<!-- ! Array Methods -->

JavaScript provides a wide range of built-in methods to manipulate and work with arrays. Here are some commonly used array methods:

1. `push()`

- **Purpose:** Adds one or more elements to the end of the array.
- **Returns:** The new length of the array.

```
let numbers = [1, 2, 3];
numbers.push(4);  // Adds 4 to the end of the array
console.log(numbers);  // Output: [1, 2, 3, 4]
```

2. `pop()`

- **Purpose:** Removes the last element from the array.
- **Returns:** The removed element.

```
let fruits = ["Apple", "Banana", "Cherry"];
let lastFruit = fruits.pop();  // Removes the last element "Cherry"
console.log(fruits);  // Output: ["Apple", "Banana"]
console.log(lastFruit);  // Output: Cherry
```

3. `shift()`

- **Purpose:** Removes the first element from the array.
- **Returns:** The removed element.

```
let fruits = ["Apple", "Banana", "Cherry"];
let firstFruit = fruits.shift();  // Removes "Apple"
console.log(fruits);  // Output: ["Banana", "Cherry"]
console.log(firstFruit);  // Output: Apple
```

4. `unshift()`

- **Purpose:** Adds one or more elements to the beginning of the array.
- **Returns:** The new length of the array.

```
let fruits = ["Banana", "Cherry"];
fruits.unshift("Apple");  // Adds "Apple" to the beginning
console.log(fruits);  // Output: ["Apple", "Banana", "Cherry"]
```

5. `indexOf()`

Definition:
The `indexOf()` method returns the **first index** at which a specified element is found in the array. If the element is not found, it returns `-1`.

Syntax:

```
array.indexOf(searchElement, fromIndex);
```

Example:

```
let fruits = ["Apple", "Banana", "Cherry", "Banana"];
let index = fruits.indexOf("Banana");  // Finds the first occurrence
console.log(index);  // Output: 1
```

6. `lastIndexOf()`

Definition:
The `lastIndexOf()` method returns the **last index** at which a specified element is found in the array. If the element is not found, it returns `-1`.

Syntax:

```
array.lastIndexOf(searchElement, fromIndex);
```

Example:

```
let fruits = ["Apple", "Banana", "Cherry", "Banana"];
let lastIndex = fruits.lastIndexOf("Banana");  // Finds the last occurrence
console.log(lastIndex);  // Output: 3
```

7. `concat()`

- **Purpose:** Merges two or more arrays into a new array.
- **Returns:** A new array.

```
let arr1 = [1, 2];
let arr2 = [3, 4];
let mergedArray = arr1.concat(arr2);
console.log(mergedArray);  // Output: [1, 2, 3, 4]
```

8. `includes()`

- **Purpose:** Determines whether an array contains a certain value.
- **Returns:** `true` if the array contains the value, otherwise `false`.

```
let fruits = ["Apple", "Banana", "Cherry"];
console.log(fruits.includes("Banana"));  // Output: true
```

10. `reverse()`

Definition:
The `reverse()` method reverses the order of the elements in an array in place. The first array element becomes the last, and the last becomes the first.

Syntax:

```
array.reverse();
```

- The `reverse()` method directly modifies the original array.
- It does not create a new array, but rather changes the order of elements in the array itself.

Example:

```
let numbers = [1, 2, 3, 4, 5];
numbers.reverse();  // Reverses the order of the array
console.log(numbers);  // Output: [5, 4, 3, 2, 1]
```

Key Points:
- **In-Place Modification:** The original array is modified.

11. `join()`

Definition:

The `join()` method joins all elements of an array into a string, with a specified separator between the elements. If no separator is provided, a comma (`,`) is used by default.

Syntax:

```
array.join(separator);
```

- `separator` (optional): Specifies the string to separate each element. If omitted, a comma is used.

Example:

```
let fruits = ["Apple", "Banana", "Cherry"];
let joinedFruits = fruits.join(" - ");
console.log(joinedFruits);  // Output: "Apple - Banana - Cherry"
```

12. `slice()`

- **Purpose:** Extracts a section of an array and returns a new array.
- **Returns:** A new array containing the extracted elements.

```
let fruits = ["Apple", "Banana", "Cherry", "Date"];
let slicedFruits = fruits.slice(1, 3);  // Extracts elements at index 1 and 2
console.log(slicedFruits);  // Output: ["Banana", "Cherry"]
```

13. `splice()`

- **Purpose:** Adds or removes elements from the array.
- **Returns:** An array containing the deleted elements.

```
let fruits = ["Apple", "Banana", "Cherry"];
fruits.splice(1, 1, "Mango");  // Replaces "Banana" with "Mango"
console.log(fruits);  // Output: ["Apple", "Mango", "Cherry"]
```

14. `Array.isArray()`

Definition:

The `Array.isArray()` method determines whether the passed value is an array.

Syntax:

```
Array.isArray(value);
```

- **value:** The value you want to check.

Returns: `true` if the value is an array; otherwise, `false`.

Example:

```
console.log(Array.isArray([1, 2, 3]));  // Output: true
console.log(Array.isArray("Hello"));    // Output: false
console.log(Array.isArray({ key: "value" }));  // Output: false
```

15. `flat()`

Definition:

The `flat()` method creates a new array with all sub-array elements concatenated into it recursively up to the specified depth.

Syntax:

```
array.flat(depth);
```

- **depth** (optional): Specifies how deep to flatten the array. Default is `1`.

Example:

```
let nestedArray = [1, [2, [3, [4]]]];
console.log(nestedArray.flat(2));  // Output: [1, 2, 3, [4]]
```

```
// !  how to declare array


let arr1 = [10,'san',true,[20,40],()=>{}]

console.log(arr1[4])

// ! how to know the length of array

let len = arr1.length
console.log(len)


// !   methods of array


let arr2 = [20,40,60,33,60,12]

// ! 1. push()

// let lenForPush = arr2.push(99)
// console.log(lenForPush)
// console.log(arr2)


// ! 2. pop()

// let lastEle = arr2.pop()
// console.log(lastEle)
// console.log(arr2)


// !  3. unshift()


// let lenForUnshift = arr2.unshift('hi')
// console.log(lenForUnshift)
// console.log(arr2)
```

```
// ! 4. shift()

// let removedEleFromStart = arr2.shift()
// console.log(removedEleFromStart)
// console.log(arr2)


// ! 5.  includes()

console.log(arr2.includes(60))

// ! 6. indexOf()

console.log(arr2.indexOf(60))

// ! 7. lastIndexOf()

console.log(arr2.lastIndexOf(60))

// ! 8. concat()

let arr3 = [3,4,5,6]
let arr4 = [8,9,6,5]

let newArr = arr3.concat(arr4)
console.log(newArr)

// ! 9. flat()

let arr5 = [1,3,4,[67,90,90,[3,4,5,['hi','hello']]]]

console.log(arr5)

let flatArr = arr5.flat(Infinity)
console.log(flatArr)


// ! 10. reverse()

let arr6 = [ 4,6,8,9,23]

arr6.reverse()
console.log(arr6)
```

```
// ! 11. join()

let str = arr6.join("")
console.log(str)

// ! 12. slice()

let arr7 = [3,6,7,8,9,20]

let slicedArr = arr7.slice(1,4)

console.log(slicedArr)   // 6,7,8

// ! 13. splice()

// arr7.splice(1,3,'hi')
// console.log(arr7)


arr7.splice(1,0,'hi')   //  it will add hi at the index 1
console.log(arr7)

// ! 14  Array.isArray()

console.log(Array.isArray(arr7))
console.log(Array.isArray("hi"))
```