JavaScript Array Methods - 5 Examples Each ==========================================

1. Array.every()

-----------------

Example 1: Check if all numbers are positive

[1, 2, 3].every(x => x > 0); // true

Example 2: Check if all are strings ["a", "b", "c"].every(x => typeof x === "string"); // true

Example 3: Validate age > 18

[{age: 20}, {age: 25}].every(p => p.age > 18); // true

Example 4: Check if array is not empty and all elements are truthy

[1, "a", true].every(Boolean); // true

Example 5: Check if all numbers are even [2, 4, 6].every(x => x % 2 === 0); // true

2. Array.some()

----------------

Example 1: Check if any number is even

[1, 3, 4].some(x => x % 2 === 0); // true

Example 2: Check if array has null

[null, "a"].some(x => x === null); // true

Example 3: At least one user is admin [{admin: false}, {admin: true}].some(u => u.admin); // true

Example 4: Check if any string includes "a"

["cat", "dog"].some(str => str.includes("a")); // true

Example 5: Some elements are greater than 10 [5, 8, 15].some(x => x > 10); // true

3. Array.filter()

------------------

Example 1: Filter even numbers [1, 2, 3, 4].filter(x => x % 2 === 0); // [2, 4]

Example 2: Get users over 18 [{age: 17}, {age: 22}].filter(u => u.age > 18); // [{age: 22}]

Example 3: Filter out falsy values

[0, "", "hello", null].filter(Boolean); // ["hello"]

Example 4: Words longer than 3 letters ["hi", "hello", "yes"].filter(w => w.length > 3); // ["hello"]

Example 5: Items with price > 10 [{price: 5}, {price: 15}].filter(item => item.price > 10);

4. Array.reduceRight()

------------------------

Example 1: Subtract from right to left

[10, 1, 1].reduceRight((a, b) => a - b); // 0

Example 2: Concatenate from right ["a", "b", "c"].reduceRight((a, b) => a + b); // "cba"

Example 3: Reduce with initial value

[1, 2].reduceRight((a, b) => a + b, 5); // 8

Example 4: Create reversed string ["1", "2", "3"].reduceRight((a, b) => a + b); // "321"

Example 5: Nest arrays [[3], [2], [1]].reduceRight((a, b) => [b, a]);

5. Array.reduce()

------------------

Example 1: Sum numbers [1, 2, 3].reduce((a, b) => a + b); // 6

Example 2: Flatten array [[1], [2], [3]].reduce((a, b) => a.concat(b), []); // [1, 2, 3]

Example 3: Count occurrences ['a', 'b', 'a'].reduce((acc, val) => { acc[val] = (acc[val] || 0) + 1; return acc; }, {});

Example 4: Max value [1, 5, 3].reduce((a, b) => a > b ? a : b); // 5

Example 5: Build object ['x', 'y'].reduce((obj, key, i) => ({ ...obj, [key]: i }), {});

6. Array.pop()

---------------Example 1: Remove last item const arr1 = [1, 2, 3]; arr1.pop(); // [1, 2]

Example 2: Pop from empty array

[].pop(); // undefined

Example 3: Chain pop and log const arr2 = [5, 6]; console.log(arr2.pop()); // 6

Example 4: Pop inside loop let arr3 = [1,2,3]; while(arr3.length) console.log(arr3.pop());

Example 5: Pop returns removed element const last = [10, 20].pop(); // 20

7. Array.push()

---------------Example 1: Add one element const arr4 = [1]; arr4.push(2); // [1, 2]

Example 2: Add multiple elements arr4.push(3, 4); // [1, 2, 3, 4]

Example 3: Push returns new length let len = arr4.push(5); // 5

Example 4: Add array as element arr4.push([6, 7]); // [1, 2, 3, 4, 5, [6, 7]]

Example 5: Push in loop let result = []; for (let i = 0; i < 3; i++) result.push(i);

8. Array.shift()

----------------Example 1: Remove first item let q = [1, 2, 3]; q.shift(); // [2, 3] Example 2: Shift returns removed element let first = q.shift(); // 2

Example 3: Shift from empty array

[].shift(); // undefined

Example 4: Chain shift let a = [10, 20, 30]; console.log(a.shift()); // 10

Example 5: Shift in loop let b = [1,2,3]; while(b.length) console.log(b.shift());

9. Array.unshift()

------------------Example 1: Add element at beginning let c = [2, 3]; c.unshift(1); // [1, 2, 3]

Example 2: Add multiple items

c.unshift(-1, 0); // [-1, 0, 1, 2, 3]

Example 3: Unshift returns new length let newLen = c.unshift(-2); // 6

Example 4: Add array as first element

c.unshift(["x", "y"]); // [["x", "y"], -2, -1, 0, 1, 2, 3]

Example 5: Use in loop (reverse push) let d = []; for(let i=3; i>0; i--) d.unshift(i); // [1, 2, 3]  
+++++++++++++++++++++++++++++++++++++++++++++++++++++++

// 1. Array.find() - Returns first element that satisfies the condition

const numbers1 = [3, 7, 10, 15];

console.log(numbers1.find(n => n > 5)); // 7

console.log(numbers1.find(n => n > 20)); // undefined

console.log(["cat", "dog", "bat"].find(a => a.includes("o"))); // dog

console.log([false, false, true].find(Boolean)); // true

console.log([1, 2, 3].find(x => x % 2 === 0)); // 2

// 2. Array.findIndex() - Returns index of first element that satisfies condition

console.log(numbers1.findIndex(n => n > 5)); // 1

console.log(numbers1.findIndex(n => n > 20)); // -1

console.log(["a", "b", "c"].findIndex(c => c === "b")); // 1

console.log([0, 0, 1].findIndex(Boolean)); // 2

console.log(["apple", "banana"].findIndex(f => f.startsWith("b"))); // 1

// 3. Array.findLast() - Returns last element that satisfies condition

console.log(numbers1.findLast(n => n < 15)); // 10

console.log(numbers1.findLast(n => n > 15)); // undefined

console.log([1, 2, 3, 4, 5].findLast(n => n % 2 === 0)); // 4

console.log(["one", "two", "three"].findLast(w => w.includes("e"))); // three

console.log([true, false, true].findLast(v => v === true)); // true

// 4. Array.findLastIndex() - Returns index of last match

console.log(numbers1.findLastIndex(n => n < 15)); // 2

console.log(numbers1.findLastIndex(n => n > 20)); // -1

console.log([1, 2, 3, 4, 5].findLastIndex(n => n % 2 === 0)); // 3

console.log(["x", "y", "z"].findLastIndex(c => c === "y")); // 1

console.log(["apple", "banana"].findLastIndex(f => f.includes("a"))); // 1

// 5. Array.keys() - Returns an iterator of indexes

const fruits = ["apple", "banana", "cherry"];

for (let key of fruits.keys()) console.log(key); // 0, 1, 2

console.log([...fruits.keys()]); // [0, 1, 2]

console.log(Object.keys(fruits)); // ["0", "1", "2"]

console.log(fruits.keys().next().value); // 0

console.log(typeof fruits.keys()); // object

// 6. Array.values() - Returns an iterator of values

for (let value of fruits.values()) console.log(value); // apple, banana, cherry

console.log([...fruits.values()]); // ["apple", "banana", "cherry"]

console.log(fruits.values().next().value); // apple

console.log(typeof fruits.values()); // object

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

// 7. Array.entries() - Returns an iterator of [index, value] pairs

for (let [index, value] of fruits.entries()) {

console.log(index, value);

}

// 0 apple

// 1 banana

// 2 cherry

console.log([...fruits.entries()]); // [[0, 'apple'], [1, 'banana'], [2, 'cherry']]

console.log(fruits.entries().next().value); // [0, "apple"]

console.log(typeof fruits.entries()); // object

console.log(Array.from(fruits.entries())); // same as above

// 8. Array.toString() - Converts to comma-separated string

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

console.log(["a", "b", "c"].toString()); // "a,b,c"

console.log([true, false].toString()); // "true,false"

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

console.log([null, undefined].toString()); // ","

// 9. Array.includes() - Checks for presence of element

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

console.log(["a", "b", "c"].includes("d")); // false

console.log([NaN].includes(NaN)); // true

console.log(["one", "two"].includes("one")); // true

console.log(["x", "y", "z"].includes("x", 1)); // false