EJERCICIOS JAVASCRIPT

MÉTODOS DE ARRAYS

Realizado por Tomás Cabello Fernández

Desarrollo Web en Entorno Servidor

Desarrollo de Aplicaciones Web

MÉTODOS DE ARRAYS

1. **Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.**

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

const resultado = numeros.map(num => num * 2);

console.log(resultado);
```

```
> const numeros = [1, 2, 3];
undefined
> const resultado = numeros.map(num => num * 2);
undefined
> console.log(resultado);
[ 2, 4, 6 ]
undefined
```

Al hacer el método map con el array números, hacemos dentro del metodo una función que coge cada valor de dentro del array y lo multiplica por dos, y por ultimo lo mete en otro array resultado.

2. **Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.**

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

const resultado = numeros.filter(num => num % 2 === 0);

console.log(resultado);
```

```
> const numeros = [1, 2, 3, 4, 5, 6];
undefined
> const resultado = numeros.filter(num => num % 2 === 0);
undefined
> console.log(resultado);
[ 2, 4, 6 ]
undefined
```

Al hacer el método filter y al hacer la función dentro del método que calcula el resto de cada numero del arreglo y lo compara con 0, si es cero significa que ese numero es par, con lo cual lo que se guarda en el array resultado son únicamente los numero pares del array números.

3. **Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.**

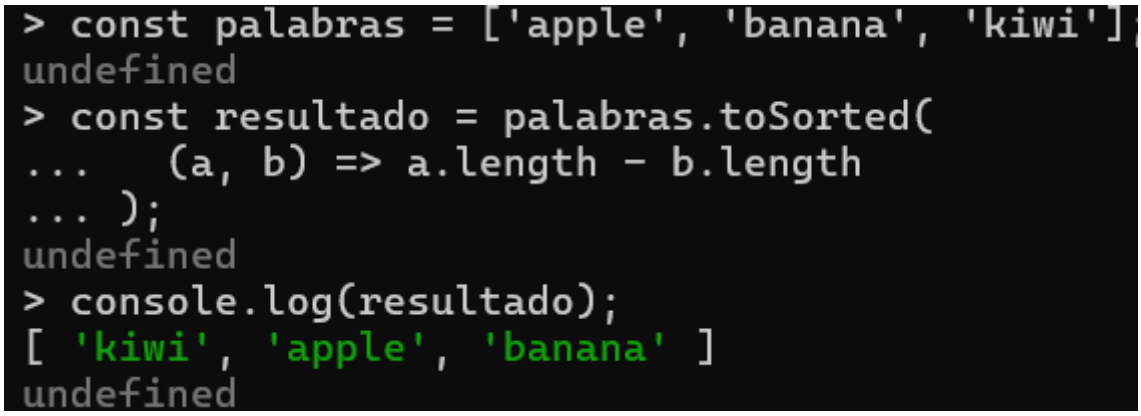
```
const palabras = ['apple', 'banana', 'kiwi'];

const resultado = palabras.toSorted(

 (a, b) => a.length - b.length

);

console.log(resultado);
```



El código mete en el array resultado las palabras contenidas en el array palabras pero de forma ordenada, esto lo hace con el método toSorted(), con una función que resta a – b, si a es mayor que b entonces a se colocará detrás de b y lo mismo de la forma contraría.

4. **Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.**

```
const palabras = ['hola', 'mundo'];

const resultado = palabras.map(

 palabra => palabra.toUpperCase()

);

console.log(resultado);
```

```
> const palabras = ['hola', 'mundo'];
undefined
> const resultado = palabras.map(
...     palabra => palabra.toUpperCase()
... );
undefined
> console.log(resultado);
[ 'HOLA', 'MUNDO' ]
undefined
```

Se recorre el array palabras con el método map y se coge palabra como variable y se hace una función con ella que lo que hace es poner cada elemento del nuevo array en mayúsculas.

5. **Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.**

```
const edades = [15, 22, 17, 19];

const adultos = edades.filter(edad => edad >= 18);

console.log(adultos);
```

```
> const edades = [15, 22, 17, 19];
undefined
> const adultos = edades.filter(edad => edad >= 18);
undefined
> console.log(adultos);
[ 22, 19 ]
undefined
```

Hace un nuevo arreglo y usa el método filter con el array edades en el que se compara cada elemento del primer array y si el elemento es mayor o igual a 18 lo añade al nuevo array, los demás los excluye.

6. **Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.**

```
const numeros = [3, 1, 4, 1, 5];

const resultado = numeros.sort((a, b) => b - a);

console.log(resultado);
```

```
> const numeros = [3, 1, 4, 1, 5];
undefined
> const resultado = numeros.sort((a, b) => b - a);
undefined
> console.log(resultado);
[ 5, 4, 3, 1, 1 ]
undefined
```

Similar al código del ejercicio 3 pero esta vez con el método sort y cambiando el sentido de la comparación es decir va a crear el array resultado de manera ordenada de mayor a menor, y esta vez con números.

7. **Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.**

```
const palabras = ['hola', 'mundo'];

const resultado = palabras.map(palabra => palabra.length);

console.log(resultado);
```

```
> const palabras = ['hola', 'mundo'];
undefined
> const resultado = palabras.map(palabra => palabra.length);
undefined
> console.log(resultado);
[ 4, 5 ]
undefined
```

Recorre el primer array con el método map y crea una variable en la función a la que le pasamos el método length que devuelve la cantidad de caracteres de cada elemento, esta cantidad se almacena en cada elemento del nuevo array.

8. **Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.**

```
const frutas = ['albaricoque', 'banana', 'uva'];

const resultado = frutas.filter(

 fruta => 'aeiou'.includes(fruta[0].toLowerCase())

);

console.log(resultado);
```

```
> const frutas = ['albaricoque', 'banana', 'uva'];
undefined
> const resultado = frutas.filter(
...   fruta => 'aeiou'.includes(fruta[0].toLowerCase())
... );
undefined
> console.log(resultado);
[ 'albaricoque', 'uva' ]
undefined
```

Este código va a guardar en el array resultado todas las palabras que empiecen por alguna vocal, se hace con el método filter y comparando la primera letra de cada elemento comparando aeiou con el método includes y convertiendo a minúsculas con toLowerCase para evitar fallos a la hora de comparar la primera letra de cada palabra.

9. **Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.**

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

const pos = numeros.filter(num => num > 0);

console.log(pos);

```
> const numeros = [-1, 2, -3, 4];
undefined
> const pos = numeros.filter(num => num > 0);
undefined
> console.log(pos);
[ 2, 4 ]
undefined
```

Este código es sencillo, crea un array pos en el se guardan los números mayor a cero utilizando el método filter e indicándole que compare cada elemento del array números para que sea mayor a cero.

10. Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.

```
const personas = [
 { nombre: 'Alicia', edad: 30 },
 { nombre: 'Roberto', edad: 25 }
];
const resultado = personas.sort((a, b) => a.edad - b.edad);
console.log(resultado);
```

```
> const personas = [
...   { nombre: 'Alicia', edad: 30 },
...   { nombre: 'Roberto', edad: 25 }
... ];
undefined
> const resultado = personas.sort((a, b) => a.edad - b.edad);
undefined
> console.log(resultado);
[ { nombre: 'Roberto', edad: 25 }, { nombre: 'Alicia', edad: 30 } ]
undefined
```

Este código guarda en la variable resultado el array de objetos personas pero esta vez ordenando cada objeto por edad con el método sort y restando edad a y edad b, si edad a es mayor que edad b, se pondrá el objeto de edad a en el lugar del objeto de edad b.

11. Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.

```
const nums = [5, 8, 12, 3, 7, 1, 14];
const resultado = nums
 .filter(num => num % 2 !== 0)
 .map(num => num * 2)
 .sort((a, b) => a - b);
console.log(resultado);
```

```
> const nums = [5,8,12,3,7,1,14]
undefined
> const resultado = nums.filter(num => % 2 !== 0).map(num => num * 2).sort((a,b) => a - b);
const resultado = nums.filter(num => % 2 !== 0).map(num => num * 2).sort((a,b) => a - b);
                                       ^

Uncaught SyntaxError: Unexpected token '%'
> const resultado = nums.filter(num => num % 2 !== 0).map(num => num * 2).sort((a,b) => a - b);
undefined
> console.log(resultado)
[ 2, 6, 10, 14 ]
undefined
```

Este código lo que hace es filtrar primeramente que números son impares, después los multiplica por dos, los ordena de menor a mayor y por ultimo los mete en un nuevo array resultado.

12. Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.

```
const personas = [
 { nombre: "Alice", edad: 25 },
 { nombre: "Bob", edad: 35 },
 { nombre: "Charlie", edad: 30 },
 { nombre: "David", edad: 40 }
];

const resultado = personas.filter(person => person.edad > 30).map(person => person.nombre.toUpperCase());
console.log(resultado);
```

```
> const personas = [
...   { nombre: "Alice", edad: 25 },
...   { nombre: "Bob", edad: 35 },
...   { nombre: "Charlie", edad: 30 },
...   { nombre: "David", edad: 40 }
... ];
undefined
> const resultado = personas.filter(person => person.edad > 30).map(person => person.nombre.toUpperCase());
undefined
> console.log(resultado);
[ 'BOB', 'DAVID' ]
undefined
```

Lo que hace este trozo de código es filtrar la edad de cada objeto del array personas y comprobar si es mayor de 30, si se cumple la condición se convierte el nombre de esa persona a mayúsculas y se guarda en el array resultado.

13. Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.

```
const nums = [4, 15, 8, 23, 11, 5, 19];

const resultado = nums.filter(num => num > 10).map(num => num * 2).sort((a, b) => b - a);

console.log(resultado);
```

```
> const nums = [4, 15, 8, 23, 11, 5, 19];
undefined
> const resultado = nums.filter(num => num > 10).map(num => num * 2).sort((a, b) => b - a);
undefined
> console.log(resultado);
[ 46, 38, 30, 22 ]
undefined
```

El código del ejercicio 13 primeramente filtra cada elemento del array nums y coge únicamente los números mayores a 10, después los multiplica por dos y los ordena de mayor a menor para después almacenarlos en el array resultado.

14. Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.

```
const nums = [-2, 5, -7, 3, -1, 8];

const resultado = nums.filter(num => num < 0).map(num => num * num).sort((a, b) => a - b);

console.log(resultado);
```

```
> const nums = [-2, 5, -7, 3, -1, 8];
undefined
> const resultado = nums.filter(num => num < 0).map(num => num * num).sort((a, b) => a - b);
undefined
> console.log(resultado);
[ 1, 4, 49 ]
undefined
```

En este código, en resultado se guardará la primero los números negativos, después los multiplica por si mismo y después los ordena de menor a mayor.

15. Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.

```
const personas = [

 { nombre: "Alicia", edad: 25 },

 { nombre: "Roberto", edad: 35 },

 { nombre: "Carlos", edad: 30 },
```

{ nombre: "David", edad: 40 }

];

const resultado = personas.filter(persona => persona.nombre.length > 5).map(persona => persona.edad * 2).sort((a, b) => a - b);

console.log(resultado);

```
> const personas = [
...   { nombre: "Alicia", edad: 25 },
...   { nombre: "Roberto", edad: 35 },
...   { nombre: "Carlos", edad: 30 },
...   { nombre: "David", edad: 40 }
... ];
undefined
> const resultado = personas.filter(persona => persona.nombre.length > 5).map(persona => persona.edad * 2).sort((a, b) =
> a - b);
undefined
> console.log(resultado);
[ 50, 60, 70 ]
undefined
```

En este caso primeramente se filtra el nombre de cada persona que esta dentro del array de objetos personas, si el numero de caracteres del nombre es mayor a 5 después multiplicará la edad por 2 y por ultimo la ordena y lo guarda en la variable resultado.

16. Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.

const nombres = ["Ángel", "Anabel", "Eva", "Ana", "Elena", "david"];

const resultado = nombres.filter(nombre => nombre.length > 4).sort();

console.log(resultado);

```
> const nombres = ["Ángel", "Anabel", "Eva", "Ana", "Elena", "david" ];
undefined
> const resultado = nombres.filter(nombre => nombre.length > 4).sort();
undefined
> console.log(resultado);
[ 'Anabel', 'Elena', 'david', 'Ángel' ]
undefined
```

Esta parte del código es sencilla ya que lo único que hace es comparar la longitud de cada nombre del array nombres, si es mayor a 4 lo ordena alfabéticamente con el método sort en el array resultado.

17. Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.

const nombres = ["Ángel", "Anabel", "Eva", "Ana", "Elena", "david"];

const resultado = nombres.filter(nombre => nombre.length > 4).sort((a, b) => a - b);

console.log(resultado);

```
> const nombres = ["Ángel", "Anabel", "Eva", "Ana", "Elena", "david" ];
undefined
> const resultado = nombres.filter(nombre => nombre.length > 4).sort((a, b) => a - b);
undefined
> console.log(resultado);
[ 'Ángel', 'Anabel', 'Elena', 'david' ]
undefined
```

Este código hace exactamente lo mismo que el código anterior, solo qué a la forma de ordenar, ordena de la palabra con menos caracteres a la palabra con mas caracteres, en lugar de alfabéticamente.

18. Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.

const nombres = ["Ángel", "Anabel", "Eva", "Ana", "elena", "David"];

const resultado = nombres.filter(nombre => nombre.length > 4).sort((a, b) => a.localeCompare(b));

console.log(resultado);

```
> const nombres = ["Angel", "Anabel", "Eva", "Ana", "elena", "David" ];
undefined
> const resultado = nombres.filter(nombre => nombre.length > 4).sort( (a, b) => a.localeCompare(b));

undefined
> console.log(resultado);
[ 'Anabel', 'Ángel', 'David', 'elena' ]
undefined
```

Este código hace exactamente lo mismo que el código anterior, solo qué a la forma de ordenar utiliza a.localCompare(b) que ordena alfabéticamente teniendo en cuanto las tildes y los caracteres especiales de cada región.

19. Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.

const nums = [-10, 5, -3, 8, -7];

const resultado = nums.map(num => Math.abs(num)).sort((a, b) => a - b);

console.log(resultado);

```
> const nums = [-10, 5, -3, 8, -7];
undefined
> const resultado = nums.map(num => Math.abs(num)).sort((a, b) => a - b);
undefined
> console.log(resultado);
[ 3, 5, 7, 8, 10 ]
undefined
```

En esta función recorre el array convirtiendo cada numero del array nums en positivo, es decir si es positivo se queda tal cual el número, pero si es negativo, lo cambia a positivo, después lo ordena de menor a mayor con el método sort y lo almacena en resultado.

20. **Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.**

```
const jugadores = [
 { nombre: "Alicia", score: 4 },
 { nombre: "Roberto", score: 7 },
 { nombre: "Carlos", score: 9 },
 { nombre: "David", score: 3 }
];
const resultado = jugadores.filter(jugador => jugador.score > 5)  .map(jugador => jugador.nombre).sort( (a, b) => a.localeCompare(b) );
console.log(resultado);
```

```
> const jugadores = [
...   { nombre: "Alicia", score: 4 },
...   { nombre: "Roberto", score: 7 },
...   { nombre: "Carlos", score: 9 },
...   { nombre: "David", score: 3 }
... ];
undefined
> const resultado = jugadores.filter(jugador => jugador.score > 5)  .map(jugador => jugador.nombre).sort( (a, b) => a.lo
caleCompare(b) );
undefined
> console.log(resultado);
[ 'Carlos', 'Roberto' ]
undefined
```

Básicamente lo que hace el código es filtrar que jugador del array objetos jugadores, tiene un score mayor a 5, si es mayor, se ordenaran los jugadores que cumplan la condición con localeCompare para que se ordene alfabéticamente teniendo en cuenta los caracteres especiales de cada región.

21. **Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.**

```
const nums = [4, 10, 5, 12, 6];
const resultado = nums.map(num => (num % 2 === 0 ? num * 2 : num)).filter(num => num > 15);
console.log(resultado);
```

```
> const nums = [4, 10, 5, 12, 6];
undefined
> const resultado = nums.map(num => (num % 2 === 0 ? num * 2 : num)).filter(num => num > 15);
undefined
> console.log(resultado);
[ 20, 24 ]
undefined
```

Este código compara cada valor del array nums y mira si son pares o no, si lo son los multiplica por dos y por ultimo compara si el resultado de la función es mayor a 15, si es así lo almacena en el array resultado.

22. Describe la operación u operaciones realizadas sobre el array inicial y los valores mostrados por pantalla finalmente.

const nombres = ["Ángel", "Anabel", "Eva", "Ana", "elena", "David"];

const resultado = nombres.filter(nombre => nombre.length > 2).sort((a, b) => a.localeCompare(b)).reverse();

console.log(resultado);

```
> const nombres = ["Ángel", "Anabel", "Eva", "Ana", "elena", "David" ];
undefined
> const resultado = nombres.filter(nombre => nombre.length > 2).sort( (a, b) => a.localeCompare(b) ).reverse();

undefined
> console.log(resultado);
[ 'Eva', 'elena', 'David', 'Ángel', 'Anabel', 'Ana' ]
undefined
```

En realidad este código solo filtra alfabéticamente por region con localCompare pero de manera inversa con el método reverse, es así por en la primera parte del código donde se compara el numero de caracteres de cada palabra, resulta que es siempre verdadero, porque cada palabra tiene mas de 2 caracteres.

TOMÁS CABELLO FERNÁNDEZ