METODOS DE ARRAY EN JAVASCRIPT

1. **map()** : Permite recorrer el array y modificar los elementos presentes en él, retornando un nuevo array con la misma longitud que el original.
2. **filter()** : Recorre el array y retorna un nuevo array con aquellos elementos que pasen una determinada condición.
3. **forEach()** : Permite iterar el contenido de un array. Recibe un callback que toma como parámetro el elemento actual de la iteración y el indice del mismo.
4. **find()** : Recorre el array y retorna la primera coincidencia del elemento que se busca.
5. **sort()** : Ordena los elementos del array y retorna el arreglo ordenado. Los elementos se ordenarán en orden ascendente (de la A a la Z) por defecto.
6. **some()** : Itera el array y retorna un booleano si como mínimo uno de los elementos presentes en el array pasa una condición determinada. Recibe un callback que se encargara de preguntar aquello que queremos dentro del array.
7. **every()** : Es similar al some(), ya que itera el array y retorna un booleano. Pero esta vez, para que dicho booleano sea true todos los elementos del array deberán pasar la condición dada.
8. **concat()** : Se utiliza para unir dos o más arrays. Este método no cambia los arrays existentes, sino que devuelve un nuevo array.
9. **includes()** : Determina si un array incluye un determinado elemento y retorna un booleano según corresponda.
10. **join()** : Une todos los elementos de un array en una cadena. Podemos pasarle como parámetro el carácter de separación que debe agregar entre los elementos.
11. **reduce()** : Aplica una función a un acumulador y a cada valor de una array (de izquierda a derecha) para reducirlo a un único valor.
12. **indexOf()** : Retorna el primer índice en el que se puede encontrar un elemento dado en el array, ó retorna -1 si el elemento no esta presente.
13. **findIndex()** : Retorna el índice del primer elemento de un array que cumpla con la función de prueba proporcionada. En caso contrario devuelve -1.
14. **fill()** : Cambia todos los elementos de un array por un valor estático, desde el índice de inicio hasta el índice final. Retorna el array modificado.
15. **push()** : Añade uno o más elementos al final de un array y devuelve la nueva longitud del array.
16. **pop()** : Elimina el último elemento de un array y lo devuelve. Este método cambia la longitud del array.
17. **shift()** : Elimina el primer elemento del array y lo retorna. Este método modifica la longitud del array.
18. **unshift()** : Agrega uno o más elementos al inicio del array, y devuelve la nueva longitud del array.
19. **slice()** : Devuelve una copia de una parte del array dentro de un nuevo array empezando por inicio hasta fin (fin no incluido). El array original no se modificará.
20. **reverse()** : Invierte el orden de los elementos de un array. El primer elemento pasa a ser el último y el último pasa a ser el primero.
21. **splice()** : Cambia el contenido de un array eliminando elementos existentes y/o agregando nuevos elementos.
22. **lastIndexOf()** : Busca un elemento en un array y devuelve su posición. Comienza buscando por el final del array. Retorna -1 si el elemento no se encontrara.
23. **flat()** : Crea una nuevo array con todos los elementos de sub-array concatenados recursivamente hasta la profundidad especificada.
24. **isArray()** : Determina si el valor pasado es un Array.
25. **from()** : Crea una nueva instancia de Array a partir de un objeto iterable.