```javascript
// ✅ JavaScript Arrays – Mastering the Fundamentals

// An array is a data structure that allows you to store multiple values in a single variable.

// It can hold various data types, including numbers, strings, objects and even other arrays.

// Arrays are zero-indexed, meaning that the first element in an array is at index 0, the second element with index 1 and so on.

// 📘 We will cover the following topics:

/*

1. Creating Arrays, Accessing Elements and Modifying Elements

2. Array Traversal, Iterations

3. Updating and Deleting Elements

4. Filter and Search

5. Sort and Compare

6. Important Array Methods

...and a lot more things.

*/

console.clear();

// 🔷 Example: Creating an Array

// ✅ Using Array Literal

let arr = [1, 2, 3, 4, 5];

console.log(typeof arr); // object

console.log(arr);

// ✅ Using Array Constructor

let fruitsArr = new Array("apple", "banana", "orange", "grapes", "pineapple");

console.log(fruitsArr);

// 🔷 Accessing Elements in an Array

console.log(fruitsArr[0]); // apple

console.log(fruitsArr[1]); // banana

console.log(fruitsArr[2]); // orange

// 🔷 Modifying Elements in an Array

fruitsArr[0] = "mango";

console.log(fruitsArr);

// 🔷 Array Traversal

// ✅ 1. for...of loop → Iterates over values

console.log("Using for...of loop");

for (const fruitItem of fruitsArr) {

    console.log(fruitItem);
}

// Output: mango, banana, orange, grapes, pineapple

// ✅ 2. for loop → Index-based iteration

```

```javascript
console.log("Using for loop");

for (let item = 0; item < fruitsArr.length; item++) {

    console.log(fruitsArr[item]);
}

// ✅ 3. for...in loop → Iterates over keys (indexes)

console.log("Using for...in loop");

for (const key in fruitsArr) {

    console.log(key);
}

// ✅ 4. forEach() → Executes callback for each element. It doesn't return anything.

/*

1. array: The array on which the forEach() method is called.

2. callback: A function that is called once for each element in the array.

3. currentValue: The current element being processed in the array.

4. index: The index of the current element being processed in the array.

*/

fruitsArr.forEach((fruitItem, index, arr) => {

    console.log(index, fruitItem, arr);
});

// ✅ 5. map() → Creates a new array with the results of the callback

const newArr = fruitsArr.map((fruitItem, index) => {

    return fruitItem + " " + index;
});

console.log(newArr);

// 🔷 Array Update Methods

/*

✅ 1. push(): Adds one or more elements to the end of an array.

   ➤ Modifies the original array. Returns the new length.

✅ 2. pop(): Removes the last element from an array.

   ➤ Returns the removed element.

✅ 3. unshift(): Adds one or more elements to the beginning of an array.

   ➤ Modifies the original array. Returns the new length.

✅ 4. shift(): Removes the first element from an array.

   ➤ Returns the removed element.

✅ 5. splice(): Removes elements from an array and optionally inserts new ones.

   ➤ Returns the removed elements.

✅ 6. split(): Splits a string into an array of substrings based on a separator.

   ➤ Returns the new array of substrings.

✅ 7. slice(): Returns a shallow copy of a portion of an array.

   ➤ Returns the new array.

```

```
✅ 8. concat(): Concatenates two or more arrays.

   ➤ Returns the new concatenated array.

✅ 9. join(): Joins all elements of an array into a string.

   ➤ Returns the joined string.

*/

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

console.log(numbersArray);

numbersArray.push(6);       // [1, 2, 3, 4, 5, 6]

console.log(numbersArray);

numbersArray.pop();         // [1, 2, 3, 4, 5]

console.log(numbersArray);

numbersArray.unshift(0);    // [0, 1, 2, 3, 4, 5]

console.log(numbersArray);

numbersArray.shift();       // [1, 2, 3, 4, 5]

console.log(numbersArray);

numbersArray.splice(3, 1, 7); // [1, 2, 3, 7, 5]

console.log(numbersArray);

const stringArray = "Hello World";

console.log(stringArray.split("")); // ['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd']

console.log(stringArray.slice(0, 5)); // Hello

const arr1 = [1, 2, 3];

const arr2 = [4, 5, 6];

const arr3 = arr1.concat(arr2);

console.log(arr3); // [1, 2, 3, 4, 5, 6]

console.log(arr3.join("")); // 123456

// 🔷 Searching in an Array

/*

✅ 1. indexOf(): Returns the index of the first occurrence of a specified value.

    ➤ Returns -1 if the value is not found.

✅ 2. lastIndexOf(): Returns the index of the last occurrence of a specified value.

    ➤ Returns -1 if the value is not found.

✅ 3. includes(): Returns true if an array contains a specified value.

    ➤ Returns false if the value is not found.

*/

const searchArray = [1, 2, 3, 6, 4, 5, 6, 7, 8, 9];

console.log(searchArray);

console.log("The index of 4 is", searchArray.indexOf(4));           // 4

console.log("The last index of 6 is", searchArray.lastIndexOf(6));  // 6

console.log("Array contains 2 ?", searchArray.includes(2));         // true

// 🔷 Filter Methods

/*

✅ 1. find(): Returns the value of the first element that satisfies the condition.

✅ 2. findIndex(): Returns the index of the first element that satisfies the condition.

✅ 3. filter(): Returns a new array with all elements that pass the condition.

✅ 4. sort(): Sorts the array in place and returns it. Accepts a comparator function.

✅ 5. every(): Returns true if all elements in the array pass the condition.

✅ 6. some(): Returns true if at least one element in the array passes the condition.

✅ 7. reduce(): Applies a function against an accumulator and each value of the array (from left to right) to reduce it to a single value.

*/

console.log("Filtering an Array");

const filterArray = [1, 2, 3, 4, 5, 6, 7, 8, 6, 9];

console.log(filterArray);

// find → First element > 5

const findElement = filterArray.find((currentElement) => {

    return currentElement > 5;
});

console.log(findElement); // 6

```

```
// findIndex → Index of first element > 5

const findIndexElement = filterArray.findIndex((currentElement) => {

    return currentElement > 5;
});

console.log(findIndexElement); // 5

// filter → All elements except value 6

let value = 6;

const newArray = filterArray.filter((currentElement) => {

    return currentElement !== value;
});

console.log(newArray);

// every → All elements > 0

const everyElement = filterArray.every((currentElement) => {

    return currentElement > 0;
});

console.log(everyElement); // true

// some → At least one element > 4

const someElement = filterArray.some((currentElement) => {

    return currentElement > 4;
});

console.log(someElement); // true

```

```javascript
// reduce → Sum of all elements - Initial value of accumulator is 0

const reduceElement = filterArray.reduce((accumulator, currentElement) => {

    return accumulator + currentElement;

}, 0);

// ✅ Sort Method → Sorts the elements in ascending order or descending order using comparator logic

console.log("Sorting an Array in Ascending Order:");

const jumbledArray = [2, 5, 1, 4, 3, 9, 6, 8, 7];

jumbledArray.sort((a, b) => {

    return a - b;
});

console.log(jumbledArray);

console.log("Sorting an Array in Descending Order:");

jumbledArray.sort((a, b) => {

    return b - a;
});

console.log(jumbledArray);
```