Report Day: 5.3

Submitted By: Ayesha Abid

Submitted To: Mr. Ali Hyder Hidayat

Group: Frontend Development @ProSensia

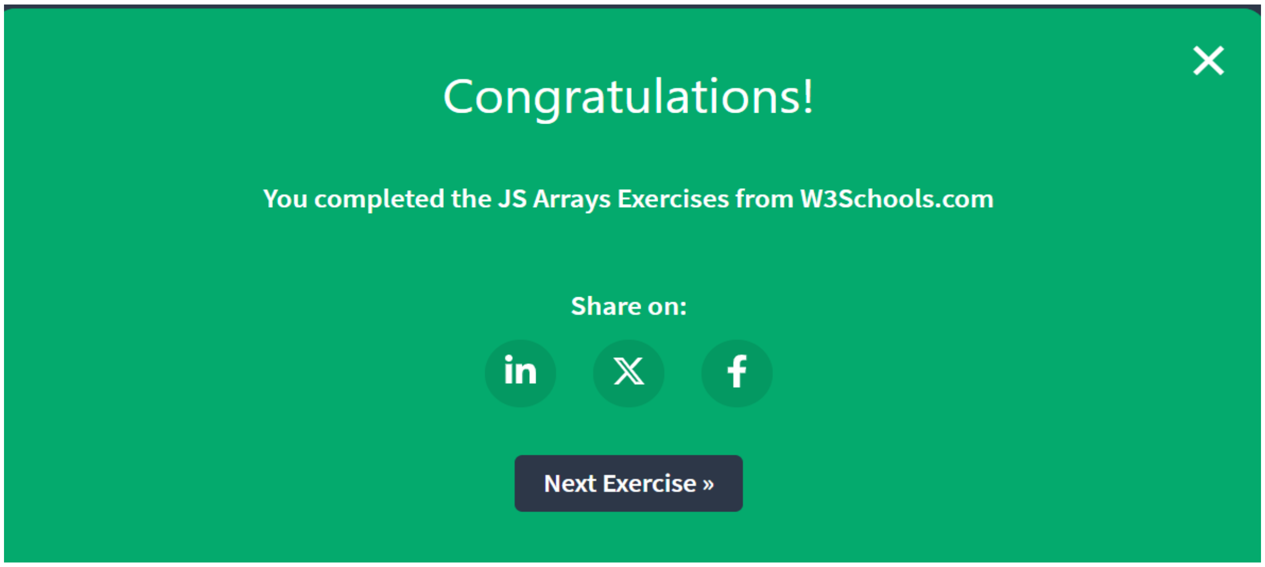
Dated: 23/07/25

Task: Array Methods (Map, Filter, Reduce)

An Array is an object type designed for storing data collections.

Key characteristics of JavaScript arrays are:

- **Elements**: An array is a list of values, known as elements.
- **Ordered**: Array elements are ordered based on their index.
- **Zero indexed**: The first element is at index 0, the second at index 1, and so on.
- **Dynamic size**: Arrays can grow or shrink as elements are added or removed.
- **Heterogeneous**: Arrays can store elements of different data types (numbers, They are part of the **functional programming** style in JavaScript and make code **shorter, cleaner, and more readable**.


Congratulations!

You completed the JS Arrays Exercises from W3Schools.com

Share on:

Next Exercise »

1. map() Method

Purpose:

The map() method creates a **new array** by applying a **function to each element** of the original array.

Syntax:

```
array.map(function(element, index, array) {

  // return new value

});
```

Example:

```
let numbers = [1, 2, 3];

let doubled = numbers.map(num => num * 2);

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

Use Cases:

- Transforming data (e.g., converting Celsius to Fahrenheit)
- Creating a list of modified objects
- Rendering UI elements from data

2. filter() Method

Purpose:

The filter() method returns a **new array containing only the elements that pass a specific condition** (i.e., for which the callback function returns true).

Syntax:

```
array.filter(function(element, index, array) {

  // return condition
});
```

Example:

```
let ages = [18, 22, 15, 30];

let adults = ages.filter(age => age >= 18);

console.log(adults); // [18, 22, 30]
```

Use Cases:

- Filtering valid or required data
- Searching with conditions
- Removing unwanted elements

3. reduce() Method

Purpose:

The reduce() method applies a **function to each element**, resulting in **a single output value** (e.g., sum, product, or object).

Syntax:

```
array.reduce(function(accumulator, currentValue, index, array) {

  // return updated accumulator

}, initialValue);
```

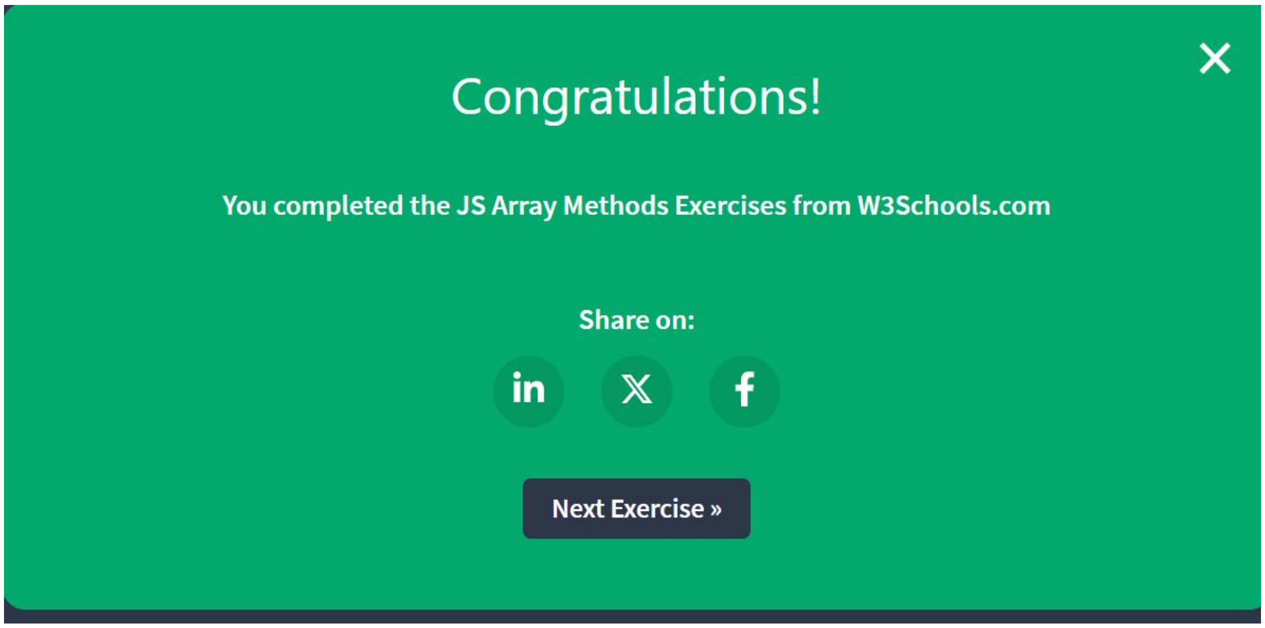
Example:

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

let total = numbers.reduce((sum, num) => sum + num, 0);

console.log(total); // 10
```

Use Cases:

- Summing numbers

- Flattening arrays
- Counting values
- Building objects from arrays



Congratulations!

You completed the JS Array Methods Exercises from W3Schools.com

Conclusion:

The map(), filter(), and reduce() methods are essential tools in modern JavaScript programming. They allow you to **manipulate, filter, and reduce arrays efficiently**, encouraging a cleaner and more functional coding style. Mastering these methods is crucial for working with data and building complex JavaScript applications.