JavaScript provides a wide range of array methods that make it easy to work with arrays. Here's a detailed overview of commonly used array methods along with real-world examples:

**1. push() -**Adds one or more elements to the end of the array and returns the new length.

const fruits = ["Apple", "Banana"];

fruits.push("Orange");

console.log(fruits); // ["Apple", "Banana", "Orange"]

**2. pop() -** Removes the last element from an array and returns it.

const fruits = ["Apple", "Banana", "Orange"];

const lastFruit = fruits.pop();

console.log(lastFruit); // "Orange"

console.log(fruits); // ["Apple", "Banana"]

**3. unshift() -**Adds one or more elements to the beginning of an array and returns the new length.

const fruits = ["Banana", "Orange"];

fruits.unshift("Apple");

console.log(fruits); // ["Apple", "Banana", "Orange"]

**4. shift() -** Removes the first element from an array and returns it.

const fruits = ["Apple", "Banana", "Orange"];

const firstFruit = fruits.shift();

console.log(firstFruit); // "Apple"

console.log(fruits); // ["Banana", "Orange"]

**5. map()** Creates a new array by applying a function to each element in the original array.

const numbers = [1, 2, 3];

const squares = numbers.map(num => num \* num);

console.log(squares); // [1, 4, 9]

**6. filter() -**Creates a new array with elements that pass a condition.

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

const evenNumbers = numbers.filter(num => num % 2 === 0);

console.log(evenNumbers); // [2, 4]

**7. reduce()** Reduces an array to a single value by applying a function.

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

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

console.log(sum); // 10

**8. forEach() -**Executes a provided function once for each array element.

const fruits = ["Apple", "Banana", "Orange"];

fruits.forEach(fruit => console.log(fruit));

// Output:

// Apple

// Banana

// Orange

**9. find() -**Returns the first element that satisfies the provided condition.

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

const firstEven = numbers.find(num => num % 2 === 0);

console.log(firstEven); // 2

**10. findIndex()** Returns the index of the first element that satisfies the condition.

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

const index = numbers.findIndex(num => num > 2);

console.log(index); // 2

**11. some()** Checks if at least one element satisfies the condition.

const numbers = [1, 2, 3];

const hasEven = numbers.some(num => num % 2 === 0);

console.log(hasEven); // true

**12. every()-**Checks if all elements satisfy the condition.

const numbers = [2, 4, 6];

const allEven = numbers.every(num => num % 2 === 0);

console.log(allEven); // true

**13. concat()** Merges two or more arrays into one.

const arr1 = [1, 2];

const arr2 = [3, 4];

const combined = arr1.concat(arr2);

console.log(combined); // [1, 2, 3, 4]

**14. slice()** Returns a shallow copy of a portion of an array.

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

const sliced = numbers.slice(1, 4);

console.log(sliced); // [2, 3, 4]

**15. splice()** Adds or removes elements at a specific index.

const fruits = ["Apple", "Banana", "Orange"];

fruits.splice(1, 1, "Grape");

console.log(fruits); // ["Apple", "Grape", "Orange"]

**16. sort() -** Sorts elements of an array.

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

numbers.sort((a, b) => a - b);

console.log(numbers); // [1, 2, 4, 5]

The sorting function sort ((a, b) => a - b) is commonly used in JavaScript to sort an array of numbers in ascending order. Here's how it works:

### ****How It Works:****

The sort() method takes a comparison function (a, b):

1. If the result of (a - b) is negative (a < b), a comes before b.
2. If the result of (a - b) is zero (a === b), their order remains unchanged.
3. If the result of (a - b) is positive (a > b), b comes before a.

This comparison function ensures the array is sorted in numerical order.

### ****Example: Sorting Numbers in Ascending Order****

const numbers = [5, 3, 8, 1, 2];

numbers.sort((a, b) => a - b);

console.log(numbers); // [1, 2, 3, 5, 8]

#### ****Steps in Sorting:****

1. Compare two elements (a and b).
2. Subtract b from a:
   * If a - b < 0, place a before b.
   * If a - b > 0, place b before a.

For example, with [5, 3, 8, 1, 2]:

* Compare 5 and 3: 5 - 3 = 2 → 3 comes before 5.
* Compare 5 and 8: 5 - 8 = -3 → 5 stays before 8.
* Continue this process until the array is sorted.

### ****Sorting Numbers in Descending Order****

To sort in descending order, reverse the subtraction:

const numbers = [5, 3, 8, 1, 2];

numbers.sort((a, b) => b - a);

console.log(numbers); // [8, 5, 3, 2, 1]

Using sort((a, b) => a - b) ensures numerical sorting.

**17. reverse() -**Reverses the order of elements.

const numbers = [1, 2, 3];

numbers.reverse();

console.log(numbers); // [3, 2, 1]

**18. join()-** Joins all elements of an array into a string.

const words = ["Hello", "world"];

const sentence = words.join(" ");

console.log(sentence); // "Hello world"

**19. indexOf()-**Returns the first index of a given element.

const fruits = ["Apple", "Banana", "Orange"];

const index = fruits.indexOf("Banana");

console.log(index); // 1

**20. includes() -**Checks if an array contains a specific value.

const fruits = ["Apple", "Banana", "Orange"];

console.log(fruits.includes("Banana")); // true

console.log(fruits.includes("Grape")); // false

**21. flat() -**Flattens a nested array into a single array.

const nested = [1, [2, 3], [4, [5]]];

const flattened = nested.flat(2);

console.log(flattened); // [1, 2, 3, 4, 5]

**22. flatMap() -** Maps and flattens the result into a new array.

const numbers = [1, 2, 3];

const result = numbers.flatMap(num => [num, num \* 2]);

console.log(result); // [1, 2, 2, 4, 3, 6]

**23. reduceRight() -**Reduces an array from right to left.

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

const result = numbers.reduceRight((acc, num) => acc + num, 0);

console.log(result); // 10

**24. fill()-**Fills an array with a static value.

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

numbers.fill(0, 1, 3);

console.log(numbers); // [1, 0, 0, 4]

**25. copyWithin()-** Copies a sequence of array elements within the array.

### ****Example: Copy Colors in an Array****

let colors = ["red", "blue", "green", "yellow", "purple"];

colors.copyWithin(1, 3);

console.log(colors);

### ****Step-by-Step Explanation:****

**Before:** ["red", "blue", "green", "yellow", "purple"]  
**copyWithin(1, 3)** → Copy elements **starting from index 3** ("yellow", "purple")   
**Paste them starting at index 1**

🔹 **After:** ["red", "yellow", "purple", "yellow", "purple"]

### ****Another Example: Copy Numbers in an Array****

let numbers = [10, 20, 30, 40, 50];

numbers.copyWithin(2, 0, 2);

console.log(numbers);

### ****Explanation:****

**Before:** [10, 20, 30, 40, 50]  
**copyWithin(2, 0, 2)** → Copy elements **from index 0 to 2** (10, 20) //10, 20  
**Paste them at index 2**

🔹 **After:** [10, 20, 10, 20, 50]