30012025_FUNCIONES NATIVAS(TEORIA)

En JavaScript, existen varias funciones nativas para manejar arrays. Listado de las más utilizadas:

Métodos de creación y manipulación de arrays:

1. **Array.of()**: Crea una nueva instancia de array con un número variable de elementos pasados como argumentos.
2. **Array.from()**: Crea un nuevo array a partir de un objeto iterable (como una cadena o un conjunto).
3. **push()**: Añade uno o más elementos al final de un array y devuelve la nueva longitud del array.
4. **pop()**: Elimina el último elemento de un array y lo devuelve.
5. **shift()**: Elimina el primer elemento de un array y lo devuelve.
6. **unshift()**: Añade uno o más elementos al principio de un array y devuelve la nueva longitud del array.
7. **splice()**: Cambia el contenido de un array eliminando, reemplazando o añadiendo elementos.
8. **concat()**: Combina dos o más arrays y devuelve un nuevo array.
9. **slice()**: Devuelve una copia superficial de una porción de un array, sin modificar el array original.

Métodos de búsqueda y transformación:

10. **indexOf()**: Devuelve el primer índice en el que se encuentra un determinado elemento, o -1 si no lo encuentra.
11. **lastIndexOf()**: Devuelve el último índice en el que se encuentra un determinado elemento.
12. **includes()**: Determina si un array contiene un determinado elemento.
13. **find()**: Devuelve el primer elemento que cumple con la condición de la función de prueba.
14. **findIndex()**: Devuelve el índice del primer elemento que cumple con la condición de la función de prueba.
15. **map()**: Crea un nuevo array con los resultados de la función aplicada a cada elemento del array original.
16. **filter()**: Crea un nuevo array con todos los elementos que pasen la prueba de la función dada.
17. **reduce()**: Aplica una función a un acumulador y a cada elemento de un array (de izquierda a derecha), para reducirlo a un solo valor.
18. **reduceRight()**: Similar a `reduce()`, pero procesa los elementos de derecha a izquierda.
19. **forEach()**: Ejecuta una función en cada elemento de un array, pero no devuelve nada (solo efectos secundarios).

Métodos de ordenación y manipulación de elementos:

20. **sort()**: Ordena los elementos de un array en su lugar.
21. **reverse()**: Invierte el orden de los elementos de un array.
22. **fill()**: Rellena todos los elementos de un array con un valor estático, desde un índice inicial hasta un índice final.

Métodos de comparación y prueba:

23. **every()**: Determina si todos los elementos del array cumplen con una condición.
24. **some()**: Determina si al menos un elemento del array cumple con una condición.
25. **includes()**: Comprueba si un array contiene un elemento específico.
26. **join()**: Une todos los elementos de un array en una cadena, usando un separador especificado.

Métodos de transformación de estructuras:

27. **flat()**: Devuelve un nuevo array con todos los elementos de subarrays concatenados de manera recursiva.
28. **flatMap()**: Similar a `map()`, pero luego aplica `flat()` al resultado.

Métodos de iteración avanzada:

29. **keys()**: Devuelve un objeto Array Iterator que contiene las claves (índices) de los elementos de un array.
30. **values()**: Devuelve un objeto Array Iterator que contiene los valores de los elementos de un array.
31. **entries()**: Devuelve un objeto Array Iterator con pares clave-valor (índice y valor) de los elementos del array.