Metodos de arrays más usados

- **concat**: El método concat() se usa para unir dos o más arrays. Este método no cambia los arrays existentes, sino que devuelve un nuevo array.

Ejemplo:

```
const array1 = ['a', 'b', 'c'];
const array2 = ['d', 'e', 'f'];
const array3 = array1.concat(array2);

console.log(array3);
// expected output: Array ["a", "b", "c", "d", "e", "f"]
```

- **entries**: El método entries() retorna un nuevo objeto Array Iterator que contiene los pares clave/valor para cada índice de la matriz.

Ejemplo:

```
const array1 = ['a', 'b', 'c'];

const iterator1 = array1.entries();

console.log(iterator1.next().value);
// expected output: Array [0, "a"]

console.log(iterator1.next().value);
// expected output: Array [1, "b"]
```

- **every**: Determina si todos los elementos en el array satisfacen una condición. **Precaución**: ¡Llamar este método en un array vacío devuelve `true` para cualquier condición!

Ejemplo:

```
39
40  const  esMenor40 = (currentValue) => currentValue < 40;
41
42  const array1 = [1, 30, 39, 29, 10, 13];
43
44  console.log(array1.every(esMenor40));
45  // expected output: true
46
47
```

- **filter**: El método filter() crea un nuevo array con todos los elementos que cumplan la condición implementada por la función dada.

Ejemplo:

```
39
40  const palabras = ['perro', 'gato', 'elefante', 'aguila', 'leon', 'hipopotamo'];
41
42  const resultado = palabras.filter(palabra => palabra.length > 6);
43
44  console.log(resultado);
45  // salida esperada: Array ["elefante", "aguila", "hipopotamo"]
46
47
```

- **find**: El método find() devuelve el valor del primer elemento del array que cumple la función de prueba proporcionada.

Ejemplo:

```
39
40  const numeros = [5, 12, 8, 130, 44];
41
42  const encontrado = numeros.find(elemento => elemento > 10);
43
44  console.log(encontrado);
45  // salida esperada: 12
46
47
```

- **forEach**: El método forEach() ejecuta la función indicada una vez por cada elemento del array.

Ejemplo:

```
39
40  const caracteres = ['a', 'b', 'c'];
41
42  caracteres.forEach(elemento => console.log(elemento));
43
44  // salida esperada: "a"
45  // salida esperada: "b"
46  // salida esperada: "c"
47
48
```

- **includes**: El método includes() determina si una matriz incluye un determinado elemento, devuelve true o false según corresponda.

Ejemplo:

```
39
40  const numeros = [1, 2, 3];
41
42  console.log(numeros.includes(2));
43  // salida esperada: true
44
45  const animales = ['gato', 'perro', 'gallo'];
46
47  console.log(animales.includes('gato'));
48  // salida esperada: true
49
50  console.log(animales.includes('vaca'));
51  // salida esperada: false
52
```

- **indexOf**: El método indexOf() retorna el primer índice en el que se puede encontrar un elemento dado en el array, ó retorna -1 si el elemento no está presente.

Sintaxis: *array*.indexOf(*searchElement*[, *fromIndex*])

searchElement
Elemento a encontrar en el array.

fromIndex Opcional
Indica el índice por el que se comienza la búsqueda. Por defecto es 0, por lo que se busca en todo el array. Si el índice es mayor o igual a la longitud del array, devuelve -1, ya que no se buscaría en el array. Si el valor es negativo, se toma restando posiciones desde el final del array. Hay que tener en cuenta que aunque el índice sea negativo, la búsqueda seguirá realizándose en un orden incremental. Si el índice calculado es menor de 0, la búsqueda se realizará por todo el array.

Ejemplo:

```
39
40  const  numeros = [2, 9, 9];
41  numeros.indexOf(2);     // 0
42  numeros.indexOf(7);     // -1
43  numeros.indexOf(9, 2);  // 2
44  numeros.indexOf(2, -1); // -1
45  numeros.indexOf(2, -3); // 0
46
47
```

- **join**: El método join() une todos los elementos de una matriz (o un objeto similar a una matriz) en una cadena y devuelve esta cadena.

Ejemplo:

```
39
40  const elementos = ['Fuego', 'Aire', 'Agua', 'Tierra'];
41
42  console.log(elementos.join());
43  // salida esperada: "Fuego,Aire,Agua,Tierra"
44
45  console.log(elementos.join(''));
46  // salida esperada: "FuegoAireAguaTierra"
47
48  console.log(elementos.join('-'));
49  // salida esperada: "Fuego-Aire-Agua-Tierra"
50
```

- **keys**: El método keys() devuelve un nuevo objeto Array Iterator que contiene las claves de índice con las que acceder a cada elemento en el array.

Ejemplo:

```
40  const caracteres = ['a', 'b', 'c'];
41  const iterador = caracteres.keys();
42
43  for (const key of iterador) {
44    console.log(key);
45  }
46
47  // salida esperada: 0
48  // salida esperada: 1
49  // salida esperada: 2
50
```

- **map**: El método map() crea un nuevo array con los resultados de la llamada a la función indicada aplicados a cada uno de sus elementos.

Ejemplo:

```
39
40  const  numeros = [1, 5, 10, 15];
41  const  dobles = numeros.map( x =>  x * 2 );
42  // dobles ahora es [2, 10, 20, 30]
43  // numeros es aún [1, 5, 10, 15]
44
45  numeros = [1, 4, 9];
46  const  raiz = numeros.map( num =>  Math.sqrt(num) );
47  // raiz ahora es [1, 2, 3]
48  // numeros aún es [1, 4, 9]
49
50
```

- **pop**: El método pop() elimina el último elemento de un array y lo devuelve. Este método cambia la longitud del array.

Ejemplo:

```
40  const vegetales = ['brócoli', 'coliflor', 'repollo', 'pepino', 'tomate'];
41
42  console.log(vegetales.pop());
43  // salida esperada: "tomate"
44
45  console.log(vegetales);
46  // salida esperada: Array ["brócoli", "coliflor", "repollo", "pepino"]
47
48  vegetales.pop();
49
50  console.log(vegetales);
51  // salida esperada: Array ["brócoli", "coliflor", "repollo"]
52
```

- **push**: El método push() añade uno o más elementos al final de un array y devuelve la nueva longitud del array.

Ejemplo:

```
const animales = ['cerdo', 'cabra', 'oveja'];

const count = animales.push('vaca');
console.log(count);
// salida esperada: 4
console.log(animales);
// salida esperada: Array ["cerdo", "cabra", "oveja", "vaca"]

animales.push('pollo', 'gato', 'perro');
console.log(animales);
// salida esperada: Array ["cerdo", "cabra", "oveja", "vaca", "pollo", "gato", "perro"]
```

- **reduce**: El método reduce() ejecuta una función reductora sobre cada elemento de un array, devolviendo como resultado un único valor.

Ejemplo:

```
const numeros = [1, 2, 3, 4];

// 0 + 1 + 2 + 3 + 4
const valorInicial = 0;
const total = numeros.reduce(
  (valorAnterior, valorActual) => valorAnterior + valorActual,
  valorInicial
);

console.log(total);
// salida esperda: 10
```

- **reverse**: El método reverse() invierte el orden de los elementos de un array in place. El primer elemento pasa a ser el último y el último pasa a ser el primero.

Ejemplo:

```
39
40  const numeroString = ['one', 'two', 'three'];
41  console.log('numeroString:', numeroString);
42  // salida esperda: "numeroString:" Array ["one", "two", "three"]
43
44  const reversed = numeroString.reverse();
45  console.log('reversed:', reversed);
46  // salida esperda: "reversed:" Array ["three", "two", "one"]
47
48  // Cuidado: reverse es destructivo -- cambia el array original.
49  console.log('numeroString:', numeroString);
50  // salida esperda: "array1:" Array ["three", "two", "one"]
51
52
```

- **shift**: El método shift() elimina el primer elemento del array y lo retorna. Este método modifica la longitud del array.

Ejemplo:

```
40  const pescados = ['ángel', 'payaso', 'mandarín', 'cirujano'];
41
42  console.log('pescados antes: ' + pescados);
43  // "pescados antes: ángel,payaso,mandarín,cirujano"
44
45  const eliminado = pescados.shift();
46
47  console.log('pescados después: ' + pescados);
48  // "pescados after: payaso,mandarín,cirujano"
49
50  console.log('Elemento eliminado: ' + eliminado);
51  // "Elemento eliminado: ángel"
52
```

- **slice**: El método 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.

Ejemplo:

```
38
39  const  nombres = ['Rita', 'Pedro', 'Miguel', 'Ana', 'Vanesa'];
40  const  masculinos = nombres.slice(1, 3);
41
42  // masculinos contiene ['Pedro','Miguel']
43
```

- **some**: El método some() comprueba si al menos un elemento del array cumple con la condición implementada por la función proporcionada.

Ejemplo:

```
38
39  const numeros = [1, 2, 3, 4, 5];
40
41  // verificar si un elemento es pares
42  const pares = (elemento) => elemento % 2 === 0;
43
44  console.log(numeros.some(pares));
45  // salida esperada : true
46
47
48
```

- **sort**: El método sort() ordena los elementos de un arreglo (array) localmente y devuelve el arreglo ordenado. La ordenación no es necesariamente estable. El modo de ordenación por defecto responde a la posición del valor del string de acuerdo a su valor Unicode.

Ejemplo:

```
38
39  const frutas = ['mango', 'manzanas', 'bananas'];
40  frutas.sort(); // ['bananas', 'mango', 'manzanas']
41
42  const puntos = [1, 10, 2, 21];
43  puntos.sort(); // [1, 10, 2, 21]
44  // Tenga en cuenta que 10 viene antes que 2
45  // porque '10' viene antes que '2' según la posición del valor Unicode.
46
47  const cosas = ['word', 'Word', '1 Word', '2 Words'];
48  cosas.sort(); // ['1 Word', '2 Words', 'Word', 'word']
49  // En Unicode, los números vienen antes que las letras mayúsculas
50  // y estas vienen antes que las letras minúsculas.
51
52
```

- **splice**: El método splice() cambia el contenido de un array eliminando elementos existentes y/o agregando nuevos elementos.

Ejemplo:

```
39  const meses = ['Jan', 'March', 'April', 'June'];
40  meses.splice(1, 0, 'Feb');
41
42  // inserta en la posición 1
43  console.log(meses);
44  // salida esperada: Array ["Jan", "Feb", "March", "April", "June"]
45
46  meses.splice(4, 1, 'May');
47
48  // replaza el elemento 1 en la posición 4
49  console.log(meses);
50  // salida esperada: Array ["Jan", "Feb", "March", "April", "May"]
51
52
```

- **toString**: El método toString() devuelve una cadena de caracteres representando el array especificado y sus elementos.

Ejemplo:

```
36
37  const restar = (a, b) => a - b;
38
39  const numeros = [1, 2, 'a', '1a'];
40
41  console.log(numeros.toString());
42  // salida esperada: "1,2,a,1a"
43
```

- **unshift**: El método unshift() agrega uno o más elementos al inicio del array, y devuelve la nueva longitud del array.

Ejemplo:

```
39
40  const numeros = [1, 2, 3];
41
42  console.log(numeros.unshift(4, 5));
43  // salida esperada: 5
44
45  console.log(numeros);
46  // salida esperada: Array [4, 5, 1, 2, 3]
47
48
```

- **values**: El método values() devuelve un nuevo objeto Array Iterator que contiene los valores para cada índice del array.

Ejemplo:

```
39
40  const caracter = ['w', 'y', 'k', 'o', 'p'];
41  const iterador = caracter.values();
42
43  console.log(iterador.next().value); // w
44  console.log(iterador.next().value); // y
45  console.log(iterador.next().value); // k
46  console.log(iterador.next().value); // o
47  console.log(iterador.next().value); // p
48
49
```