**ForEach:**

The forEach method in JavaScript is used to iterate over elements of an array and execute a provided callback function once for each array element. It doesn't return anything (i.e., it returns undefined), but it's mainly used for its side effects (like updating elements or logging). The callback function accepts three arguments: the current element being processed, the index of that element, and the array that forEach is being called upon.

Syntax:

array.forEach(function(currentValue, index, array) {

// Your code here

});

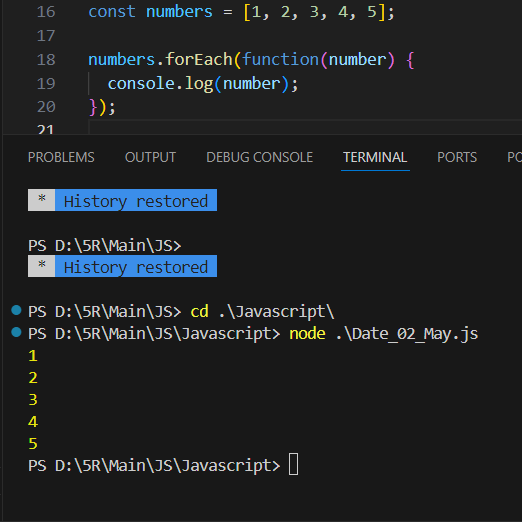
(or)

array.forEach((currentValue, index, array) => {

// Your code here

});

Example:



**Map:**

The map function in JavaScript is used to transform elements of an array according to a provided callback function, creating a new array with the results of calling the callback function on every element in the original array. It's useful when you want to apply a function to each element of an array and collect the results into a new array without mutating the original array.

Syntax:

const newArray = array.map(function(currentValue, index, array) {

// Your code here

// Return the new value

});

or

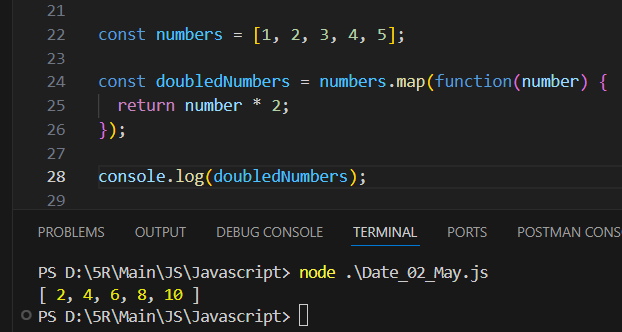
const newArray = array.map((currentValue, index, array) => {

// Your code here

// Return the new value

});

Example:



**filter:**

The filter method in JavaScript is used to create a new array with all elements that pass the test implemented by the provided callback function. It's particularly useful when you want to extract certain elements from an array based on a condition.

Syntax:

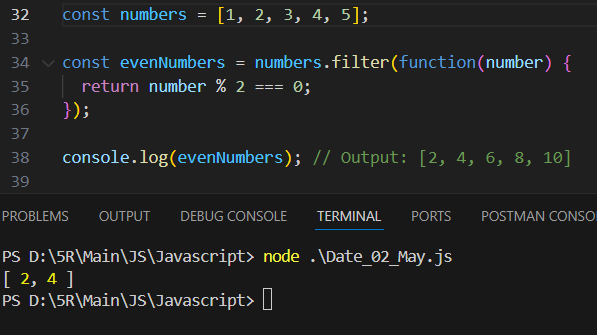
const newArray = array.filter(function(currentValue, index, array) {

// Your code here

// Return true to keep the element, false otherwise

});

Example:



**Reduce**

The reduce method in JavaScript is used to reduce the elements of an array to a single value. It executes a provided callback function once for each element in the array, resulting in a single output value. This method is quite versatile and can be used for various tasks such as summing up values, concatenating strings, or any other aggregation operation.

Syntax:

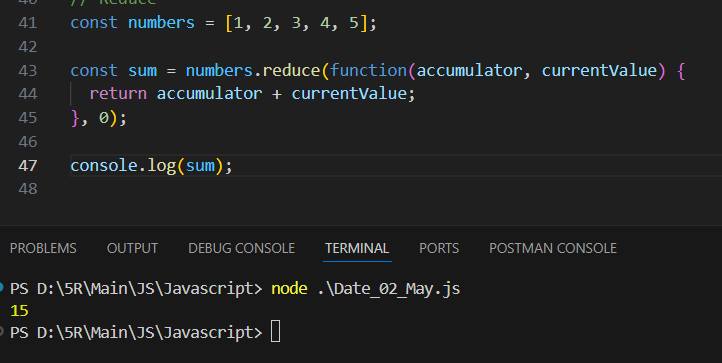
const result = array.reduce(function(accumulator, currentValue, index, array) {

// Your code here

// Return the updated accumulator value

}, initialValue);

Example:



**ReduceRight:**

In JavaScript, the reduceRight method is similar to the reduce method, but it reduces the array from right to left instead of from left to right. It iterates over the elements of the array in reverse order and applies a callback function against an accumulator and each value of the array, resulting in a single output value.

Syntax:

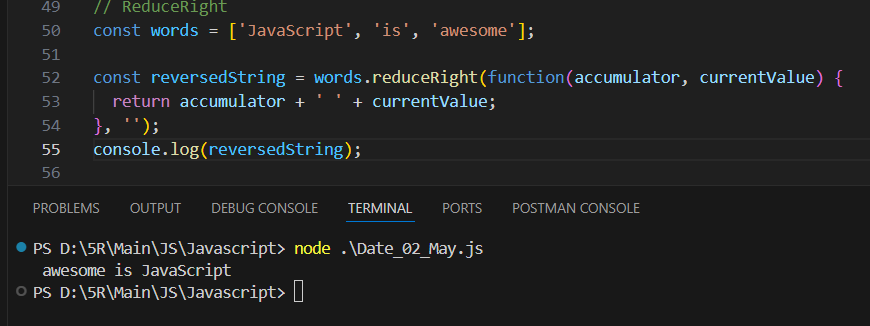
const result = array.reduceRight(function(accumulator, currentValue, index, array) {

// Your code here

// Return the updated accumulator value

}, initialValue);

Example:



**Some**

The some method in JavaScript is used to test whether at least one element in the array passes the test implemented by the provided callback function. It returns true if at least one element in the array satisfies the condition; otherwise, it returns false. This method is useful when you want to check if any element in an array meets a certain condition.

Syntax:

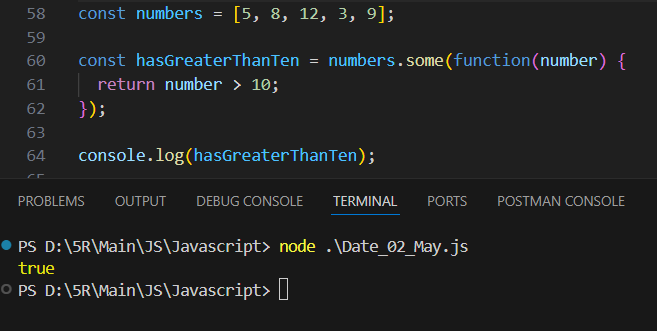
const result = array.some(function(currentValue, index, array) {

// Your code here

// Return true if the condition is met, false otherwise

});

Example:



**Every**

The every method in JavaScript is used to test whether all elements in the array pass the test implemented by the provided callback function. It returns true if every element in the array satisfies the condition; otherwise, it returns false. This method is useful when you want to check if all elements in an array meet a certain condition.

Syntax:

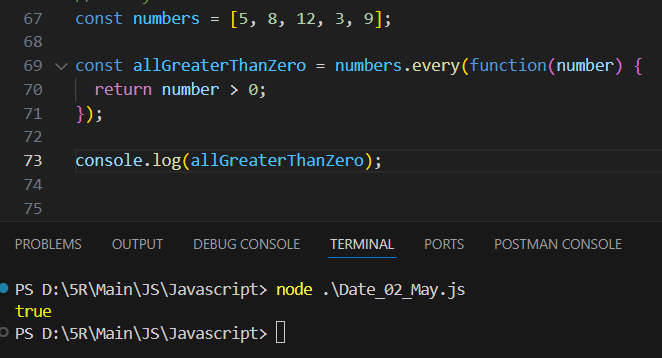
const result = array.every(function(currentValue, index, array) {

// Your code here

// Return true if the condition is met, false otherwise

});

Example:



**Find**

The find method in JavaScript is used to retrieve the first element in an array that satisfies a provided callback function. It returns the value of the first element found in the array that satisfies the provided testing function, or undefined if no elements in the array satisfy the function. This method is handy when you want to find a specific element in an array based on a certain condition.

Syntax:

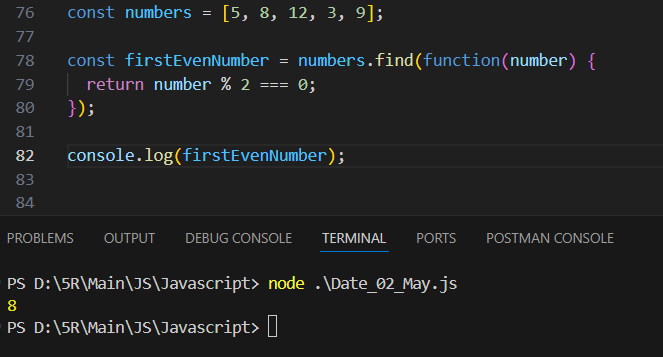
const result = array.find(function(currentValue, index, array) {

// Your code here

// Return true if the condition is met, false otherwise

});

Example:



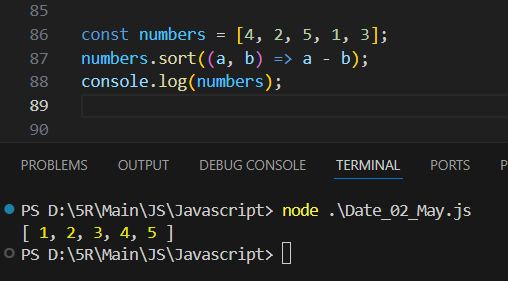
**sort**

In JavaScript, the sort method is used to sort the elements of an array in place and returns the sorted array. By default, the sort method sorts the elements as strings in Unicode code point order. This can lead to unexpected results when sorting numbers.

Syntax:

array.sort([compareFunction]);

Example:



**Reverse**

In JavaScript, the reverse method is used to reverse the order of elements in an array. It modifies the original array in place and returns the reversed array.

Syntax:

array.reverse();

Example:

