ForEach() method:

The **forEach()** method is a built-in method in JavaScript that is used to execute a provided function once for each element in an array. It allows you to iterate over the elements of an array and perform an operation on each element. The **forEach()** method does not mutate the original array.

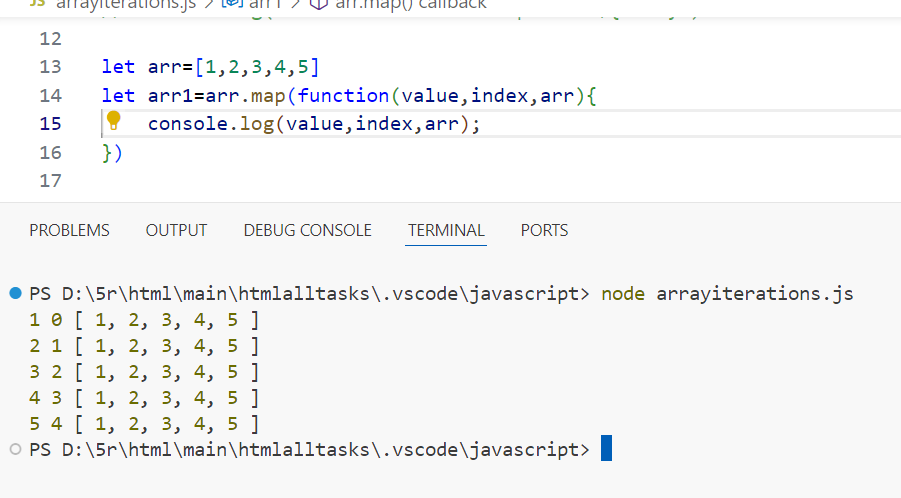
Here is the example:



Map:

The **map()** method is a built-in method in JavaScript used to create a new array populated with the results of calling a provided function on every element in the calling array. It allows you to transform each element of an array based on a function and returns a new array with the transformed values. The original array remains unchanged.

Here is the example:

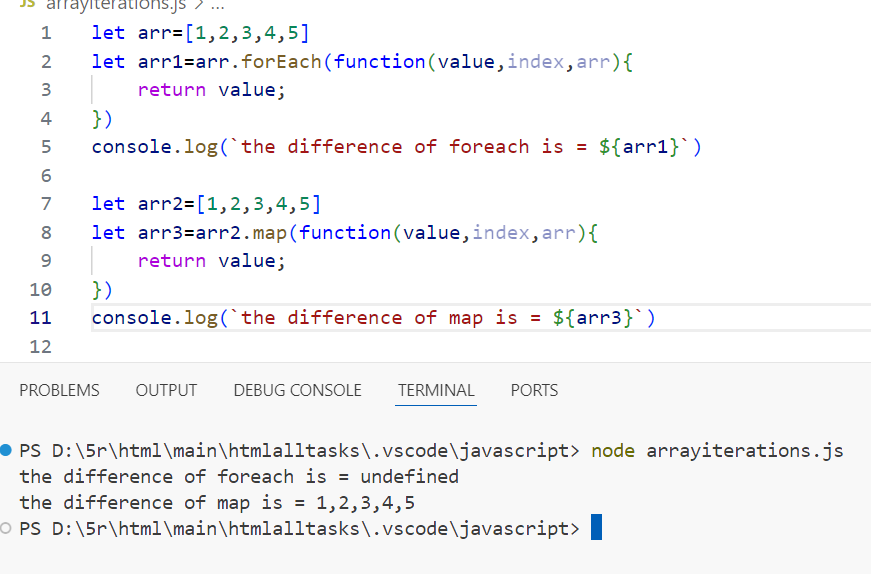


Difference between forEach and map:

The key differences between **forEach()** and **map()** in JavaScript lie in their purposes, return values, and how they handle the array elements:

1. **Purpose**:
   * **forEach()**: It executes a provided function once for each array element. It's mainly used when you want to perform an operation on each element of the array without necessarily modifying the array itself.
   * **map()**: It creates a new array populated with the results of calling a provided function on every element in the calling array. It's used when you want to transform each element of an array and produce a new array with the transformed values.
2. **Return Value**:
   * **forEach()**: It doesn't return anything. It simply iterates over the array and executes the provided function for each element.
   * **map()**: It returns a new array containing the results of applying the provided function to each element in the original array, in order.
3. **Handling Elements**:
   * **forEach()**: It iterates over each element of the array and executes the provided function for each element. It doesn't create a new array or modify the original array.
   * **map()**: It also iterates over each element of the array, but it creates a new array with the results of calling the provided function on each element. It returns this new array without modifying the original array.
4. **Mutability**:
   * **forEach()**: It does not modify the original array.
   * **map()**: It does not modify the original array but produces a new array with transformed values.

Here is the example:



Filter:

The **filter()** method in JavaScript is used to create a new array with all elements that pass a test implemented by the provided function. It does not modify the original array; instead, it returns a new array containing only the elements for which the provided function returns **true**.

* **callback**: The function to test each element of the array. It can take up to three arguments:
  + **element**: The current element being processed in the array.
  + **index** (optional): The index of the current element being processed.
  + **array** (optional): The array **filter()** was called upon.

The **filter()** method iterates over each element in the array, calls the provided callback function once for each element, and creates a new array with all elements that pass the test implemented by the callback function.

Here is the example:



Reduce:

The **reduce()** method in JavaScript is used to apply a function to each element in the array to reduce the array to a single value. It executes the provided reducer function for each element of the array, resulting in a single output value.

* **callback**: The function to execute on each element in the array. It takes four arguments:
  + **accumulator**: The accumulator accumulates the callback's return values. It is the accumulated value previously returned in the last invocation of the callback, or the **initialValue**, if supplied.
  + **currentValue**: The current element being processed in the array.
  + **index** (optional): The index of the current element being processed.
  + **array** (optional): The array **reduce()** was called upon.
* **initialValue**: A value to use as the first argument to the first call of the callback. If no initial value is supplied, the first element in the array will be used as the initial accumulator value, and iteration will start from the second element.

The **reduce()** method executes the provided callback function once for each element present in the array (excluding holes in the array, such as undefined elements), taking four arguments: the accumulator, the current element, the current index, and the array. The callback function's return value is assigned to the accumulator, whose value is remembered across each iteration throughout the array, and ultimately becomes the final, single resulting value.

Here is the example:



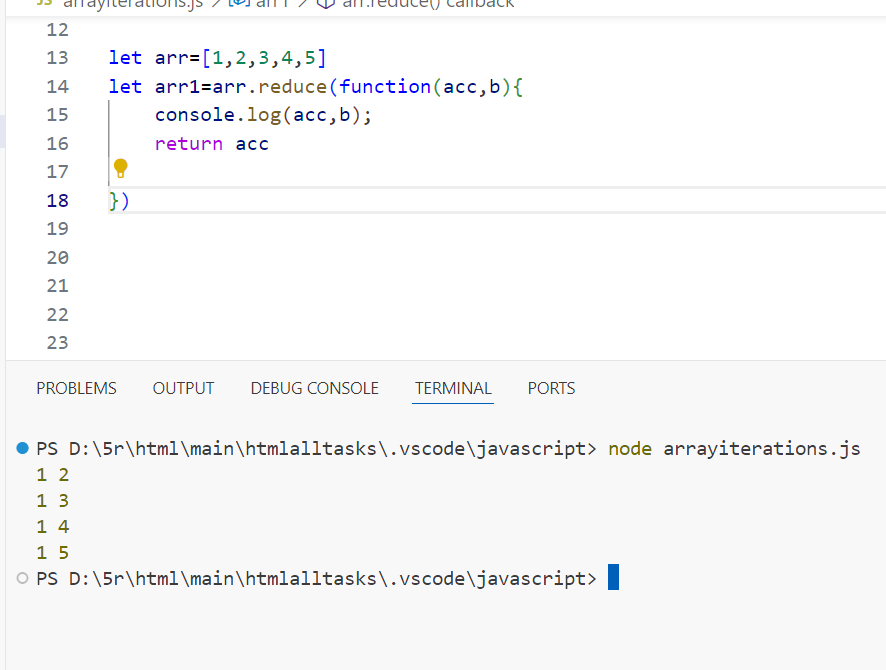
Here the output is undefined because we didn’t give any return function .

When we give return function:



When we not give that accumulator then it takes the input as a accumulator

Here we go



We have also reduceRight:

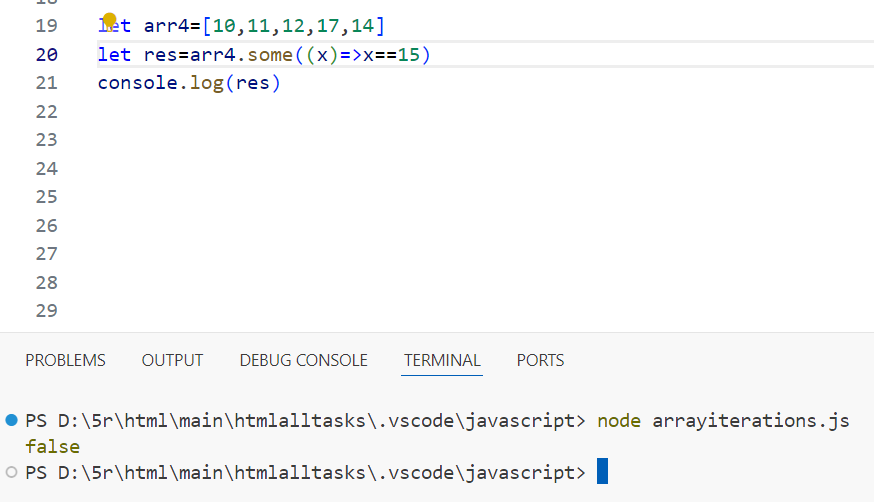


These all methods do in arrow function also

Here is the example:



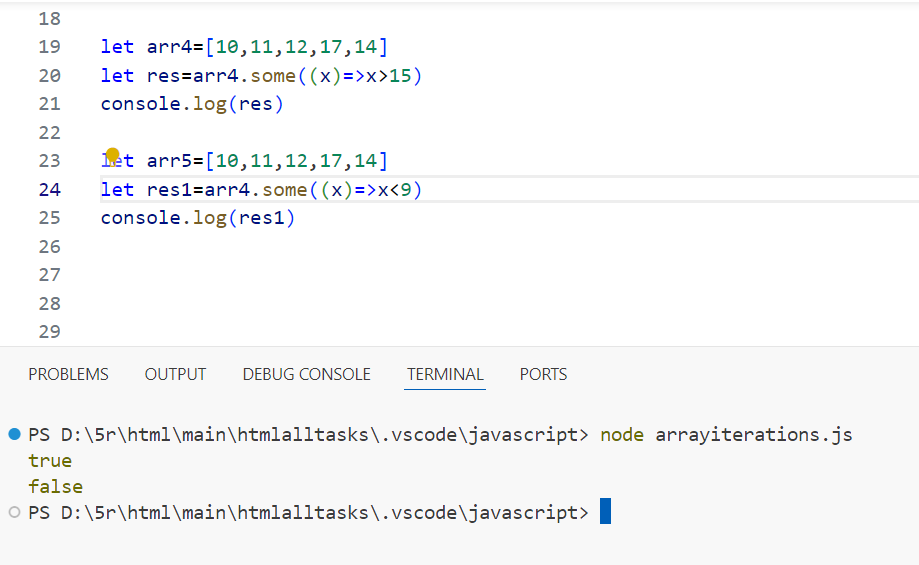
Array some method:



every() method:

The **every()** method in JavaScript is used to test whether all elements in an array pass the test implemented by the provided function. It returns a Boolean value indicating whether all elements in the array satisfy the provided testing function.

Here is the example:

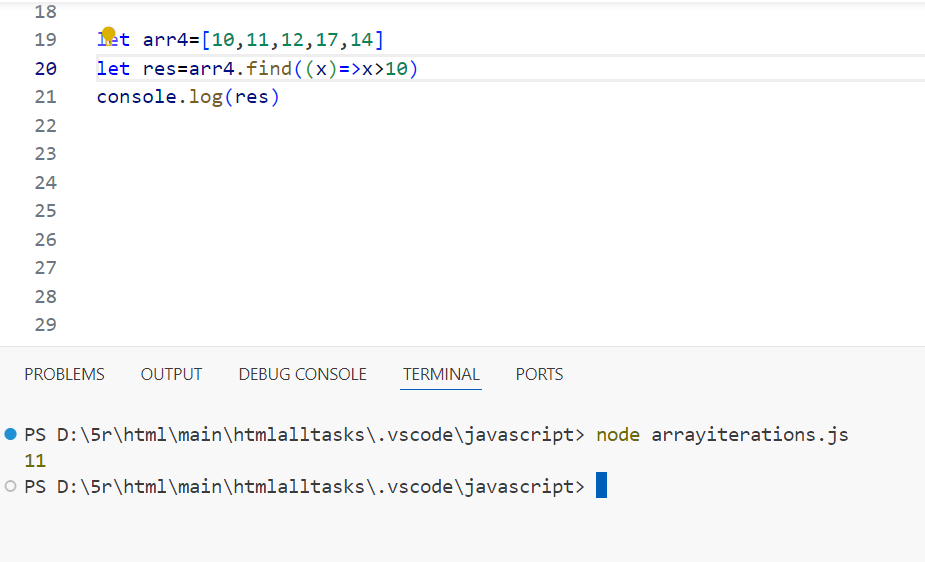


In this method all conditions are satisfy then it will print .even one condition is fail it will not give the output.

find() method:

The **find()** method in JavaScript is used to return the value of the first element in the array that satisfies the provided testing function. It searches the array from left to right, stopping once it finds an element that passes the test. It returns **undefined** if no such element is found.

Here is the example:



It will give the first value when condition is satisfied.

Reverse() method:

The **reverse()** method in JavaScript is used to reverse the order of the elements in an array. It modifies the array in place and returns a reference to the reversed array. The first element becomes the last, and the last element becomes the first.



It don’t have any callback function.

Sort() method:

The **sort()** method in JavaScript is used to sort the elements of an array in place and returns the sorted array. By default, the elements are sorted in ascending order based on their string Unicode code points.

