



# Trabajo con arrays en JavaScript

Aprende a trabajar con arrays en JavaScript

Empezar

Requisitos previos

## Descripción general

En este curso aprenderás las principales técnicas y funciones para trabajar con arrays en JavaScript. Desde la creación y manipulación de arrays hasta el uso de métodos avanzados para filtrar, ordenar y transformar la información contenida en ellos. ¡No te lo pierdas!

01 Introducción

A decorative header bar with a blue background. It features a row of eight geometric shapes: a circle with a square inside, a square with a circle inside, a circle with a square inside, a square with a circle inside, a circle with a square inside, a square with a circle inside, a circle with a square inside, and a square with a circle inside. The shapes are arranged in a repeating pattern of circle-square-circle-square.

# Introducción a los arrays en JavaScript

01 | Introducción a los arrays en JavaScript

Los arrays son una parte fundamental en JavaScript ya que nos permiten almacenar y organizar múltiples valores en una sola variable. Un array puede contener elementos de diferentes tipos, como números, cadenas de texto, booleanos e incluso otros arrays.

En JavaScript, los arrays se crean utilizando corchetes ([]). Dentro de los corchetes colocaremos los elementos que deseamos almacenar, separados por comas. Por ejemplo, podemos crear un array de números de la siguiente manera:

```
let numeros = [1, 2, 3, 4, 5];
```

También es posible crear un array vacío y luego agregar elementos utilizando el método `push()`. Por ejemplo:

```
let frutas = [];  
frutas.push("manzana");  
frutas.push("plátano");  
frutas.push("naranja");
```

Para acceder a un elemento específico de un array, podemos utilizar su posición dentro del mismo. Los arrays en JavaScript están indexados, lo que significa que cada elemento tiene asignado un número que indica su posición. El primer elemento tiene un índice de 0, el segundo de 1 y así sucesivamente. Por ejemplo, para acceder al primer elemento del array `numeros`, podemos hacer lo siguiente:

```
let primerNumero = numeros[0];
```

Podemos modificar o reemplazar un elemento existente utilizando su posición también. Por ejemplo, si queremos cambiar el segundo elemento del array `frutas` por "pera", podemos hacer lo siguiente:

```
frutas[1] = "pera";
```

Además de acceder a elementos específicos, también podemos obtener la longitud de un array utilizando la propiedad `length`. Esta propiedad nos devuelve

la cantidad de elementos que contiene el array. Por ejemplo:

```
let cantidadNumeros = numeros.length;
```


Los arrays en JavaScript también nos permiten realizar diversas operaciones, como agregar elementos al final del array utilizando el método `push()`, eliminar el último elemento con el método `pop()`, agregar elementos al principio con el método `unshift()`, eliminar el primer elemento con el método `shift()`, entre otros.

También es posible recorrer un array utilizando bucles, como el bucle `for`. Podemos utilizar la propiedad `length` para saber cuántas veces debe repetirse el bucle. Por ejemplo, para recorrer el array `numeros` e imprimir cada uno de sus elementos, podemos hacer lo siguiente:

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

En resumen, los arrays nos permiten almacenar y organizar múltiples valores en una sola variable en JavaScript. Conocer cómo crear, acceder, modificar y recorrer arrays es esencial para trabajar con los mismos de manera efectiva y aprovechar al máximo sus funcionalidades.

En esta introducción a los arrays en JavaScript, hemos aprendido los conceptos básicos de cómo crear y acceder a arrays, así como cómo iterar sobre ellos utilizando bucles for y forEach. También hemos explorado las diferentes formas de recorrer un array y acceder a sus elementos utilizando los índices. Los arrays son una herramienta fundamental en JavaScript y son ampliamente utilizados en el desarrollo web.



## Manipulación de arrays: agregar, eliminar y modificar elementos

En JavaScript, un array es una estructura de datos que nos permite almacenar y organizar múltiples valores en una sola variable. Una de las principales ventajas de los arrays es que nos permiten manipular los elementos que contienen de diversas formas, como agregar nuevos elementos, eliminar elementos existentes o modificar los valores de los elementos.

## Agregar elementos a un array

Para agregar elementos a un array en JavaScript, podemos utilizar el método `push()`. Este método nos permite añadir uno o más elementos al final del array.

```
let miArray = [1, 2, 3];
miArray.push(4);
miArray.push(5, 6);

console.log(miArray); // [1, 2, 3, 4, 5, 6]
```

Además del método `push()`, también podemos usar la asignación directa para agregar elementos a un array utilizando el índice correspondiente:

```
let miArray = [1, 2, 3];
miArray[3] = 4;

console.log(miArray); // [1, 2, 3, 4]
```

## Eliminar elementos de un array

Existen diferentes formas de eliminar elementos de un array en JavaScript.

Para eliminar el último elemento de un array, podemos utilizar el método `pop()`. Este método elimina el último elemento del array y devuelve dicho elemento.

```
let miArray = [1, 2, 3, 4, 5];
let ultimoElemento = miArray.pop();

console.log(miArray); // [1, 2, 3, 4]
console.log(ultimoElemento); // 5
```

Podemos eliminar un elemento específico de un array utilizando el método `splice()`. Este método nos permite especificar el índice del elemento que queremos eliminar y la cantidad de elementos a eliminar a partir de ese índice.

```
let miArray = [1, 2, 3, 4, 5];
miArray.splice(2, 1);

console.log(miArray); // [1, 2, 4, 5]
```

También podemos eliminar elementos utilizando la asignación directa y asignando el valor `undefined` al elemento que queremos eliminar:

```
let miArray = [1, 2, 3, 4, 5];
miArray[2] = undefined;

console.log(miArray); // [1, 2, undefined, 4, 5]
```

## Modificar elementos de un array

Para modificar el valor de un elemento de un array, simplemente asignamos un nuevo valor al elemento utilizando su índice.

---

```
let miArray = [1, 2, 3, 4, 5];  
miArray[2] = 6;  
  
console.log(miArray); // [1, 2, 6, 4, 5]
```

También podemos modificar múltiples elementos a la vez utilizando el método `splice()`. Al especificar el índice del primer elemento a modificar y la cantidad de elementos a eliminar (en este caso 0), podemos insertar nuevos elementos en esas posiciones.

```
let miArray = [1, 2, 3, 4, 5];  
miArray.splice(1, 3, 7, 8, 9);  
  
console.log(miArray); // [1, 7, 8, 9, 5]
```

#### Conclusión - Manipulación de arrays: agregar, eliminar y modificar elementos

En esta sección sobre la manipulación de arrays, hemos aprendido cómo agregar, eliminar y modificar elementos en un array en JavaScript. Hemos analizado métodos como `push`, `pop`, `shift`, `unshift` y `splice`, que nos permiten realizar estas operaciones de manera efectiva. También hemos explorado cómo utilizar el método `slice` para crear copias de arrays y cómo concatenar arrays utilizando el operador `spread`. Con esta información, ahora tenemos el



conocimiento necesario para manipular arrays de manera eficiente en JavaScript.



# Operaciones avanzadas con arrays: filtrar, mapear y reducir

03 | Operaciones avanzadas con arrays: filtrar, mapear y reducir

## Filtrar elementos de un array

En JavaScript, es común encontrarse con la necesidad de filtrar elementos en un array con base en ciertos criterios. La operación de filtrado permite seleccionar

únicamente aquellos elementos que cumplan con una condición específica.

Para realizar esta operación, podemos utilizar el método `filter()` proporcionado por los arrays en JavaScript. Este método crea un nuevo array con todos los elementos que pasen la prueba implementada por la función de filtro proporcionada. Por ejemplo, si tenemos un array de números y queremos obtener únicamente los que son pares, podríamos hacer lo siguiente:

```
const numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];  
const numerosPares = numeros.filter(numero => numero % 2 === 0);  
console.log(numerosPares); // Output: [2, 4, 6, 8, 10]
```

En este caso, la función de filtro `numero => numero % 2 === 0` verifica si el número es divisible por 2 (es decir, si es par). Si la expresión se evalúa como verdadera, el número es incluido en el nuevo array `numerosPares`.

## Mapear elementos de un array

Otra operación común con arrays es la de mapear, que consiste en aplicar una transformación a cada elemento del array original y obtener un nuevo array con los resultados de dicha transformación.

En JavaScript, podemos utilizar el método `map()` para realizar esta operación. Este método crea un nuevo array con los resultados de llamar a la función proporcionada por cada elemento del array original. Por ejemplo, si tenemos un array de números y queremos obtener un nuevo array con el cuadrado de cada número, podríamos hacer lo siguiente:

---

```
const numeros = [1, 2, 3, 4, 5];  
const cuadrados = numeros.map(numero => numero ** 2);  
console.log(cuadrados); // Output: [1, 4, 9, 16, 25]
```

En este caso, la función `numero => numero ** 2` eleva al cuadrado cada número del array original, obteniendo un nuevo array `cuadrados` con los resultados.

## Reducir elementos de un array

La operación de reducir, o "reduce", permite combinar todos los elementos de un array en un solo valor. Esto es útil, por ejemplo, cuando queremos obtener la suma total de todos los elementos de un array.

En JavaScript, podemos utilizar el método `reduce()` para realizar esta operación. Este método aplica una función "reductora" a cada elemento del array, de izquierda a derecha, para reducirlo a un solo valor. Por ejemplo, si tenemos un array de números y queremos obtener la suma de todos ellos, podríamos hacer lo siguiente:

```
const numeros = [1, 2, 3, 4, 5];  
const sumaTotal = numeros.reduce((acumulador, numero) => acumulador + numero,  
console.log(sumaTotal); // Output: 15
```

En este caso, la función reductora `(acumulador, numero) => acumulador + numero` toma el acumulador (inicializado en 0) y le suma cada número del array original, obteniendo la suma total.

Estas operaciones avanzadas con arrays, como filtrar, mapear y reducir, son herramientas poderosas que nos permiten manipular y transformar datos de

manera eficiente en JavaScript. Conocer su funcionamiento y aplicaciones nos ayudará a escribir código más conciso y legible.

#### Conclusión - Operaciones avanzadas con arrays: filtrar, mapear y reducir

En esta sección sobre operaciones avanzadas con arrays, hemos explorado cómo filtrar, mapear y reducir arrays en JavaScript. Hemos aprendido a utilizar métodos como `filter`, `map` y `reduce` para realizar estas operaciones de manera concisa y legible. Estos métodos nos permiten realizar transformaciones y cálculos complejos en los elementos de un array de manera eficiente. Con esta información, tenemos las herramientas necesarias para manipular y transformar arrays en JavaScript de manera avanzada.



# Ejercicios Practicos

Pongamos en práctica tus conocimientos

04 | Ejercicios Practicos

En esta lección, pondremos la teoría en práctica a través de actividades prácticas. Haga clic en los elementos a continuación para verificar cada ejercicio y desarrollar habilidades prácticas que lo ayudarán a tener éxito en el tema.

## Crear un array



Crea un array llamado 'frutas' que contenga los siguientes elementos: 'manzana', 'banana', 'naranja' y 'pera'. Imprime el array en la consola.

## Agregar y eliminar elementos



Crea un array llamado 'numeros' que contenga los números del 1 al 5. Agrega el número 6 al final del array y elimina el número 3. Imprime el array final en la consola.

## Filtrar números pares



Crea un array llamado 'numeros' que contenga los números del 1 al 10. Filtra los números pares y crea un nuevo array llamado 'numerosPares' con dichos números. Imprime el array 'numerosPares' en la consola.



# Resumen

Repasemos lo que acabamos de ver hasta ahora

- ✓ En esta introducción a los arrays en JavaScript, hemos aprendido los conceptos básicos de cómo crear y acceder a arrays, así como cómo iterar sobre ellos utilizando bucles for y forEach. También hemos explorado las diferentes formas de recorrer un array y acceder a sus elementos utilizando los índices. Los arrays son una herramienta fundamental en JavaScript y son ampliamente utilizados en el desarrollo web.
- ✓ En esta sección sobre la manipulación de arrays, hemos aprendido cómo agregar, eliminar y modificar elementos en un array en JavaScript. Hemos analizado métodos como push, pop, shift, unshift y splice, que nos permiten realizar estas operaciones de manera efectiva. También hemos explorado cómo utilizar el método slice para crear copias de arrays y cómo concatenar arrays utilizando el operador spread. Con esta información, ahora tenemos el conocimiento necesario para manipular arrays de manera eficiente en JavaScript.
- ✓ En esta sección sobre operaciones avanzadas con arrays, hemos explorado cómo filtrar, mapear y reducir arrays en JavaScript. Hemos aprendido a utilizar métodos como filter, map y reduce para realizar estas operaciones de manera concisa y legible. Estos métodos nos permiten realizar transformaciones y cálculos complejos en los elementos de un array de manera eficiente. Con esta información, tenemos las herramientas necesarias para manipular y transformar arrays en JavaScript de manera avanzada.



# Prueba

Comprueba tus conocimientos respondiendo unas preguntas

06 | Prueba

Pregunta 1/6

¿Cuál es la forma correcta de crear un array en JavaScript?

- ☐ `var array = [];`
  - ☐ `array = [];`
  - ☐ `var array = {}`
- 

Pregunta 2/6

¿Cómo se agrega un elemento al final de un array en JavaScript?



- ☐ `array.push(elemento);`
  - ☐ `array.add(elemento);`
  - ☐ `array.append(elemento);`
- 

Pregunta 3/6

¿Cuál de las siguientes opciones no es una forma de eliminar un elemento de un array en JavaScript?

- ☐ `array.splice(index, 1);`
  - ☐ `array.pop();`
  - ☐ `array.delete(index);`
- 

Pregunta 4/6

¿Cómo se modifica un elemento de un array en JavaScript?

- ☐ `array[index] = nuevoElemento;`
  - ☐ `array.replace(index, nuevoElemento);`
  - ☐ `array.modify(index, nuevoElemento);`
- 

Pregunta 5/6

¿Cuál de los siguientes métodos no se usa para filtrar elementos de un array en JavaScript?

- ☐ `array.filter(condición);`
- ☐ `array.find(condición);`

☐ `array.map(transformación);`

---

Pregunta 6/6

¿Cuál de los siguientes métodos se usa para reducir un array a un solo valor en JavaScript?

☐ `array.find(condición);`

☐ `array.map(transformación);`

☐ `array.reduce(función);`

---

Entregar

Conclusión

# Felicidades!

¡Felicitaciones por completar este curso! Has dado un paso importante para desbloquear todo tu potencial. Completar este curso no se trata solo de adquirir

conocimientos; se trata de poner ese conocimiento en práctica y tener un impacto positivo en el mundo que te rodea.



Comparte este curso