Métodos de Primera Clase

`.forEach()` --> FOR...OF

Úsalo como un `for...of`. Simplemente recorre un array y haz lo que quieras en cada vuelta.
Te regala:

1. Cada valor del array
2. El índice
3. El array entero

```
array.forEach((valor, indice, array) => {
  // haz lo qe quieras aquí
  console.log(valor, indice)
})
```

`.find()` --> UN ELEMENTO

Úsalo cuando quieras encontrar UN SOLO ELEMENTO según una condición. **Siempre te devolverá el primero que encuentre.**

- NORMA 1: Para quete devuelva un elemento, el CALBACK TIENE QUE RETORNAR `true`, si no, significa que no ha encontrado ningñun elemento y te devolverá `undefined`

Te regala:

1. Cada valor del array
2. El índice
3. El array entero

```
array.find((valor, indice, array) => {
  return (condicion) ? true : false
})
```

```
array.find((valor, indice, array) => (condicion))
```

`.filter()` --> VARIOS ELEMENTOS

Úsalo cuando quieras FILTRAR UN ARRAY y obtener un NUEVO ARRAY con solo los elementos que cumplan una condición.

- NORMA 1: Para que el elemento que estés iterando se meta en el nuevo array, el callback TIENE QUE RETORNAR `true`, si no se cumple NINGUNO, devolverá un array vacío `[]`

```
array.filter((valor, indice, array) => {
  return (condicion) ? true : false
})
```

```
array.filter((valor, indice, array) => (condicion))
```

`.map()` --> MODIFICAR

Úsalo cuando quieras coger cada valor del array original y, después de modificarle lo que quieras guardarlo en UN NUEVO ARRAY

- NORMA 1: Siempre tienes queretornar algo en el callback, o tendrás un array con valores `undefined`
- CONSEJO 1: Piensa en el `return` como si fuera el `push` de lo que quieras meter en el nuevo array.
- CONSEJO 2: Siempre te devuelve un array con EL MISMO NÚMERO DE ELEMENTOS QUE HAN ENTRADO.
- CONSEJO 3: Cuando tengas que devolver objetos, devuelve siempre nuevas referencias escritas con las llaves y el `spread operator` para no guardar las referencias de los originales

```
const nuevoArray = array.map((valor, indice, array) => {
  return (modificación)
})
```

```
const nuevoArray = array.map((valor, indice, array) => {
  return (condición)
    ? { ...valor, modificación }
    : { ...valor }
})
```

`.some()` --> PREGUNTA SI ALGUNO CUMPLE

Úsalo cuando quieras preguntar solo si ALGUNO DE LOS ELEMENTOS DEL ARRAY cumple la condición.
Solo con que uno lo haga, ya devuelve `true`

- NORMA 1: Necesita un return de `true` o `false` dependiendo de una condición.

```
const verdaderoFalso = array.some((valor, indice, array) => {
  return (condición)
})
```

`.every()` --> PREGUNTA SI TODOS CUMPLEN

Úsalo cuando quieras preguntar si TODOS LOS ELEMENTOS DEL ARRAY cumplen la condición.
Todos tienen que hacerlo paraque devuelva `true`

- NORMA 1: Necesita un return de `true` o `false` dependiendo de una condición.

```
const verdaderoFalso = array.every((valor, indice, array) => {
  return (condición)
})
```

`.sort()` --> ORDENAR

Ordena elementos según el código `Unicode(ASCII)` que tenga el carácter.

¡Ojo! porque el método `.sort()` **muta el array original, siempre intenta hacerlo con un duplicado**

`const nuevoArrayOrdenado = [...array].sort()`

- NORMA 1: El callback que le pasemos necesita 2 parámetros que van a ser siempre los valores segundo y primero que va a ir comparando. En la primera vuelta seran el primero y el segundo, luego el segundo y el tercero, y así sucesivamente.
- NORMA 2: Para que funcione el método necesita:
  - Retornar un valro negativo (-1) si el primer valor es más pequeño.
  - Retornar un valor positivo (1) si el primer valor es más grande.
  - Retornar 0 si los dos valores son iguales.

Con palabras

```
const arrayOrdenado = array.sort((primerValor, segundoValor) => {
  if (primerValor < segundoValor) return -1
  else if (primerValor > segundoValor) return 1
  else return 0
})
// Si lo queremos ordenado a la inversa, cambiamos el -1 por el 1
```

Con números

```
const arrayNumsOrdenado = array.sort((primerValor, segundoValor) => {
  return primerValor - segundoValor
})
// si queremos ordenado a la inversa cambiamos el orden del
primerValor por el segundoValor
```