1.7 Array in Javascript

Table of content

1. Arrays Fundamentals

An array is an object that can store multiple values at once. For example,

```
const words = ['hello', 'world', 'welcome'];
```

Here, `words` is an array. The array is storing 3 values.

▼ **Creating an Array:**

- The easiest way to create an array is by using an array literal `[]`. For example,

  ```
  const array1 = ["eat", "sleep"];
  ```

- You can also create an array using JavaScript's `new` keyword.

  ```
  const array2 = new Array("eat", "sleep");
  ```

> **Note**: Array's index starts with 0, not 1.

We can use the `length` property to find the length of the array.

▼ **Some Common Array Methods**

- `push()` adds a new element to the end of an array and returns the new length of an array
- `pop()`: removes the last element of an array and returns the removed element
- `forEach()`: calls a function for each element
- `sort()`: sorts the elements alphabetically in strings and in ascending order

- `includes()`: checks if an array contains a specified element
- `indexOf()`: searches an element of an array and returns its position
- `splice()`: removes or replaces existing elements and/or adds new elements

2. Advanced Array Methods

All these are higher-order functions. A higher-order function is a function that either takes other functions as arguments or returns a function.

▼ **Array filter method**

The `filter` () method returns a new array with all elements that pass the test defined by the given function.

```
let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

// function to check even numbers
function checkEven(number) {
  if (number % 2 == 0)
    return true;
  else
    return false;
}

// create a new array by filter even numbers from the nu
mbers array
let evenNumbers = numbers.filter(checkEven);
console.log(evenNumbers);

// Output: [ 2, 4, 6, 8, 10 ]
```

- `filter()` does not change the original array.
- `filter()` does not execute `callback` for array elements without values

▼ **Array map method**

The `map` () method creates a new array with the results of calling a function for every array element.

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

// function to return the square of a number
function square(number) {
  return number * number;
}

// apply square() function to each item of the numbers l
ist
let square_numbers = numbers.map(square);
console.log(square_numbers);

// Output: [ 4, 16, 36, 64, 100 ]
```

- `map()` does not change the original array.
- `map()` executes `callback` once for each array element in order.
- `map()` does not execute `callback` for array elements without values.

▼ Array find method

```
const emp = [
    {
        name: "Ram",
        empID: 101
    },
    {
        name: "Sham",
        empID: 102
    },
    {
        name: "Mohan",
        empID: 103
    }
];

const res = emp.find(el => el.empID === 102);
```

```
console.log("res is: ", res); // res is: {name: 'Sham',
empID: 102}
```

▼ **Array reduce method**

The `.reduce()` method in JavaScript is one of the most **powerful** array methods. It is used to **accumulate** values and **reduce** an array into a **single value** (number, object, array, etc.).

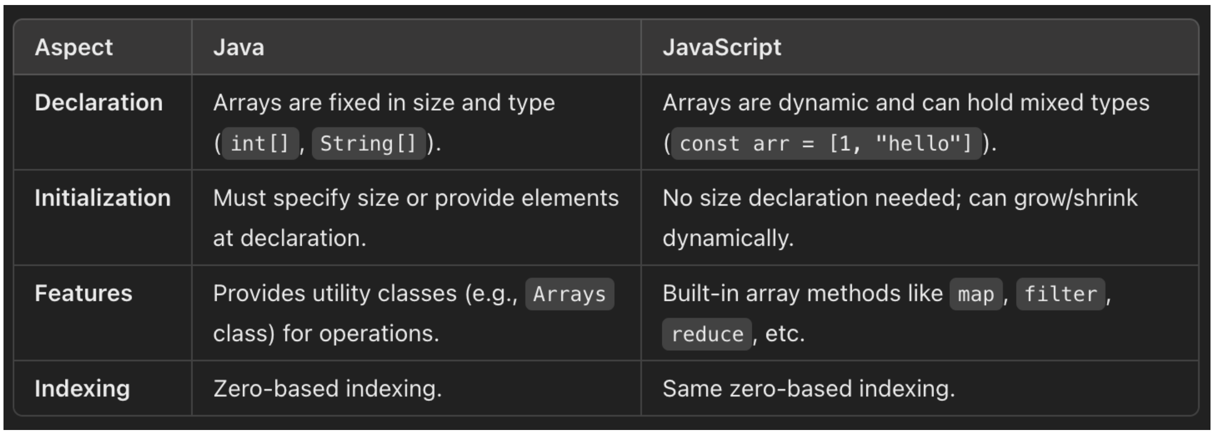
```
array.reduce(callback, initialValue);
```

Parameters:

1. `callback` (Required) – A function that runs on each array element. It takes:
    - `accumulator` – Stores the accumulated result.
    - `currentValue` – The current array element.
    - `index` (Optional) – The index of the current element.
    - `array` (Optional) – The original array.
2. `initialValue` (Optional) – The starting value of the accumulator. *If omitted, the first array element is used.*

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

const sum = numbers.reduce((acc, curr) => acc + curr, 0);
console.log(sum); // 👉 15
```



Aspect	Java	JavaScript
Declaration	Arrays are fixed in size and type (`int[]`, `String[]`).	Arrays are dynamic and can hold mixed types (`const arr = [1, "hello"]`).
Initialization	Must specify size or provide elements at declaration.	No size declaration needed; can grow/shrink dynamically.
Features	Provides utility classes (e.g., `Arrays` class) for operations.	Built-in array methods like `map`, `filter`, `reduce`, etc.
Indexing	Zero-based indexing.	Same zero-based indexing.

Assignments

1. Write a JavaScript function that returns a passed string with letters in alphabetical order.
2. Write a JavaScript function to calculate the average of marks passed in an array.
3. Write a Javascript function to remove duplicates from an array using reduce method
4. For the following input, use the reduce method to Group Items by Category

```
const products = [
    { name: "Laptop", category: "Electronics" },
    { name: "Shirt", category: "Clothing" },
    { name: "Phone", category: "Electronics" },
    { name: "Shoes", category: "Clothing" },
];
```

Use the reduce method to return the following:

```
{
    Electronics: ["Laptop", "Phone"],
    Clothing: ["Shirt", "Shoes"]
}
```