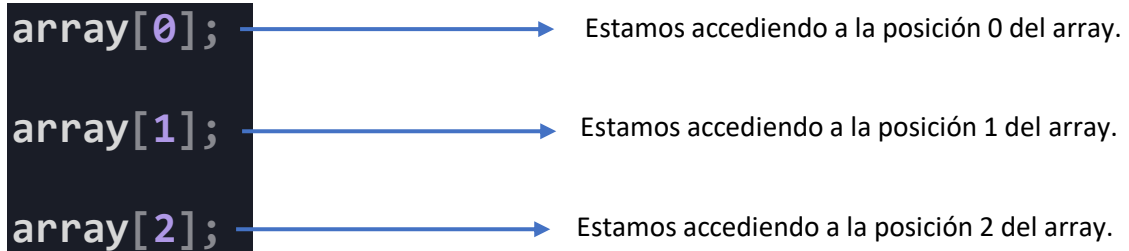
4 es el índice

0	1	2	3	4	5	6
18	35	16	15	12	28	10

**EN INFORMÁTICA SIEMPRE SE CUENTA DESDE CERO
NUNCA DESDE UNO.**

Acceder a los índices

Haciendo uso de los corchetes [] accedemos a la posición, indicando el número de la posición que queremos.



```
array[0];
```

→ Estamos accediendo a la posición 0 del array.

```
array[1];
```

→ Estamos accediendo a la posición 1 del array.

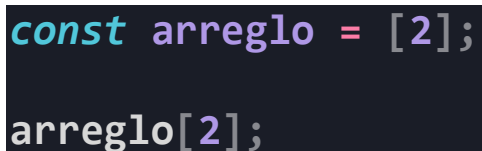
```
array[2];
```

→ Estamos accediendo a la posición 2 del array.

OJO, NO CONFUNDIRSE

Al poner el **const** delante, estamos **creando** el array y le estamos dando un valor.

Si no tiene el const, estamos accediendo a la posición indicada



```
const arreglo = [2];
```

```
arreglo[2];
```

Estamos creando un array que guarda el número 2 en la posición 0.

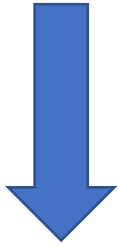
Estamos accediendo a la posición 2 del arreglo.

Mostrando los elementos por el índice

Podemos mandar a mostrar por pantalla los elementos que nosotros queramos indicando la posición.

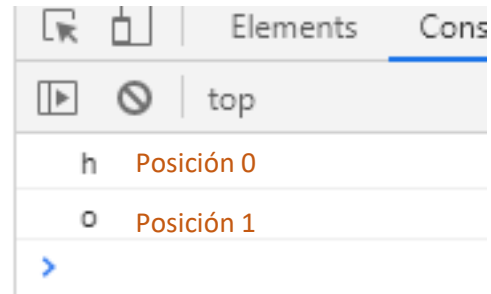
```
const letras = ['h', 'o', 'l', 'a'];
```

Creamos el array de letras



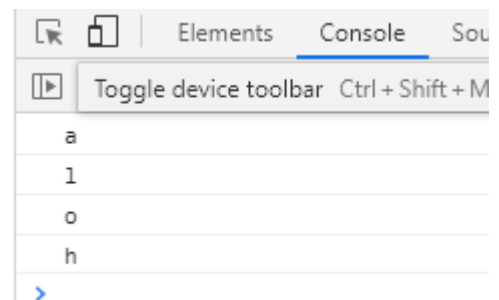
Mando a imprimir el elemento en posición 0 y el que está en posición 1

```
console.log( letras[0] );  
console.log( letras[1] );
```



Podemos imprimir las letras al revés

```
console.log( letras[3] );  
console.log( letras[2] );  
console.log( letras[1] );  
console.log( letras[0] );
```



Recorrer el array con for

El acceder a los índices nos permite obtener **todos** los valores del array por medio de un for.

Esa acción de acceder a **todos** los elementos por su índice con un for, se le denomina como recorrer el array.

```
const letras = ['h', 'o', 'l', 'a'];  
for(let i = 0; i < 4; i++){  
  console.log( letras[i] );  
}
```

A la variable *i* le asignamos cero para que comience a recorrer el array desde cero.

Va a recorrer el for desde cero hasta 3 porque hay cuatro elementos en el array

Accedemos a la posición haciendo uso de la variable *i* ya que va a ir recorriendo las posiciones

Primera ejecución del for

segunda ejecución del for

tercera ejecución del for

cuarta ejecución del for

Inclusive imprimir los datos con el for nos da la posibilidad de formatear la salida.
Es decir, escribir como queremos que aparezca el dato.

```
for(let i = 0; i < 4; i++){  
  console.log( "la letra en la posición " + i + " es " + letras[i] );  
}
```

En este ejemplo vamos a mostrar
el índice del elemento que se
guarda en la variable i

Y mostramos el valor
correspondiente que se
encuentra en esa posición

Variable i

letra[i]

Resultado

```
top  
la letra en la posición 0 es h  
la letra en la posición 1 es o  
la letra en la posición 2 es l  
la letra en la posición 3 es a  
>
```


Poner valores en una posición indicada

Los índices también nos dejan poner valores en la posición que le indiquemos.

```
const numeros = [];  
numeros[0] = 2;  
console.log( numeros[0] );  
console.log( numeros[4] );
```

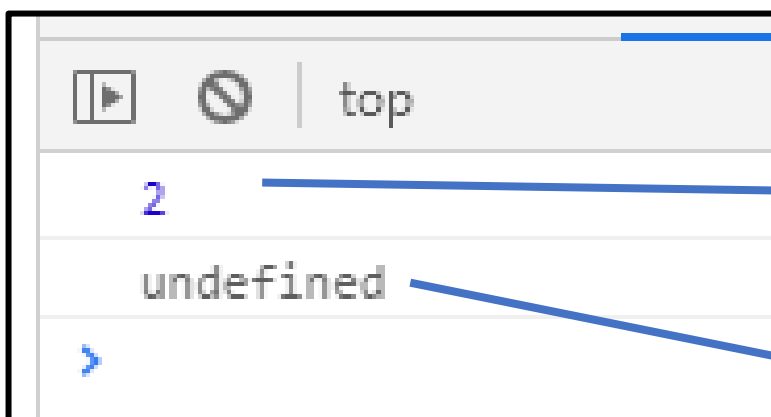
En este caso hice un array

En la primera posición pongo un 2 como valor

Lo muestro en la consola

Acá intento acceder a la posición 4, pero no hay nada en esa posición, porque solo puse un número 2 en la posición cero. Es decir, el array tiene un solo valor.

Esta línea tirará un error



Me muestra el valor en la posición cero

Undefined significa que la posición a la queremos acceder (en este caso la 4) no tiene ningún valor, está vacía

Pedir al usuario los valores del array

```
const numeros = [];  
  
for(let i = 0; i < 7; i++){  
    numeros[i] = parseInt(prompt("ingresar número"));  
}
```

En este caso el usuario agrega 7 números

Luego los imprimo por consola en un segundo for con un formato

```
for(let i = 0; i < 7; i++){  
    console.log( "posición " + i + ": " + numeros[i]);  
}
```

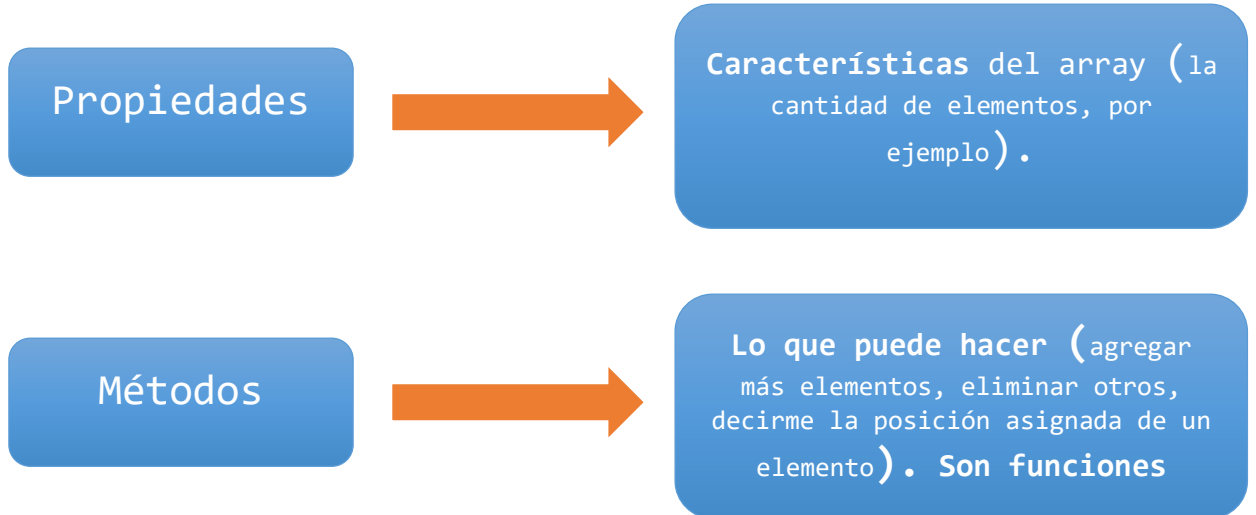
Números que ingresó el usuario

Resultados

posición 0:	10
posición 1:	20
posición 2:	30
posición 3:	40
posición 4:	50
posición 5:	60
posición 6:	70
>	

Propiedades y métodos

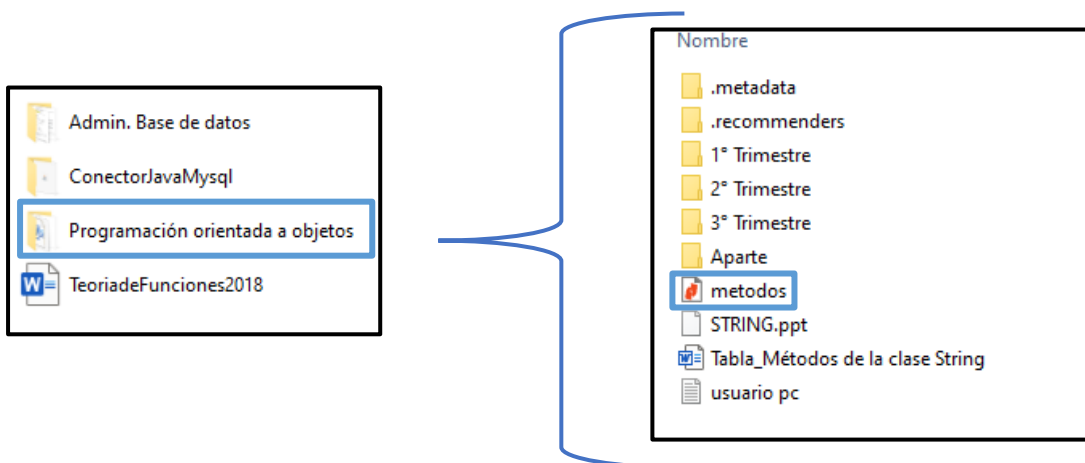
Todo en JavaScript tiene propiedades y métodos. Variables, arrays, todo.



Nomenclatura del punto

Cuando usamos el punto en programación, nos referimos a que **queremos acceder a algo**. ¿a qué queremos acceder? A los métodos y propiedades del array, por ejemplo.

Vamos a entender esto: supongamos que tenemos varias carpetas sobre unas materias del colegio y quiero acceder a un PDF llamado “métodos” que se encuentra dentro de la carpeta “programación orientada a objetos”.



programaciónOrientadaAobtejos.métodos

El programa busca la carpeta
“programación orientada a objetos”

Y mediante el punto se mete
dentro de la carpeta

Y entra a al PDF
“métodos”

Con esta explicación tenemos entendido que el punto hace referencia a acceder a los métodos y propiedades del array.

Por lo tanto, si quiero por ejemplo obtener la cantidad de elementos de un array, usaría el punto de la siguiente manera



array.length

Accedo a la longitud del array.

De la misma manera, si quiero acceder a un método (por ejemplo, el de indicar la posición de un elemento), lo haría de la siguiente manera.



Array.indexOf(elem)

Me dice en qué posición se guardó el elemento indicado en los paréntesis

Usar paréntesis en los métodos

Los métodos son funciones, como alert(), prompt(), typeof(), etc.

Las funciones cuando van **acompañadas del punto**, se les llama **métodos**.

Las **propiedades** no llevan paréntesis porque son **variables**.

alert()



Función

Windows.alert()



Método

Lista de propiedades y métodos de arrays

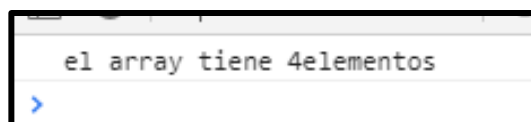
1)array.length

Esta propiedad indica el tamaño del array. Es decir, indica cuántos elementos tiene.

Básicamente es una variable interna del array.

```
const array = [1, 2, 3, 4];  
let tam = array.length;  
console.log("el array tiene " + tam + " elementos");
```

En este caso guardo la longitud en una variable, pero también podría imprimir array.length en el console.log()



el array tiene 4 elementos

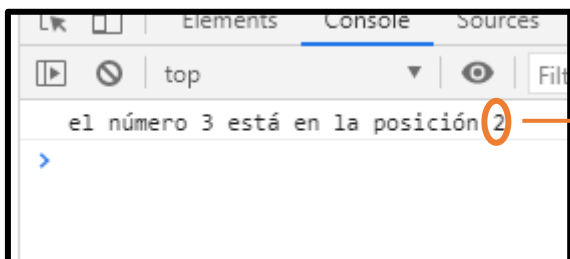
2)array.indexOf(elem)

Devuelve la posición del elemento indicado en los paréntesis.

Devuelve significa que lo deja como resultado, es decir, si queremos usarlo, tenemos que asignar a una variable el número que devuelve el método.

```
const array = [1, 2, 3, 4];  
let pos = array.indexOf(3);  
console.log("el número 3 está en la posición " + pos);
```

Le indico que me de el número de la posición donde esté el 3. Ese número que me da lo guardo en una variable



el número 3 está en la posición 2

Es el número guardado en la variable pos.

¿Qué pasa si quiero la posición de un elemento que no existe?

Si el número no se encuentra, me va a devolver un -1. El -1 indica que el elemento no existe en el array.

Veamos el siguiente ejemplo donde pido la posición de dos elementos que existen y después la posición de otros dos que no existen.

```
const array = [1, 2, 3, 4];  
  
//Estos elementos existen  
let posicionDos = array.indexOf(2);  
let posicionCuatro = array.indexOf(4);  
  
//Estos elementos no existen  
let posicionSiete = array.indexOf(7);  
let posicionnueve = array.indexOf(9);  
  
console.log("el número 2 está en la posición " + posicionDos);  
console.log("el número 4 está en la posición: " + posicionCuatro);  
console.log("el número 7 está en la posición " + posicionSiete);  
console.log("el número 9 está en la posición " + posicionnueve);
```

Van a devolver la posición

Van a devolver -1 porque no existen

Resultado

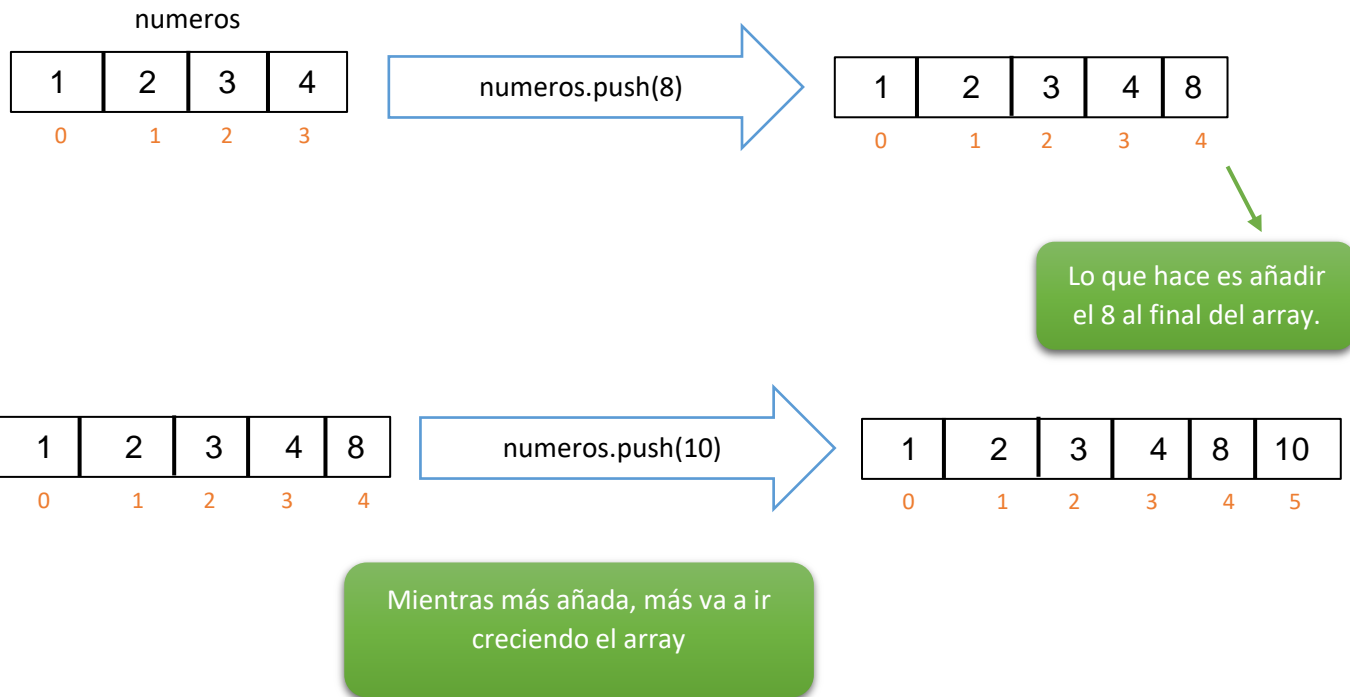
```
el número 2 está en la posición 1  
el número 4 está en la posición: 3  
el número 7 está en la posición -1  
el número 9 está en la posición -1  
>
```

Existen, por eso el método devuelve la posición

No existen, por eso devuelve -1

3) array.push(elem)

Añade al final del array un elemento que le indiquemos en los paréntesis e incrementa el tamaño del array.



Este método no devuelve nada, veamos el ejemplo en código y el resultado.

```
const array = [1, 2, 3, 4];
console.log("el array se creó y es" , array);

//Le agrego otro elemento
array.push(8);

console.log("el array se modificó y ahora es", array);

array.push(10);

console.log("el array se modificó y ahora es", array);
```

Primero muestro el array con los valores con los que empieza

Luego de agregar el elemento lo vuelvo a mostrar para que muestre el cambio

```
el array se creó y es ► (4) [1, 2, 3, 4]
el array se modificó y ahora es ► (5) [1, 2, 3, 4, 8]
el array se modificó y ahora es ► (6) [1, 2, 3, 4, 8, 10]
```

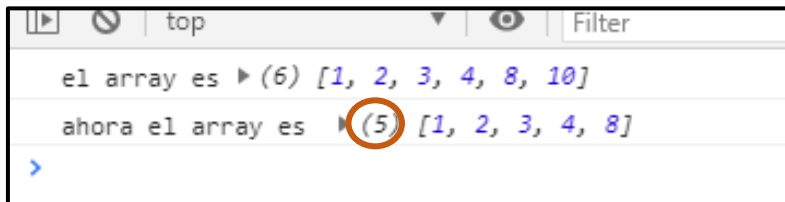
Se agregan los elementos al final del array y aumenta el tamaño

4) array.pop()

Elimina el último elemento que haya en el array y se modifica el tamaño, ya que se saca el último elemento.

```
const array = [1, 2, 3, 4, 8, 10];  
console.log("el array es" , array);  
  
array.pop();  
  
console.log("ahora el array es ", array);
```

Elimina el 10, ya que ese es el último elemento del array.



The screenshot shows a browser console with two log messages. The first message is "el array es (6) [1, 2, 3, 4, 8, 10]". The second message is "ahora el array es (5) [1, 2, 3, 4, 8]". The number 5 in the second message is circled in orange, indicating the change in array length.

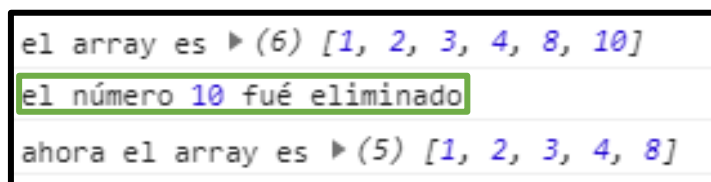
Modifica el tamaño

El método pop lo que hace es eliminar el resultado y devolverlo. En el ejemplo anterior lo único que hice fue eliminarlo y listo, pero veamos el siguiente ejemplo en donde primero guardo el número que se elimina y luego lo elimino.

```
const array = [1, 2, 3, 4, 8, 10];  
console.log("el array es" , array);  
  
let num = array.pop();  
  
console.log("el número", num, "fué eliminado");  
console.log("ahora el array es", array);
```

El método pop() elimina el número y lo devuelvo.

Lo que hago es guardarlo en una variable.



The screenshot shows a browser console with three log messages. The first message is "el array es (6) [1, 2, 3, 4, 8, 10]". The second message is "el número 10 fué eliminado". The third message is "ahora el array es (5) [1, 2, 3, 4, 8]". The second message is highlighted with a green box.

5) array.splice(index, cant, elem1)

Este es algo complicado de entender, pero no imposible.

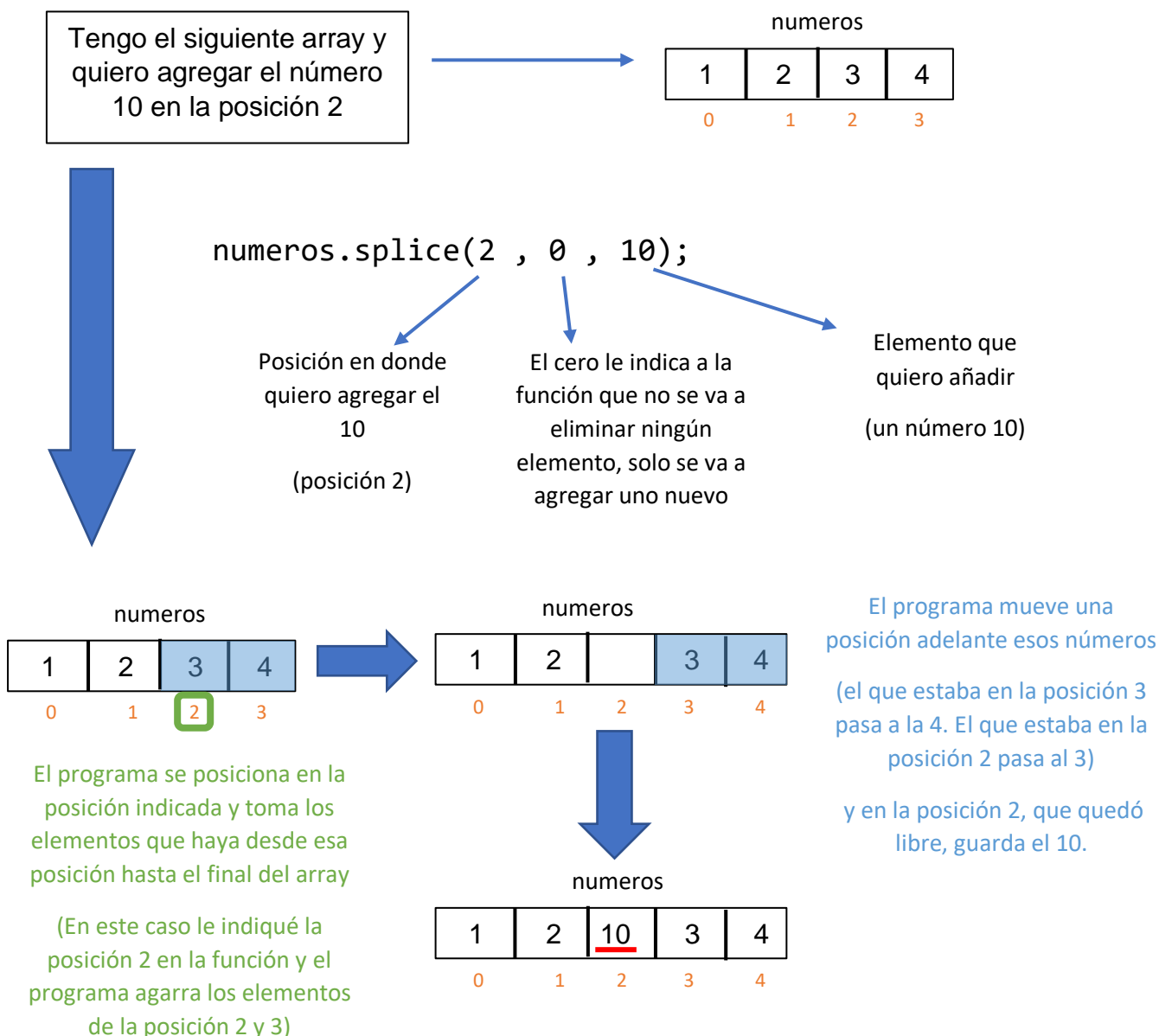
Lo que permite hacer es agregar o eliminar elementos **indicando la posición**.

Los anteriores métodos agregaban o eliminaban el elemento al final. Con este tenemos la libertad de elegir qué eliminar.

Agregar elementos

Cuando usamos splice() para agregar elementos lo que sucede es que cuando le indiquemos el índice, **el programa agarra** el elemento que está en esa posición y los que están adelante también **y los mueve una posición adelante**.

Vamos a visualizar esto:



```
const array = [1, 2, 3, 4];  
console.log("el array es" , array);  
array.splice(2, 0, 10);  
console.log("ahora el array es", array);
```

Meto el 10 en la
posición 2



```
el array es ▶ (4) [1, 2, 3, 4]  
ahora el array es ▶ (5) [1, 2, 10, 3, 4]  
|
```

Ejercicio:

(Recomiendo primero hacer el ejercicio solo y luego mirar la solución)

Estas haciendo un array que registra (en orden alfabético) cuales son los juegos que están **instalados** en la computadora.

El array queda de la siguiente manera:

juegosInstalados	"CSGO"	"Half-Life"	"Hollow Knight"	"Hotline Miami"
	0	1	2	3

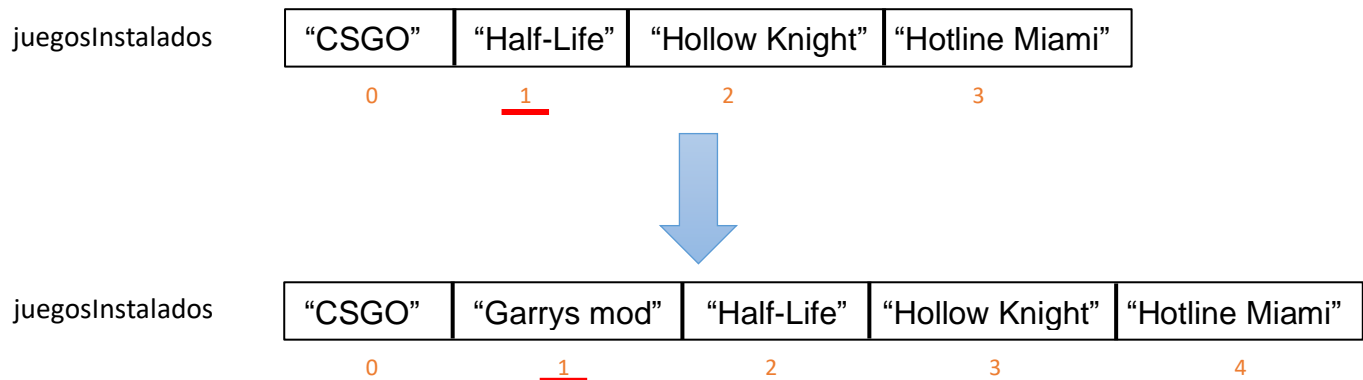
Unos días después, queda instalado Garrys mod en la computadora y tenes que agregarlo en el array.

Como el array se ordena alfabéticamente, Garrys mod debe aparecer entre CSGO y Half-Life.

Hacer el splice() correspondiente para agregar el juego en la posición que le corresponde.

Solución:

Hay que pasar de:



```
const juegosInstalados = ["CSGO", "Half-Life", "Hollow Knight", "Hotline miami"];

console.log("los juegos instalados son: ", juegosInstalados);

//modifica el array
juegosInstalados.splice(1, 0, "Garrys Mod");

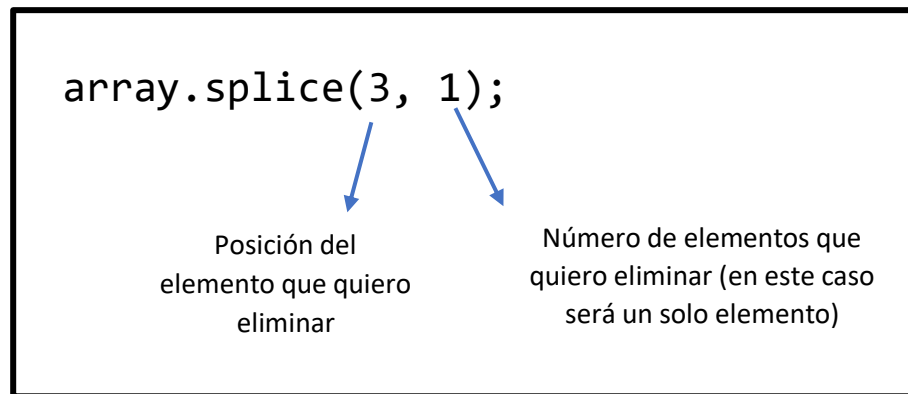
console.log("ahora los juegos son: ", juegosInstalados);
```

los juegos instalados son:
▶ (4) ["CSGO", "Half-Life", "Hollow Knight", "Hotline miami"]
ahora los juegos son:
▶ (5) ["CSGO", "Garrys Mod", "Half-Life", "Hollow Knight", "Hotline miami"]

Quitar elementos

Cuando queremos agregar elementos, el segundo parámetro de la función splice() es 0. El 0 indicaba que no se va a eliminar nada.

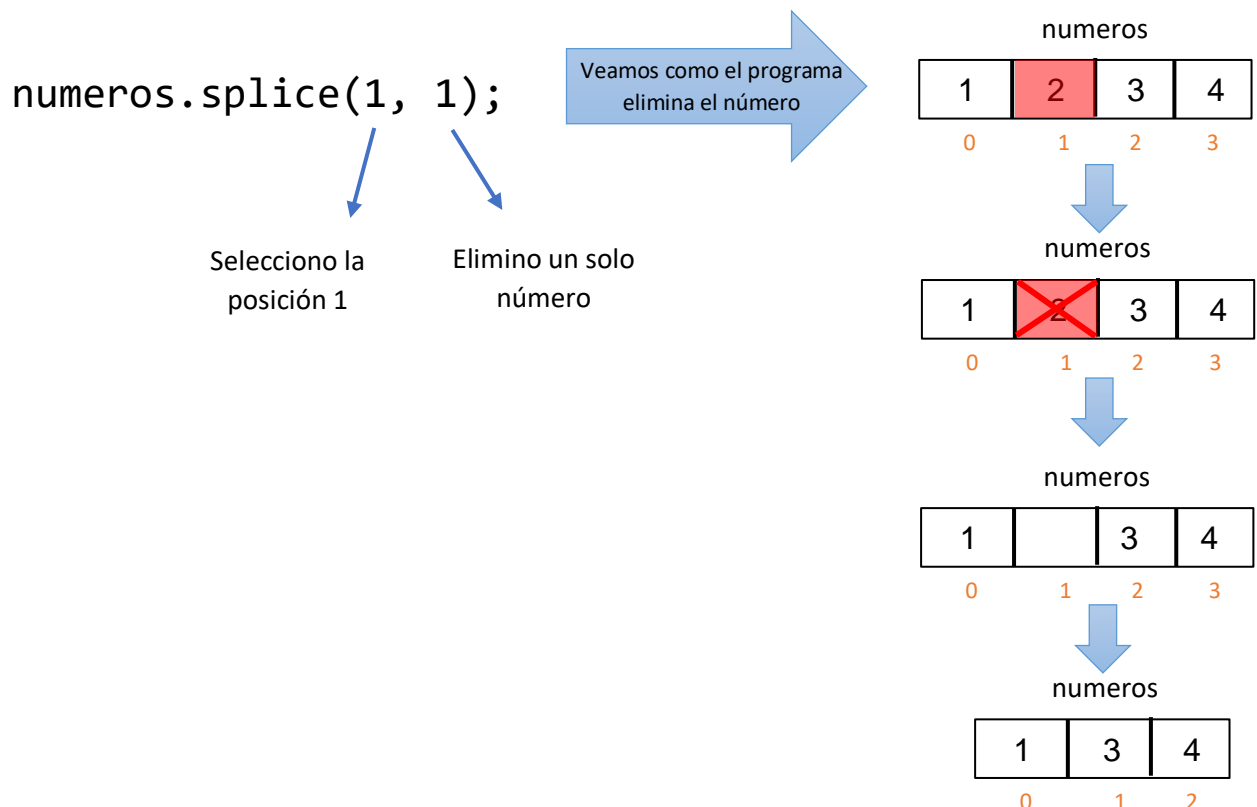
Para eliminar elementos, basta con indicar la posición e indicar cuántos elementos quitar del array.



Supongamos que tenemos el siguiente array:

numeros			
1	2	3	4
0	1	2	3

Si quiero eliminar el 2, solo debo seleccionar la posición 1 e indicar que quiero eliminar 1 solo número.



Veamos un ejercicio con código:

Supongamos que vuelvo a tener el array de los juegos instalados de la siguiente manera:

juegosInstalados	"CSGO"	"Garrys mod"	"Half-Life"	"Hollow Knight"	"Hotline Miami"
	0	1	2	3	4

Lo que quiero hacer es desinstalar el "Hollow Knight" (ósea lo quiero sacar del array).

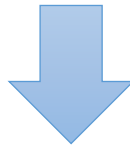
Al desinstalar el juego indicado, el array debe quedar de la siguiente manera:

juegosInstalados	"CSGO"	"Garrys mod"	"Half-Life"	"Hotline Miami"
	0	1	2	3

Solución:

```
const juegosInstalados = ["CSGO", "Garrys mod", "Half-Life", "Hollow Knight", "Hotline miami"];  
console.log("juegos: ", juegosInstalados);  
juegosInstalados.splice(3,1);  
console.log("juegos: ", juegosInstalados);
```

Se posiciona en el índice 3 y elimina un solo string.



```
juegos: ▶ (5) ["CSGO", "Garrys mod", "Half-Life", "Hollow Knight", "Hotline miami"]  
juegos: ▶ (4) ["CSGO", "Garrys mod", "Half-Life", "Hotline miami"]
```