**Day 10**



**Using Loops with Arrays in JavaScript:**

**Why Use Loops with Arrays?**

When you have multiple elements in an array, you often need to:

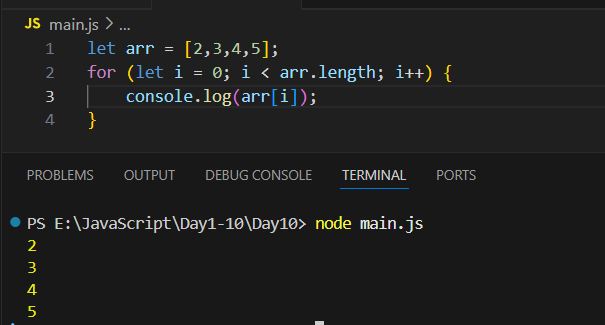
* Access each element one by one
* Perform an operation on each element
* Search, modify, or calculate something

Loops help you iterate (go through) all array elements efficiently.

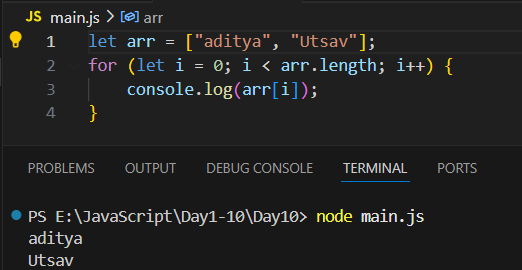
**Common Loop Types Used with Arrays**

| **Loop Type** | **Description** | **Common Use** |
| --- | --- | --- |
| **for** | Runs a block of code a set number of times | When you know how many elements are there |
| **while** | Runs while a condition is true | When number of iterations is uncertain |
| **do...while** | Similar to while, but executes once before checking condition | When you want to run at least once |
| **for...of** | Iterates directly over array values | Simplified syntax for reading values |
| **for...in** | Iterates over array *keys (indexes)* | Rarely used; better for objects |
| **forEach()** | Built-in array method to run a function for each element | Cleaner, functional approach |

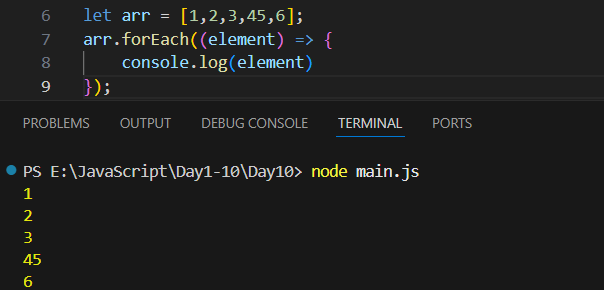
Example: to print the elements of the array using the for…loop



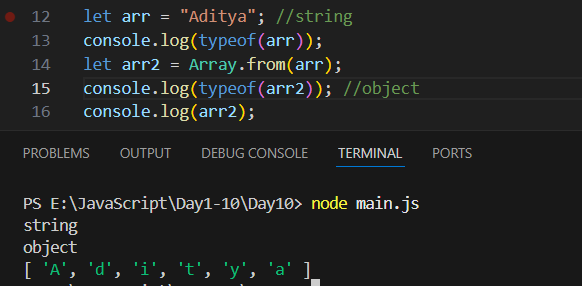
Example: to print the element where the elements of the array be the strings.



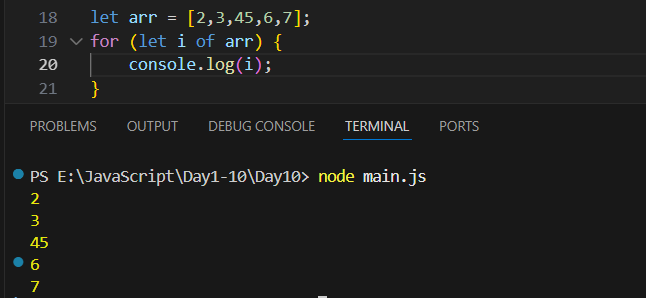
Example: we can also use forEach loop to print the elements of the array



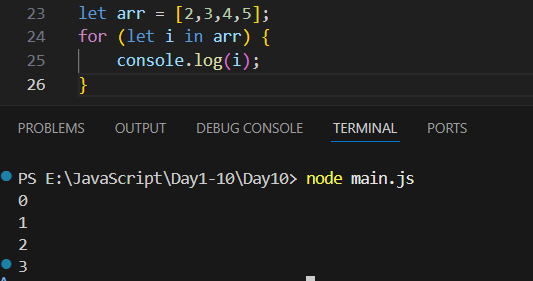
Example: Arrya.from is used to create an array from any other object.



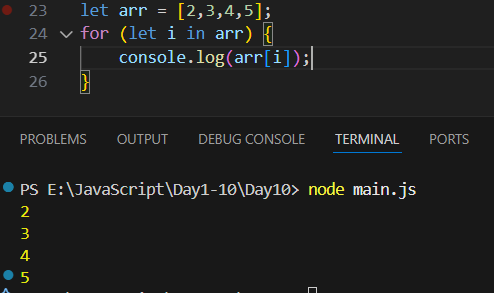
Example: for..of loop and array



Example: for … in loop and array



Example: for..in loop to print the elements of array.



**Map, Filter & Reduce in JavaScript :**

**map()**

**Concept:**

* Creates a new array by applying a function to each element of the original array.
* Original array remains unchanged.

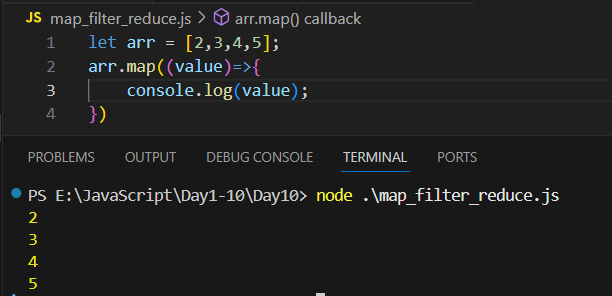
**Syntax:**

*array.map(function(element, index, array) {*

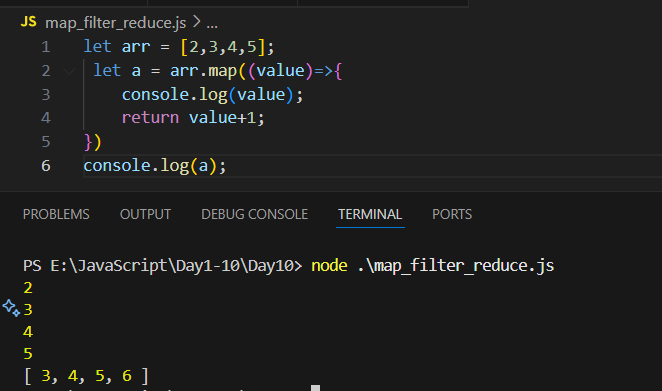
*// return new value for new array*

*})*

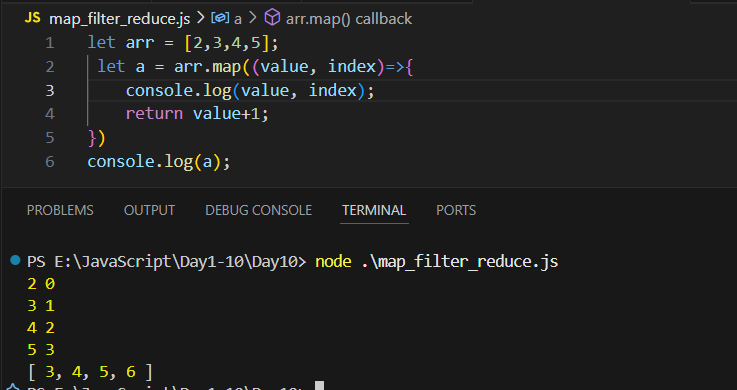
Example:



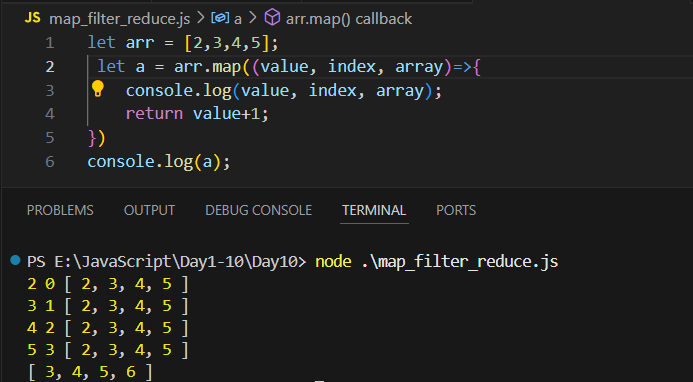
Example:



Example:



Example:



**filter()**

**Concept:**

* Creates a new array with only the elements that pass a condition.
* Original array remains unchanged.

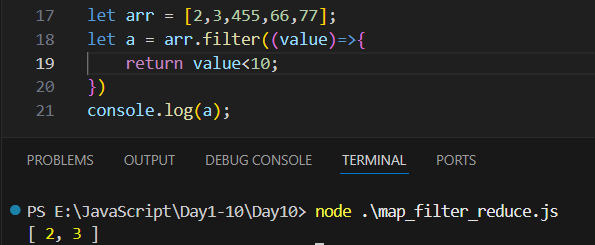
**Syntax:**

*array.filter(function(element, index, array) {*

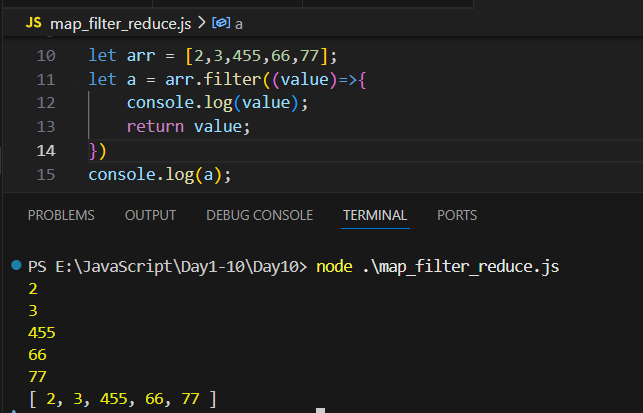
*// return true to keep element, false to discard*

*})*

Example: to filter out those values of array whose value is less than 10.



Example:



**reduce()**

**Concept:**

* Reduces an array to a single value (sum, product, object, etc.).
* Applies a function to each element, accumulating a result.

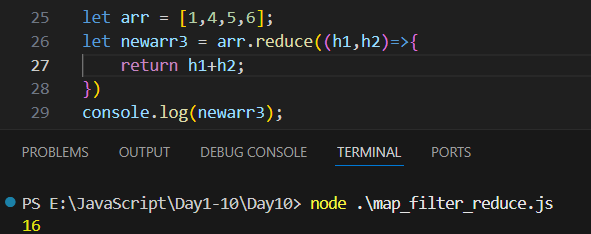
**Syntax:**

*array.reduce(function(accumulator, currentValue, index, array) {*

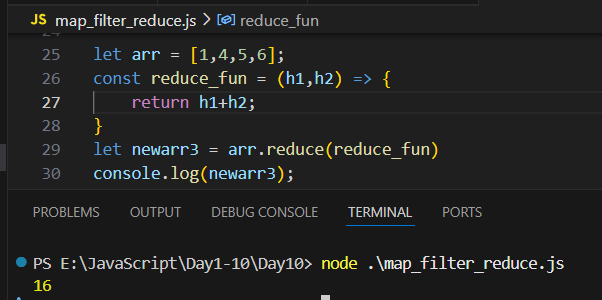
*// return updated accumulator*

*}, initialValue)*

Example:



Example:



**Key Differences**

| **Method** | **Returns** | **Original Array Modified?** | **Usage** |
| --- | --- | --- | --- |
| map() | New array | No | Transform each element |
| filter() | New array | No | Select elements by condition |
| reduce() | Single value | No | Combine elements to one result |

--The End--