# JavaScript Array Methods with Examples

## forEach()

The `forEach()` method is used to execute a function on each element in an array.

### Example:

const numbers = [1, 2, 3, 4];  
numbers.forEach((num) => {  
 console.log(num ); // Output: 1,2,3,4  
});

### Explanation:

`forEach()` goes through each item in `numbers` and then prints it.

## map()

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

### Example:

const numbers = [1, 2, 3, 4];  
const doubled = numbers.map((num) => num \* 2);  
console.log(doubled); // Output: [2, 4, 6, 8]

### Explanation:

`map()` takes each element, multiplies it by 2, and stores the results in a new array called `doubled`.

## filter()

The `filter()` method creates a new array with only the elements that pass a condition (true/false).

### Example:

const numbers = [1, 2, 3, 4];  
const evenNumbers = numbers.filter((num) => num % 2 === 0);  
console.log(evenNumbers); // Output: [2, 4]

### Explanation:

`filter()` checks each element, and only includes the ones that are even (divisible by 2).

## reduce()

The `reduce()` method reduces the array to a single value by applying a function to each element, accumulating the result.

### Example:

const numbers = [1, 2, 3, 4];  
const sum = numbers.reduce((accumulator, currentValue) => accumulator + currentValue, 0);  
console.log(sum); // Output: 10

### Explanation:

`reduce()` adds each element to an accumulator, starting at `0`, and the result is `10`.

## pop()

The `pop()` method removes the last element from an array and returns it.

### Example:

const numbers = [1, 2, 3, 4];  
const lastElement = numbers.pop();  
console.log(lastElement); // Output: 4  
console.log(numbers); // Output: [1, 2, 3]

### Explanation:

`pop()` removes the last element (`4`) from the `numbers` array.

## push()

The `push()` method adds one or more elements to the end of an array.

### Example:

const numbers = [1, 2, 3];  
numbers.push(4);  
console.log(numbers); // Output: [1, 2, 3, 4]

### Explanation:

`push()` adds `4` to the end of the `numbers` array.

## concat()

The `concat()` method merges two or more arrays into a new array.

### Example:

const array1 = [1, 2];  
const array2 = [3, 4];  
const combined = array1.concat(array2);  
console.log(combined); // Output: [1, 2, 3, 4]

### Explanation:

`concat()` combines `array1` and `array2` into a new array called `combined`.

## slice()

The `slice()` method returns a shallow copy of a portion of an array, without modifying the original array.

### Example:

const numbers = [1, 2, 3, 4, 5];  
const sliced = numbers.slice(1, 4);  
console.log(sliced); // Output: [2, 3, 4]  
console.log(numbers); // Output: [1, 2, 3, 4, 5]

### Explanation:

`slice(1, 4)` returns elements from index `1` to `3` (end index is non-inclusive).