**Arreglos en JavaScript**

Un arreglo (también conocido como matriz) es una estructura de datos que se utiliza para almacenar y organizar una colección de elementos relacionados. Los elementos en un array pueden ser de cualquier tipo de datos, como números, cadenas de texto, objetos, funciones u otros arrays. Los arrays se utilizan comúnmente para agrupar y manipular datos de manera eficiente.

**Distintas formas de declarar Arreglos**

1. **Usando la Notación de Corchetes ([]):** Esta es la forma más común y directa para declarar arreglos.

**Un reloj digital en la pantalla

Descripción generada automáticamente con confianza media**

1. **Usando el Constructor Array:** Puedes utilizar el constructor Array para crear arreglos.

* Arreglo vacío
* Arreglo con un tamaño fijo. Los valores están indefinidos pero si indicamos los validos que tendrá



**Se imprimiría: (10) [vacío x 10] –>** Esta forma utilizada no es recomendable, ya que si hacemos un push, tendrá 6 validos. Para agregar elementos en los “espacios reservados” debemos acceder al índice.

* Arreglo inicializado



1. **Usando el Constructor Array.of:** El método Array.of crea una nueva instancia de array con el número exacto de elementos que le pases como parámetros.

1. **Usando el Constructor Array.from:** El método Array.from crea una nueva instancia de array a partir de un objeto iterable o un array-like.

****

1. **Arreglo multidimensional:** Es un Array que contiene uno o más Arrays como elementos. Estos Arrays anidados pueden representar estructuras de datos más complejas, como matrices o tablas, donde se organizan los datos en filas y columnas. Ejemplo de array bidimensional.

**Texto

Descripción generada automáticamente**

**Desestructuración de Arrays**

La desestructuración es una forma sintáctica de extraer valores de arrays (y objetos) y asignarlos a variables individuales de manera más concisa.

* **Desestructuración básica de arreglos**

**Texto

Descripción generada automáticamente**

Cada variable que colocamos en los corchetes hace referencia a las posiciones del Aray. Si quisiéramos omitir una posición, colocamos una coma.

* **Desestructuración con valores predeterminados:** Puedes proporcionar valores predeterminados en caso de que el array no tenga suficientes elementos.

**Texto

Descripción generada automáticamente con confianza media**

* **Desestructuración con SpreadOperator:** Puedes usar el operador rest (...) para capturar el resto de los elementos en un array

Texto

Descripción generada automáticamente

**Spread Operator (…)**

El operador de propagación se utiliza para expandir o "propagar" elementos de un array u objeto en una nueva estructura, o para combinar arrays y objetos. En arrays, se usa para crear copias superficiales y en objetos para combinar propiedades.

El operador de propagación no solo se limita a arrays y objetos, sino que también se puede utilizar con otras estructuras de datos como Set y Map.

**Spread Operator en arreglos**

**Texto

Descripción generada automáticamente**

**Excepción con el Spread Operator:** Al usar el operador de propagación en un **arreglo unidimensional**, se crea una copia superficial del array, no se toca la referencia del arreglo original y las modificaciones que hagamos en esa copia superficial, permanecerán solo en ella.

Texto

Descripción generada automáticamente

En cambio, cuando utilizamos el Spread Operator en un **arreglo multidimensional**, solo se realiza una copia superficial del primer nivel. Esto quiere decir que los arreglos internos, siguen apuntando a los mismos objetos.

Texto

Descripción generada automáticamente

En este ejemplo **originalArray** y **newArray** son arreglos diferentes, pero los arreglos internos siguen siendo las mismas referencias, por lo que modificar un array interno en **newArray** también afecta a **originalArray**.

**Array Methods y Callbacks**

Muchos métodos de arrays como forEach(), map(), filter(), some(), every(), find(), reduce(), entre otros, utilizan funciones de callback. Estas funciones de callback permiten personalizar la operación que deseas realizar con cada elemento del array. El callback se ejecuta una vez para cada elemento del array y puede recibir hasta tres argumentos:

**Elemento Actual:** El elemento del array que está siendo procesado.

**Índice:** El índice del elemento actual en el array.

**Array:** El array en sí mismo sobre el cual se está operando.

**Métodos más comunes para Arrays**

Los métodos de los arrays pueden utilizarse con todas las formas de declarar arreglos, ya sea que utilices la notación de corchetes ([]), el constructor Array, Array.of, Array.from, o incluso arreglos anidados. Los métodos son parte del prototipo Array, por lo que están disponibles para cualquier instancia de arreglo en JavaScript.

**Métodos de Consulta:** Estos métodos son útiles para obtener información sobre los elementos del array.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media**length**: Devuelve el número de elementos en el arreglo.

**every(callback):** Comprueba si todos los elementos del array cumplen con la función de prueba proporcionada. Devuelve un valor booleano.

Texto

Descripción generada automáticamente

**some(callback):** Comprueba si al menos un elemento del array cumple con la función de prueba proporcionada.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**indexOf(elemento, desdeIndice):** Devuelve el primer índice en el que se puede encontrar un elemento dado en el array, o -1 si no se encuentra. Si no se le pasa desde que índice, toma como referencia el índice 0.

Un reloj digital en la pantalla

Descripción generada automáticamente con confianza media

**toString():** Devuelve una cadena de texto representando el array.



**Métodos de Inserción:** Estos métodos te permiten agregar elementos a un array.

**push(elemento1, elemento2, …):** Añade uno o más elementos al final de un array.

Texto

Descripción generada automáticamente

**unshift(elemento1, elemento2, ...):** Añade uno o más elementos al principio de un array.

Texto

Descripción generada automáticamente

**splice(indice, cantidadAEliminar, elemento1, elemento2, ...):** Cambia el contenido de un array eliminando, reemplazando o añadiendo elementos. En el ejemplo se está utilizando para agregar un elemento al arreglo.

**Texto

Descripción generada automáticamente**

**Métodos de Eliminación:** Estos métodos te permiten eliminar elementos de un array.

**pop():** Elimina el último elemento de un array y lo devuelve.

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**shift():** Elimina el primer elemento de un array y lo devuelve.

Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja

**splice(indice, cantidadAEliminar):** Podemos eliminar el elemento/s a partir de un índice específico. También es utilizado para inserción y reemplazo.

Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja

**Métodos de Búsqueda:** Estos métodos se utilizan para encontrar elementos en un array.

**find(callback):** Devuelve el primer elemento del array que cumple con la función de prueba proporcionada.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**findIndex(callback):** Devuelve el índice del primer elemento que cumple con la función de prueba proporcionada.

Texto

Descripción generada automáticamente

**includes(elemento, desdeIndice):** Determina si un array contiene un elemento específico.

Un reloj digital en la pantalla

Descripción generada automáticamente con confianza media

**Métodos de Modificación:** Estos métodos modifican el contenido del array.

**splice(indice, cantidadAEliminar, elemento1, elemento2, …):** Cambia el contenido de un array eliminando, reemplazando o añadiendo elementos.

Un reloj digital en la pantalla

Descripción generada automáticamente con confianza baja

**sort(comparador):** El método sort() toma una función de comparación opcional llamada *comparador* que se usa para determinar el orden de los elementos en el array. La función de comparación recibe dos argumentos, a y b, y debe devolver un valor numérico que indica el orden de estos elementos. **Comparador** (opcional): Función que define el orden. Si no se proporciona, el array se ordena como cadenas de texto.

1. **Comparador para ordenar de manera ascendente:** Para ordenar números en orden ascendente, la función de comparación debe devolver la diferencia entre a y b.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

1. **Comparador para ordenar de manera descendente:** Para ordenar números en orden descendente, la función de comparación debe devolver la diferencia entre b y a.

Imagen que contiene Aplicación

Descripción generada automáticamente

1. Existen muchos comparadores tanto para objetos como para strings.

**reverse():** Invierte el orden de los elementos en un array in-place.

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**Métodos de Transformación:** Estos métodos crean un nuevo array o valor basado en el array original.

**map(callback):** Crea un nuevo array con los resultados de la llamada a una función proporcionada sobre cada elemento del array. Se le aplica la función a cada elemento del arreglo.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**filter(callback):** Crea un nuevo array con todos los elementos que pasan la prueba implementada por la función proporcionada. El predicado que le pasamos debe de retornar un boolean

Imagen que contiene Texto

Descripción generada automáticamente

**reduce(callback, valorInicial):** Aplica una función contra un acumulador y cada elemento del array para reducirlo a un solo valor.

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**flat(nivel):** Crea un nuevo array con todos los elementos de sub-array concatenados en un único array. **Nivel** la profundidad de la reducción. Por defecto es 1.

**Texto

Descripción generada automáticamente con confianza media**

**Formas para iterar sobre los elementos de un arreglo**

* **Usar un Bucle for:** El bucle for clásico te permite iterar sobre un array utilizando un índice. Es útil cuando necesitas un control total sobre la iteración.

Texto

Descripción generada automáticamente

* **Usar un Bucle for...of:** El bucle for...of es más moderno y está diseñado para iterar sobre los elementos de arrays y otras estructuras iterables. Es más limpio y fácil de leer que el bucle for clásico.

Texto

Descripción generada automáticamente

* **Usar el Método forEach():** El método forEach() ejecuta una función de callback para cada elemento del array. Este método no retorna un nuevo array y no se puede interrumpir la iteración (no se puede usar break o continue).

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media