Arrays - Métodos de callback

Todos los métodos que veremos a continuación reciben un callback con tres parámetros (Menos los tres últimos que son especiales)

```
cb(item, index, array)
```

Los nombres de los parámetros, como en cualquier función pueden recibir el nombre que nos dé la gana.

item: Es cada elemento del array. Es obligatorio.

index: Es su posición en el array. Es opcional

array: El array que estamos recorriendo. Es opcional. No se usa casi nunca.

Arrays - forEach

forEach(cb): Recorre cada uno de los elementos del array y aplica la función de callback.

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

numbers.forEach(number => {
  console.log(number * 2); // 2 4 6 8
});
```

Éste método no devuelve nada al ejecutarse, sólo ejecuta el código que le digamos.

```
const result = numbers.forEach(number => {
  console.log(number * 2); // 2 4 6 8
});

console.log(result); // undefined
```

Arrays - map

map(cb): Recorre cada uno de los elementos del array y aplica la función de callback.
Éste método devuelve un nuevo array con el resultado de la operación que le hayamos pedido

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

const result = numbers.map(number => {
  return number * 2;
});

console.log(result); // [2, 4, 6, 8]
```

```
const letters = ['a', 'b', 'c', 'd'];

const result = letters.map(letter => {
  return letter.toUpperCase();
});

console.log(result); // ['A', 'B', 'C', 'D']
```

Arrays - filter

filter(cb): Recorre cada uno de los elementos del array y filtra los valores que cumplan la condición del filtro
Éste método devuelve un nuevo array con los elementos que cumplan la condición de filtrado

```
const numbers = [1, 2, 3, 4, 5, 6];

const result = numbers.filter(number => {
  return number % 2 === 0;
});

console.log(result); // [2, 4, 6]
```

Arrays - every

every(cb): Recorre cada uno de los elementos del array y devuelve true o false si TODOS los elementos cumplen la condición establecida.

```
const numbers = [2, 4, 6, 8, 10];

const result = numbers.every(number => {
  return number % 2 === 0;
});

console.log(result); // true
```

Arrays - some

every(cb): Recorre cada uno de los elementos del array y devuelve true o false si ALGUNO de los elementos cumple la condición establecida.

```
const numbers = [2, 3, 5, 7, 9];

const result = numbers.some(number => {
  return number % 2 === 0;
});

console.log(result); // true
```

Arrays - find

find(cb): Recorre cada uno de los elementos del array y devuelve la primera coincidencia con la condición que le establezcamos.
IMPORTANTE: No devuelve un array, devuelve el valor

```
const numbers = [1, 2, 3, 4, 5, 6];

const result = numbers.find(number => {
  return number % 2 === 0;
});

console.log(result); // 2
```

Arrays - sort

sort((a,b)): Recorre cada uno de los elementos del array y los va seleccionando por parejas, por eso recibe dos parámetros a y b

```
const numbers = [1, 2, 3, 4, 5, 6];

const result = numbers.sort((a, b) => {
  console.log(a, b);
});
```

```
2 1
3 2
4 3
5 4
6 5
```

Arrays - sort

En cada vuelta lo que hará será comparar las parejas de números con la operación de comparación que establezcamos y en función de si el resultado es positivo, negativo o igual, sabrá cuál de los dos valores es mayor o menor, y así podrá ordenar el array de forma ascendente o descendente.

```
const numbers = [4, 2, 6];

const result = numbers.sort((a, b) => {
  console.log(a, b);
  return a - b;
});
```

```
2 4
6 2
6 4
▶ (3) [2, 4, 6]
```

```
const numbers = [4, 2, 6];

const result = numbers.sort((a, b) => {
  console.log(a, b);
  return b - a;
});
```

```
2 4
2 6 2
6 4
▶ (3) [6, 4, 2]
```

Al hacer la resta hay 3 opciones posibles

- número positivo - a > b
- número negativo - a < b
- 0 - a es igual a b

Con esta información reordena el array de menor a mayor

Arrays - sort

Si quisiéramos ordenar strings no podemos hacer una resta, tenemos que comparar con mayor o menor y decirle qué resultado enviar en cada caso.
Al hacerlo manual, nosotros decidimos qué considerará valor negativo y cuál positivo, si invertimos los return, ordenará alfabéticamente.

```
const letters = ['a', 'b', 'c', 'd'];

const result = letters.sort((a, b) => {
  if (a > b) return -1;
  if (a < b) return 1;
  return 0;
});
```

```
scripts
▶ (4) ['d', 'c', 'b', 'a']
```

```
const letters = ['a', 'b', 'c', 'd'];

const result = letters.sort((a, b) => {
  if (a > b) return 1;
  if (a < b) return -1;
  return 0;
});
```

```
scripts
▶ (4) ['a', 'b', 'c', 'd']
```

Arrays - reduce

reduce(acc,current): Se utiliza para iterar sobre un array y combinar todos los elementos en un solo valor. Permite ejecutar una función de reducción en cada elemento del array y acumular el resultado en un valor único.
En acc tenemos el valor acumulado de la operación, y en current tenemos el valor actual que se está recorriendo.

```
const numbers = [1, 2, 3, 4, 5, 6];

let result = 0;

for (const number of numbers) {
  result = result + number;
}
```

```
const result = numbers.reduce((acc, number) => {
  return acc + number;
});
```